JPRS L/10522

17 May 1982

USSR Report

CYBERNETICS, COMPUTERS AND AUTOMATION TECHNOLOGY

(FOUO 10/82)

FBIS FOREIGN BROADCAST INFORMATION SERVICE

NOTE

JPRS publications contain information primarily from foreign newspapers, periodicals and books, but also from news agency transmissions and broadcasts. Materials from foreign-language sources are translated; those from English-language sources are transcribed or reprinted, with the original phrasing and other characteristics retained.

Headlines, editorial reports, and material enclosed in brackets [] are supplied by JPRS. Processing indicators such as [Text] or [Excerpt] in the first line of each item, or following the last line of a brief, indicate how the original information was processed. Where no processing indicator is given, the information was summarized or extracted.

Unfamiliar names rendered phonetically or transliterated are enclosed in parentheses. Words or names preceded by a question mark and enclosed in parentheses were not clear in the original but have been supplied as appropriate in context. Other unattributed parenthetical notes within the body of an item originate with the source. Times within items are as given by source.

The contents of this publication in no way represent the policies, views or attitudes of the U.S. Government.

JPRS L/10522

17 May 1982

USSR REPORT

CYBERNETICS, COMPUTERS AND AUTOMATION TECHNOLOGY

(FOUO 10/82)

CONTENTS

HARDWARE

UDC 681.327.07

FLOPPY DISK STORAGE CONTROLLER FOR WORK PLACE AUTOMATION SYSTEMS

Kiev TEKHNICHESKIYE SREDSTVA MINI- I MIKRO-EVM in Russian 1981 (signed to press 19 Aug 81) pp 3-10

[Article by Yu.S. Yakovlev, Ye.T. Makovenko, F.A. Tsventukh, A.A. Yurasov, A.T. Makovenko and M.A. Gayvoronskiy from collection of articles "Mini- and Microcomputer Hardware", edited by Ukrainian SSR Academy of Sciences Corresponding Member B.N. Malinovskiy, Institute of Cybernetics, Ukrainian SSR Academy of Sciences, 500 copies, 80 pages]

[Text] The effectivness of the application of mini- and microcomputers, as well as of systems based on them, is determined to a considerable extent by the characteristics of external memories. Floppy disk storages have been widely used as external memories in recent years.

The popularity of their use has been determined by the fact that floppy disk storages combine in themselves functions both of an external memory and of data preparation units, which makes it possible to a considerable extent to use software in microprocessor systems which is similar to that developed for large computing systems, such as efficient operating systems, compilers from high-level languages, assemblers, libraries of standard subroutines and user's programs, and much more [1]. This makes it possible in turn to organize on the basis of microprocessor hardware inexpensive and flexible automated work place (ARM) systems. Their main purpose is the input, processing, output and display of alphanumeric and graphic information (e.g., for developing and drawing circuit diagrams, laying out and making photomasters of printed circuit boards, technical and technological preparation for production and the like).

In automated work places floppy disk storages are widely used for storing compilers from high-level languages or assemblers, operating systems, input/output utilities, emulators of exchange protocols and packages of applied programs. A great number of automated work places which use floppy disk storages are known, such as the "Event 2000" device from the Applied Data Communications firm, the IZOT-250 "Byurokomp'yuter" [Office Computer] produced in the People's Republic of Bulgaria, the Zentec "ZMC-50", etc.

The use at an automated work place of a large-capacity ROM or tape cassette storages is inadvisable [2], since in the first case flexibility (e.g., of the compiler) is reduced along with the possibility of the adaptation of software, and in the second the low speed does not make it possible to solve an entire series of problems having strict time limitations (e.g., emulators of systems exchange protocols). Cassette memories are also inconvenient for storing compilers in connection with the overlay structure of compilers.

For using floppy disk storages in microprocessor systems as, besides, any other external memories, also, it is advisable to develop and create controllers making it possible to relieve the processor to the maximum from the functions of controlling these units, since they make it possible to construct operating systems without reference to a specific type of storage.

At the present time in CEMA countries several types of floppy disk storages are being developed which have to a considerable extent a unified interface and similar technical characteristics. The international standard for recording on an ISO track is usually used in these storages [1].

The exchange of information between the central processor (TsP) and the controller can be accomplished in the program control mode, interrupt mode, or in the direct-access channel (KPDP) mode. The relatively high speed of magnetic disk storages requires the use of the KPDP mode. In this case the controller must obtain from the central processor information on the address of the location from which readout begins or into which writing of the data field of a sector begins.

The completion and correctness of the execution of an operation can be checked through an interrupt line and/or by interrogation by the central processor of the controller's state.

Analysis of the information required for controlling a floppy disk storage makes it possible to single out the following key functions of a controller:

1) Matching central processor and floppy disk storage interfaces.

2) Receiving and decoding instruction information and sending out state signals.

3) Controlling positionining of the head and scanning the state of readiness of storages.

4) Synchronizing with the magnetic medium in executing operations.

5) Controlling the operating mode of the storage (read/write).

6) Converting a parallel code into a serial and vice-versa. Forming write signals in the execution of operations involving writing, as well as separating data and synchronizing pulses in reading information out from a disk.

7) Generating and verifying the cyclic check code.

8) Coordinating the performance of all functions.

Of course, the hardware and software implementation of the functions listed above can be quite different;

Using "hard" logic.

By means of microprogrammable controllers executed with integrated circuits with a medium degree of integration, general-purpose microprocessors and special-purpose microprocessors.

The use of "hard" logic makes it possible to construct controllers for high-speed disk units. In addition, an analysis of the operating algorithms of controllers for floppy disk storages has shown that such a controller must execute long control sequences, which requires the storage of a great number of states. In addition, the implementation of these control sequences with "hard" logic results in considerable functional redundancy because of the complexity of the multifunctional use of elements and units. For the creation of a controller employing "hard" logic 300 to 400 integrated circuits with a medium degree of integration are now required [3].

The feasibility of the microprogram principle of implementing controllers for floppy disk storages is obvious from the above. In this case sequences of the states of the controller are assigned in the form of chains of microinstructions and the number of functional units can be small, since (if the speed makes this possible) their multifunctional use is possible. For example, the majority of functions associated with positioning, as well as with determining the position of the head instantaneously, can be performed by microprogram.

A controller for a floppy disk storage can be implemented on the basis of single-chip sectionalized large-scale integrated circuits (LSIC's) of microprocessors, as well as by using special-purpose interface LSIC's.

A controller employing a single-chip microprocessor can have a high logical and computing capacity. This makes it possible to hand over to these controllers a number of functions of the operating system relating to the organization of input/output and controlling the distribution of the external memory. However, the use of these microprocessors for constructing controllers is hampered at the present time because of the insufficient speed of a microprocessor for the functions of processing and converting data, the necessity of including in the controller's structure additional units not only for performing control functions, but also for processing information, the requirement of analyzing a great number of conditions and state signals, the need to produce a great number of control signals (20 to 30 just for directly controlling the storage), and the complexity of matching the microprocessor's interface with the remaining equipment of the controller.

Controllers for floppy disk storages based on sectionalized microprocessors have found extensive application (e.g., the SBC-201 controller based on the Intel-3000 [4]).

Special-purpose LSIC's for interfacing with floppy disk storages have appeared in recent years abroad (e.g., the Intel 8271, Motorola 6843 and Rockwell 10936 [3]). However, the complexity of the formation of write signals, difficulties in separating data and the diversity of floppy disk storages have been responsible for the fact that a number of functions relating to interfacing with a central processor interface, to the recognition of data, to controlling the storage and certain others must be implemented with additional integrated circuits.

It can be concluded from the above that for the purpose of constructing a controller from floppy disk storages it is advisable to use the series K589 microprocessor set. A controller which is being introduced, developed on the basis of this series for controlling storages using type PLX45D floppy disks and included in the memory system of an automated work place, contains the following units: a system instruction unit, a unit for interfacing with the common bus, a central processor element unit, a microprogram control unit, a direct-access channel (KPDP) unit, a data processing unit and a disk drive control unit.

The system instruction unit recognizes the controller's address and receives and decodes instructions from the central processor.

Interfacing of the controller with the central processor and computer's working storage is accomplished by means of the unit for interfacing with the common bus.

The microprogram control unit receives and decodes instructions of the central processor (three out of seven instructions) and also organizes the accessing from the memory of the controller's control information required for performing disk operations. It makes possible the interpretation of three instructions of the central processor and seven instructions of the controller and controls the operation of all units of the controller. A K589IK01 LSIC is used for controlling the accessing of microinstructions. The capacity of the microprogram memory is 512 32-bit words. The microprogram memory is implemented with a K556RYe4 LSIC.

The central processor element unit is executed with type K589IK02 microcircuits. Besides this it contains multiplexers for inputing information through data lines and masks. Output of the results of the execution of logic operations and comparison operations, as well as the input of conditions from the microprogram control unit, are accomplished through control character lines. The results of processing are transferred either to the central processor or to the data processing unit.

The KPDP unit organizes the exchange of information with the central processor's memory through a direct-access channel.

The data processing unit performs the recognition of address labels and receives sequences of bits read out from the disk and organizes them into bytes. During write-in it converts bytes of information into a sequence of bits for recording on tracks. In addition, it generates two bytes of a cyclic check code by dividing each heading field and sector data field by a generating polynomial of the $X^{16} + X^{12} + X^5 + 1$ type. In recording the heading field on a track the two bytes of the cyclic check field are recorded at the end of the heading field, and in recording the data field at the end of the data field.

In readout a cyclic check is performed by comparing the two bytes of the cyclic check code recorded on the track and the bytes generated in the readout process.

The data processing unit is interfaced with the common bus through the central processor element unit. During readout the unit's circuits are synchronized by synchronizing pulses received from the disk drive, and during write-in by a quartz-stabilized write signal generator.

The unit for controlling the disk drive generates the necessary signals for controlling the floppy disk storage and also receives state signals from the storage.

All of the controller's operations are initiated by the microcomputer's central processor. For specifying each operation of the floppy disk storage, the central processor forms in the microcomputer's RAM a control data block (BUI) containing 10 bytes, which are presented in table 1.

Table 1.

No	Type of byte	No	Type of byte
1.	Channel word	6.	Address of data buffer (lower-order byte)
2.	Operation word	7.	Address of data buffer (higher-order byte)
3.	Number of entries	8.	Number of operations in chain
4.	Address of track	9.	Address of next BUI (lower-order byte)
5.	Address of sector	10.	Address of next BUI (higher-order byte)

The channel word byte contains the information necessary for organizing control of the operating modes of the controller together with the microcomputer (control of interrupt, execution of chain or individual operations, control of order of performance of operations, assignment of length of information word, etc.).

The microcomputer's central processor is linked with the controller by input/output ports by execution of the following seven instructions, which are given in table 2.

Table 2.

Microcomputer central processor instructions

No	Designation	Instruction
1.	ZMA	Write lower-order bits of RAM address
2.	ZSA	Write higher-order bits of RAM address and start operation of disk
3.	OTsO	Stop chain of operations
4.	SKN	Reset controller
5.	ChSK	Read state of controller
6.	ChTR	Read type of result
7.	ChBR	Read byte of result

[Continued on following page]

Controller's operations

No	Operation
1.	Retrieval
2.	Formating
3.	Recalibration
5. [as published]	Verification of cyclic check
6.	Write data
7.	Write erased data

The initial address of the control data unit is assigned and the operation of the controller is started by means of the ZMA and ZSA instructions of the central processor. After startup, the controller performs all operations without the participation of the microcomputer's central processor. The controller accesses bytes of the control data unit for the performance of operations independently through the memory direct access channel.

The SKN instruction sets all of the controller's circuits to the inital state. The OTsO instruction is used for stopping execution of the chain of operations of the controller.

By means of the ChSK instruction the central processor receives information on the attendance of the controller, the state of readiness of drives and the presence of an interrupt enabling signal.

Through the ChTR instruction the controller reads out the code for the type of result, by means of which the central processor determines the nature of the information contained in a byte of the result and the number of incompleted operations in a chain.

Through the ChBR instruction the controller reads out the byte of the result in which, depending on the code for the type of result, information is contained on the readiness of drives or on the nature of errors.

The controller performs seven operations assigned by the control data unit's operation word. The list of controller operations is presented in table 2.

After the execution of a single operation or chain of operations (depending on the code for controlling interrupts, which is assigned in the control data unit's channel word) the controller puts out an interrupt enabling signal. An interrupt enabling signal is put out also after the detection of an error in the process of the execution of the current operation and with a change in the state of readiness of the drive selected. The central processor receives the interrupt enabling signal from the controller either through an interrupt enabling line or through cyclic execution of the ChSK instruction. After receiving the interrupt enabling signal the central processor determines the reason for the interruption by sequential execution of ChTR and ChBR instructions. In addition, by means of the same instructions the central processor picks up the interrupt enabling signal from the controller, after which it can assign new operations.

The controller has definite universality and with slight modifications of micro-program and hardware facilities can be used for controlling various types of floppy disk storages.

The controller which has been developed can be widely used not only in constructing memory systems oriented toward use in automated work places, but also in other microprocessor systems, in particular in systems for debugging microprocessor facilities, in measuring and data processing systems, etc.

Bibliography

1. Bogachev, A.V., Kolomiyets, G.S. and Fedotov, V.P. "Floppy Disk Storages," UPRAVLYAYUSHCHIYE SISTEMY I MASHINY, No 3, 1979, pp 62-68.

2. Davis, S. "CRT Terminals--Programmability's the Key to YQ," EDN, June 5, 1978, pp 75-81.

3. Ogain, Carol A. "A Floppy Disk Interface is More Than a Chip," EDN, Vol 23, No 15, 1978, pp 115-119.

4. "Intel Data Catalog, 1977," pp 10/228-10/231.

8831
CSO: 1863/114

UDC 621.391

CONNECTION OF INFORMATION SOURCES TO UNIFIED-SERIES COMPUTERS BY MEANS OF YeS-6022 UNIT

Kiev TEKHNICHEKSIYE SREDSTVA MINI- I MIKRO-EVM in Russian 1981 (signed to press 19 Aug 81) pp 34-41

[Excerpts from article by V.Ye. Reutskiy and B.V. Smirnov from collection of articles "Mini- and Microcomputer Hardware", edited by Ukrainian SSR Academy of Sciences Corresponding Member B.N. Malinovskiy, Institute of Cybernetics, Ukrainian SSR Academy of Sciences, 500 copies, 80 pages]

[Excerpts] In carrying out scientific research and experiments the necessity often arises of connecting to a YeS [Unified Series] computer various devices which do not have an output for the input/output interface of this computer.

The implementation of a unit for mating with a YeS computer input/output interface requires considerable hardware costs (cf. description of a YeS computer interface in [1]). In addition, the situation often arises when some input/output units included in the structure of the complex to be used are underutilized. Therefore, it is of interest to consider the possibility of using the equipment of poorly utilized input/output units included in the structure of YeS computers for the purpose of connecting new external sources of information.

In [2] the possibility is discussed of using the equipment of the YeS-5517 unit for controlling tape storages for the purpose of connecting minicomputers to YeS computers. In this study the possibility is discussed of connecting information sources to YeS computers by means of the YeS-6022 unit for inputing data from punched tape.

This unit is as a rule poorly utilized in the structure of a YeS computer, since the loading of working programs into the YeS computer's working storage is performed from the YeS-6012 input unit (by means of punched cards). Punched cards are basically used as the information medium, since the correction of data entered on punched tape involves certain difficulties.

The YeS-6022 unit is designed for reading data entered onto punched tape in the form of perforations. It matches the information and physical characteristics of signals of a type FS-1501 photoelectric reader and signals used by a YeS computer interface (a description and the characteristics of the YeS-6022 and FS-1501 are given in [3]).

In certain cases it is necessary to connect to a YeS computer information sources the characteristics of whose signals do not fully agree with the characteristics of FS-1501 signals. The requirement of the complete simulation of the information and physical characteristics of the FS-1501's signals on the part of the connected information source is responsible for added hardware costs and does not make possible full utilization of the speed capabilities of the YeS-6022 unit and the YeS computer's processor.

Matching of Signal Levels

The logic section of the YeS-6022 unit is constructed with series 155 logic elements. The FS-1501 unit has "Stop" and "Start" signal levels in the following ranges: level of logical "0"--E ± 2 percent V; level of logical "1"--not less than 0.4 V; and of output signals (information--FD1 to FD8, the synchronizing signal and "Tape Loaded" signal): level of logical "0"--in the range of E ± 20 percent V, and level of logical "1"--not less than -1 V, where E is the supply voltage, which, depending on the type of FS-1501, can be -6.3 or -12.6 V.

Converters of the levels of FS-1501 signals into the logical levels of series 155 elements and vice-versa are distributed in the YeS-6022 in individual TEZ's [expansion unknown]. The replacement of these TEZ's by those developed for the information source to be connected makes it possible to connect various information sources to the YeS-6022.

If the levels of the input/output signals of the information source to be connected match series 155 levels and the distance between unit makes it possible to manage without the amplification of signals, it is sufficient in place of these TEZ's to put in plugs with jumpers between the input contacts and output contacts corresponding to them.

Conclusion

The use of the YeS-6022 unit for connecting new items to a YeS computer makes it possible on the one hand to utilize this unit more effectively and on the other eliminates the need to develop a unit for mating with a YeS computer input/output interface.

There are certain restrictions on the class of units which can be connected at the YeS-6022 end. The initiative for the input of data into the processor via the YeS-6022 unit belongs to the processor. As the result of this, it is advisable to connect to the YeS-6022 passive sources the transfer of data from which is performed after the processor issues a request for the transfer of data. In addition, as was indicated above, the information source connected to a YeS computer by means of the YeS-6022 must itself form the time parameters of output signals. If the unit connected is oriented toward the transfer of data in the request-response mode, additional hardware costs are required to implement the response signal.

A television camera for the input of video information at a speed of 234K bytes, a controlled system interface--the "Sektor" USO--for inputing information at a speed on the order of 20K bytes, and a unit for the transmission of data through a telephone communications channel at an input speed on the order of 1K to 3K bytes have been connected to a YeS computer by the method suggested. With these units connected to it, the utilization of the equipment of the YeS-6022 unit equals approximately 80 percent.

8831
CSO: 1863/114

UDC 681.327.28

ASPECTS OF USE IN FAST FOURIER TRANSFORM PROCESSOR OF MEMORY UTILIZING DYNAMIC METAL-INSULATOR SEMICONDUCTOR LARGE-SCALE INTEGRATED CIRCUITS

Kiev TEKHNICHESKIYE SREDSTVA MINI- I MIKRO-EVM in Russian 1981 (signed to press 19 Aug 81) pp 50-53

[Article by V.V. Zvyagintsev, B.I. Pavlus', A.P. Romantsov, V.I. Pis'mak and S.G. Bogdanov from collection of articles "Mini- and Microcomputer Hardware", edited by Ukrainian SSR Academy of Sciences Corresponding Member B.N. Malinovskiy, Institute of Cybernetics, Ukrainian SSR Academy of Sciences, 500 copies, 80 pages]

[Text] The advantage of memory elements of the dynamic type employing MIS [metal-insulator semiconductor] structures, consisting in a low power requirement as compared with memory elements of the static type, often turns out to be a primary factor in solving the problem of the choice of type of memory for computer hardware under development. However, when using a memory employing MIS elements of the dynamic type (a DOZU) it is necessary to take into account the need of such a memory for the periodic regeneration of stored information [1].

This fact touches upon two aspects of development--the time characteristics of the unit to be created and its hardware implementation. Expenditures of time for the regeneration of information in a DOZU inevitably result in a loss in efficiency. This loss is evidenced to a greater extent in high-capacity units with a random distribution of information in the memory's address space.

Usually memory elements in DOZU microcircuits are arranged by lines and columns, forming an array of 1-bit memory elements with a built-in control circuit. The regeneration of information takes place simultaneously over the entire line of memory elements among which the accessed memory element resides. Sequential checking of the DOZU's lines makes possible regeneration of the entire array of memory elements.

The operation of a DOZU without regeneration is possible upon the condition that the time spent on accessing all lines of the DOZU does not exceed the time for storing information in the memory elements:

$$T \geq T_s \cdot n ,$$

where T is the time for storing information in the memory elements, T_s is the time spent on accessing memory elements of a single line and n is the number of lines.

Thus, from the speed of response of hardware for implementing a fast Fourier transform (BPF) algorithm and from an analysis of addressing of the memory it is possible to provide an estimate of the possibility of the operation of a DOZU without regeneration.

If the regeneration of information is regarded as a process of the memory's internal control unit's accessing the array of stored information, then in operation of the unit, generally speaking, conflict situations are inevitable because of simultaneous accessing of data on the part of the operating unit and the internal control unit. These situations can be resolved by determining the priority of access to the memory, which can be based on certain flow control algorithms, in particular, attendance to requests in the order of their arrival and priority servicing. The specific algorithm will be determined by a number of factors, such as the purpose of the computing unit, its instruction set and the characteristics of sets of data (control information or data, their format, length, etc.).

For the above reasons, when using a DOZU stages in the development of operating and memory units are characterized by more intense interaction than in the case of using memory units which do not require the regeneration of information. In the latter instance for accessing the memory it is necessary to indicate a standard set of signals--read or write, accessing signal, the address for the allocation of data; and in writing, the code of the information to be written. In the case of a DOZU these signals must be supplemented by information on the state of the memory--"free" or "occupied" in performing regeneration--as well as by signals for the control of regeneration on the part of the operating unit, among which it is generally possible to include the following: a signal for enabling regeneration in the information storage mode, a signal for enabling regeneration in the information exchange mode, a signal for halting regeneration with its subsequent resumption from an advanced address or from the beginning of an address space, and a signal for forcing the memory into the "free" state. The memory, in turn, must be able to recognize the above-indicated information and to react to it in an appropriate manner.

In determining the specifics of the operation of a DOZU as a component of a fast Fourier transform processor and the required set of control signals, two possible operating modes of the processor were primarily used--the calculation mode and the mode of the exchange of information with a computer of the YeS [Unified Series] type through a selector channel. In the calculation mode the processor sequentially accesses elements of the array of numbers from the memory and records them after processing. Here there is no need to halt computations for carrying out regeneration, since regeneration is performed in readin/readout of the array on account of sequential accessing of the lines of the DOZU in keeping with the graph of the fast Fourier transform's algorithm, and the time for checking all lines turns out to be less than the permissible time for the storage of information in memory elements without its regeneration. In the information exchange mode, in reading out an array from the memory of the fast Fourier transform processor and in its subsequent transfer to the computer the situation is similar to that described above. As far as the write-in of an array of information from the computer into the memory of the fast Fourier transform processor is concerned, the need to perform the binary inversion of addresses results in a considerable increase in the time for checking the DOZU's lines and makes it necessary to halt exchange for the purpose of restoring the information previously found in the memory of the fast Fourier transform processor.

For the purpose of accomplishing proper interaction the operating unit generates the following signals for controlling the memory in addition to the standard signals usually used:

Enabling regeneration in the storage mode. This signal means that the fast Fourier transform processor does not perform any actions associated with computations or exchange and the DOZU can perform regeneration independently.

Enabling regeneration in the exchange mode. This signal means that the priority of the request for access to the memory on the part of regeneration circuits is higher than the priority of the operating unit's request. Through this signal the processor halts the reception of information from the computer and the memory begins to carry out regeneration. After the completion of regeneration the processor renews the write-in of information into the memory of the fast Fourier transform processor from the address which was fixed at the moment of the start of regeneration. This signal is generated cyclically with the time resolution necessary for transferring a single byte of information.

A signal for blocking regeneration is generated when it is necessary to inhibit regeneration or stop it if it has already begun. Through this signal the memory goes into the "free" state and the fast Fourier transform processor begins to exchange information with the computer.

The implementation of the above-described interaction between the memory and operating unit will make it possible to organize the effective operation of a fast Fourier transform processor and to minimize losses in efficiency resulting from the need to carry out the regeneration of information in a memory employing dynamic MIS large-scale integrated circuits.

Bibliography

1. Keylbotta, S. "Aspects of Designing Systems with Dynamic Memories," ELEKTRONIKA, No 3, 1978, p 73.

8831
CSO: 1863/114

UDC 681.31

TYPICAL MULTICOMPUTER STRUCTURES AND ARCHITECTURE OF THEIR SOFTWARE

Kiev TEKHNICHESKIYE SREDSTVA MINI- I MIKRO-EVM in Russian 1981 (signed to press 19 Aug 81) pp 58-63

[Article by V.Ye. Gorskiy, V.P. Pavlov and V.I. Shyaudkulis from collection of articles "Mini- and Microcomputer Hardware", edited by Ukrainian SSR Academy of Sciences Corresponding Member B.N. Malinovskiy, Insitute of Cybernetics, Ukrainian SSR Academy of Sciences, 500 copies, 80 pages]

[Text] Multicomputer complexes created on the basis of various kinds of hardware and software, e.g., SM [International System of Small Computers] and YeS [Unified Series] computers, are designed to improve the efficiency of the entire complex of equipment in complicated computing and data processing systems and integrated ASU's [automated control systems]. An analysis of hardware and software capabilities has demonstrated that such an improvement in efficiency is possible by creating on the basis of micro- and minicomputers group input/output units, user stations and intelligent programmable multiplexers and concentrators.

It is possible to single out the following typical structures on the basis of the capabilities of existing hardware and software which can be used in multicomputer complexes and the prospects for its development:

Local multiterminal complexes in which mini- and microcomputers of the SM-4 and SM-1800 class are used as program multiplexers making possible the implementation of various procedures received in packages of the software of the upper-level central computer.

Distributed multicomputer complexes in which mini- and microcomputers are used as terminal stations, switching processors, etc.; these complexes have a net organization.

The uniting of various kinds of computers for organizing the typical structures mentioned requires the solution of a number of first-priority problems, among which can be included:

Enabling the hardware interfacing of computers.

Development of the structure and methods of implementing software meeting the requirements of the joint operation of computers.

The unification of information systems, including the data banks of various computers.

Matching of data representation formats.

Questions relating to the hardware interfacing of computers are solved, as a rule, by employing the sequential (synchronous or asynchronous) transmission of data through communication lines meeting standards V.24, RS-232 or X.21. The utilization of similar standards in communications equipment produced by domestic industry makes it possible to solve this problem and is discussed in [1, 2].

In solving the problem of organizing a software interface for various kinds of computers, the requirement of the maximum utilization of the software existing in them is used as the basis. The approach most intelligent in this sense must make it possible for the interface software to grow and become more complicated as the system develops.

Let us consider the requirements for software for interaction between computers in multicomputer complexes using four variants as an example (figs 1 to 4).

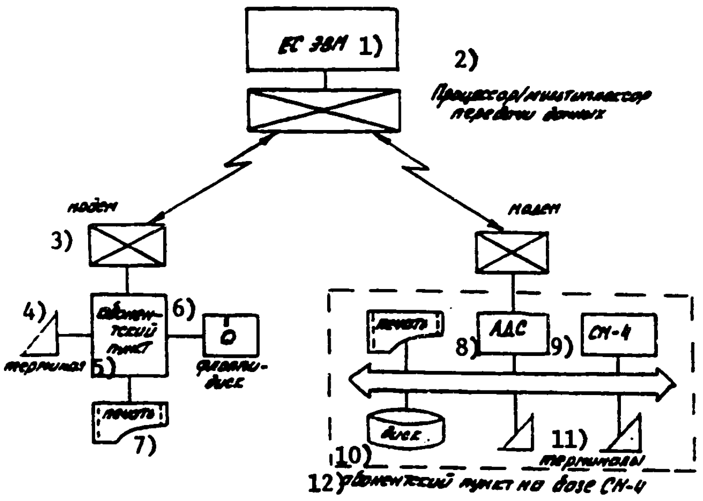


Figure 1. Implementation of User Station Based on SM-4

Key:

1. YeS computer	7. Printout
2. Processor/data transmission multiplexer	8. ADS [automatic switching system]
3. Modem	9. SM-4
4. Terminal	10. Disk
5. User station	11. Terminals
6. Floppy disks	12. User station based on SM-4

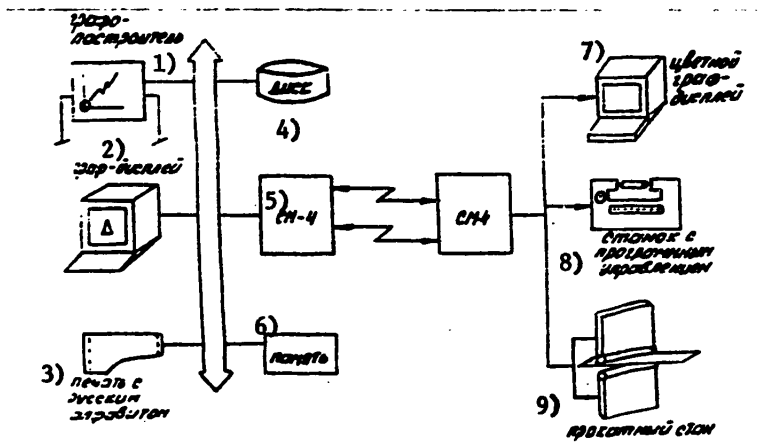


Figure 2. Division of Unique Peripheral Equipment Between Two SM-4's

Key:

1. Graph plotter
2. Graphic display
3. Russian alphabet printer
4. Disk
5. SM-4
6. Memory
7. Color graphic display
8. Machine tool with program control
9. Rolling mill

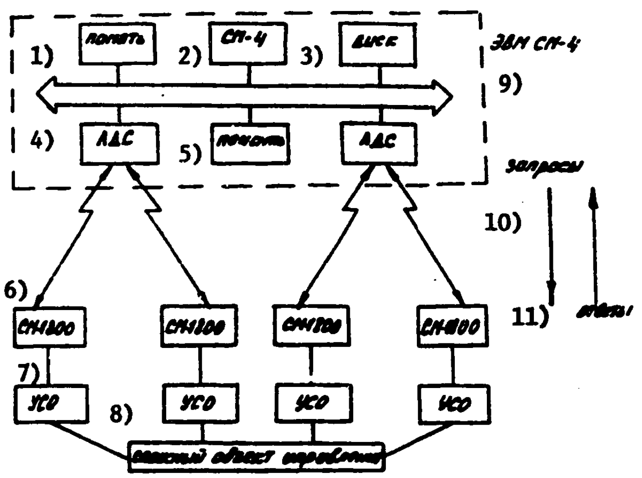


Figure 3. Centralized Control Based on an SM-1800 of Microprocessor Controllers for Controlled System Interfaces

[Key on following page]

Key:

1. Memory
2. CM-4
3. Disk
4. ADS
5. Printer
6. SM-1800
7. USQ [controlled system interface]
8. Complex controlled system
9. SM-4 computer
10. Requests
11. Responses

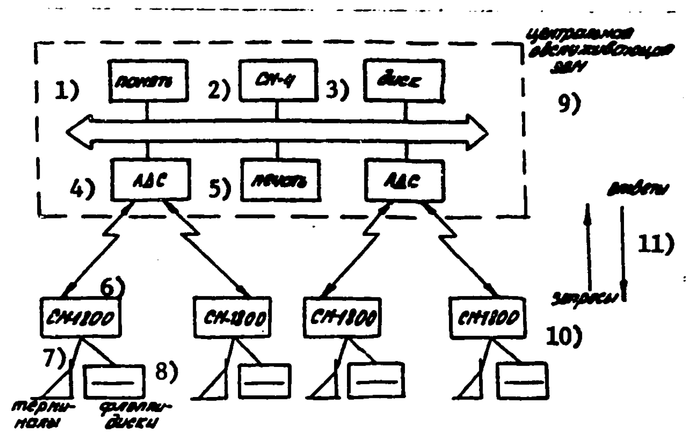


Figure 4. Intelligent SM-4 Terminals Based on SM-1800 Microcomputers

Key:

1. Memory
2. SM-4
3. Disk
4. ADS
5. Printer
6. SM-1800
7. Terminals
8. Floppy disks
9. Central servicing computer
10. Requests
11. Responses

Communication between large computers and computers of smaller capacity is necessary in order to offer low-level users the computing resources of the systems's nucleus (central processor, external storages, unique peripheral equipment such as rapid printers, color graphic displays, graph plotters, etc.). Although in this case the swapping of data is two-way, the low-level computer is always the initiator of communication. Furthermore, as a rule, there is no strict real-time mode requiring high priority in engaging the communications channel. The requirements of the communications network can be formulated as follows:

Enabling the exchange of files between peripheral units of a small computer and the host computer.

Offering facilities for initiating on the host computer tasks transferred from a small computer.

Enabling access to the host computer's data bank, etc.

As demonstrated by analysis, these requirements are in accord with the requirements for user stations based on minicomputers.

In mating computers (fig 2) whose computing resources can be compared, the main problem is to make possible the mutual utilization of unique peripheral equipment, as well as the specific capabilities of operating systems. The most interesting results can be gotten in the case when any unit in a multicomputer system has access to any processor. Thus, it is necessary to make possible in the network the access of any processor to any peripheral unit of the system regardless of their point of connection, the transfer of files (both character and binary) between peripheral units, and the interaction of processes taking place in various computers.

The computing capabilities of SM-1800 microcomputers make it possible to use them for very different purposes. Let us consider two cases:

1) An SM-1800 as an intelligent control unit for controlled system interfaces and connected to an SM-4 (fig 3).

2) An SM-1800 as the intelligent terminal of an SM-4 (fig 4).

These are the most typical applications of a microcomputer connected to a greater-capacity complex. The requirements are formulated as follows in the first variant:

Enabling the exchange of SM-4 - Sm-18C0 files (transparent mode).

The ability to initialize an assignment for an SM-1800 microcomputer.

Offering facilities for remote loading of the SM-1800 operating system from SM-4 files.

In the second case, when the SM-4 is used in the passive servicing mode, it is necessary to provide for the following:

Control of assignments upon the initiative of the SM-1800.

The exchange of character files between peripheral units.

Facilities for remote loading of the SM-1800's operating system from SM-4 files.

An analysis of the requirements presented for the interaction of computers singles out as the main networks the "small computer - large computer" and "microcomputer - small computer" (second variant). In both cases the requirements for exchange between levels agree and its implementation is based on the principle of the simulation of terminals (user stations). In both cases it is necessary to make possible the preparation of data at the lower level and the upper-level computer is connected only to process them. Thus, the problem reduces to creating appropriate software simulating the work of a human being at a console. Obviously, this problem is fairly complicated but it is necessary here to choose a reasonable division of labor between a human being and the simulation program. It is most important to free the programmer from routine operations in transferring files to the upper level, while questions relating to controlling tasks and starting programs must be controlled by the programmer. In this sense it is appropriate to provide a user from a terminal a transparent interface, offering him facilities for directing to peripheral equipment messages to be received or transmitted (fig 5).

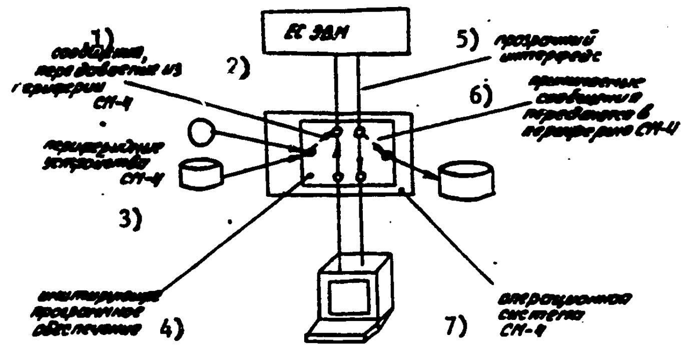


Figure 5. Principle of Design of Software of User Station Based on Minicomputer (Microcomputer)

Key:

1. Messages to be transmitted from SM-4 peripheral units	5. Transparent interface
2. YeS computer	6. Received messages are transferred to SM-4 peripherals
3. SM-4 peripheral units	7. SM-4 operating system
4. Simulation software	

The advantage of this approach is the lack of any modifications of the large computer's software (e.g., that of a YeS computer), which makes possible rapid implementation of the system. The disadvantages are as follows:

The communications channel is monopolized; therefore, the resources of the host computer are accessible to SM-4 users only by turns.

Algorithms for checking the correctness of transferred data are lacking if they are not maintained in the terminals which are simulated.

It is fairly complicated to simulate all the modes of terminals of a large computer on an SM display (e.g., various intensities of characters, screened zones, etc.), as the result of which working with some large-computer program systems oriented toward these modes is complicated.

The solution of problems of the interaction of computers of the same class depends on which computer is the leader, i.e., the control computer relative to the other.

In case of the equality of computers the solutions suggested here will not satifsy the requirements set and their implementation is possible within the framework of a net architecture.

When it is necessary to provide access of the processes of one computer to the peripheral equipment of another, simulation software is implemented in the first computer and at the other's end facilities are employed which can be accessed from terminals. This makes it possible to activate standard programs making it possible to swap data between a terminal and any peripheral unit. The reverse is also true, since all resources of the system are accessible to terminals of the first computer. In the case when access is necessary both upon the initiative of the first computer and on the initiative of the second, the simplest solution can be the cross simulation of terminals with the use of two independent communication lines. This can be used with a short distance between computers when expensive data transmission equipment is lacking (modems, coaxial cables, etc.); otherwise the mutual simulation system does not withstand criticism from the economic viewpoint. The structure of the software of a communications network for equivalent computers is shown in fig 6, using an SM-4 and Siemens-330 as an example.

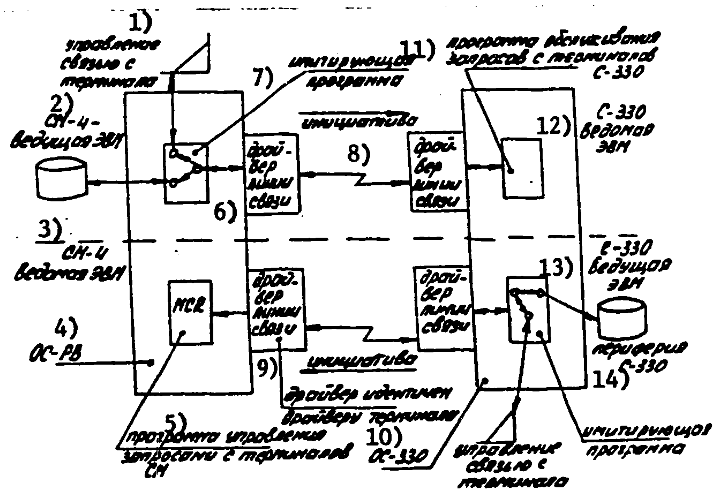


Figure 6. Structure of Software for SM-4 - Siemens-330 Interaction

Key:

1. Control of communication from terminal
2. SM-4--controlling computer
3. SM-4--controlled computer
4. Real-time operating system
5. Program for controlling requests from SM terminals
6. Communications line driver
7. Simulating program
8. Initiative
9. Driver identical to terminal driver
10. Siemens-330 operating system
11. Program for servicing requests from Siemens-330 terminals
12. Siemens-330--controlled computer
13. Siemens-330--controlling computer
14. Siemens-330 peripherals

Of interest is a system in which an SM-4 uses microcomputers as intelligent controllers of units for communicating with a controlled system (e.g., a group of machine tools, an automatic line, etc.). It is obvious that in the case of the coupling of controlled processes in the controlled system a decision for interdependent parameters must be made at the SM-4 level. This means that the microcomputer must function in the mode of subordination to the SM-4, unlike the variant in fig 4.

Thus, summarizing the above, it is possible to draw the following basic conclusions:

1. The presence in the structure of the operating systems of modern computers of developed facilities for interaction between users and terminals makes it possible to construct multicomputer systems with a radial architecture by simulating them by means of a complex of hardware and software.

2. In constructing systems according to this principle, special-purpose software must be implemented only for the computer controlling the exchange. The subordinate computer operates in the mode of passive servicing of the requests of the control computer.

3. The capabilities of the system produced are wholly determined by the capabilities of the dialogue systems whose terminal is simulated on another computer.

These principles for the design of multicomputer systems have been used in creating interaction software for the following complexes (cf. table).

Table 1.

No	Type of computer	Operating system	Type of computer	Operating system	Type of interface
1	SM-3,4	Real-time	YeS-1060	YeS	USVM [computer interface]
2	SM-3,4	Real-time		V-2000	ADS
3	SM-3,4	Real-time	SM-3,4	Real-time	Duplex register

The results of experimental operation have demonstrated that the approach suggested makes it possible to implement fairly rapidly the required special-purpose software making possible the total utilization of existing software systems. In systems No 1 and 2 minicomputers operate in the mode of an intelligent terminal of upper-level machines and in system No 3 the mutual simulation of terminals is made possible (a symmetric system).

Bibliography

1. Naumov, B.N., editor. "Malyye EVM i ikh primeneniye" [Small Computers and Their Application], Moscow, Statistika, 1980, 230 pages.

2. Glushkov, V.M., editor. "Seti EVM" [Computer Networks], Moscow, Svyaz', 1977, 279 pages.

8831
CSO: 1863/114

UDC 681.327

MICROPROGRAM CONTROLLER FOR FLOPPY DISK STORAGES

Kiev TEKHNICHESKIYE SREDSTVA MINI- I MIKRO-EVM in Russian 1981 (signed to press 19 Aug 81) pp 63-70

[Excerpt from article by V.A. Cherepanov from collection of articles "Mini- and Microcomputer Hardware", edited by Ukrainian SSR Academy of Sciences Corresponding Member B.N. Malinovskiy, Institute of Cybernetics, Ukrainian SSR Academy of Sciences, 500 copies, 80 pages]

[Excerpt] In this article a description is given of the principles of designing microprogram controllers for floppy disk storages for mini- and microcomputers used in the development of such a controller for the SM-3 and SM-4 UVK's [process control computer complexes].

The following requirements were put forth in designing the microprogram floppy disk memory:

1. The possibility of using it in other systems by replacing the unit for linking with a system interface.

2. Potential possibility for use in other applications in which controllers of the serial transfer of data are required (e.g., for controlling tape cassette memories).

3. Minimizing utilization of the central processor and of system resources.

4. The possibility of organizing the asynchronous transfer of the data of synchronous storages.

Taking the above-listed requirements into account, in designing, work must be performed in the following sequence:

1. Analysis of the specifications for the system interface and of parameters of the floppy disk storage.

2. Development of the operating algorithm for the unit for linking with the system interface.

3. Development of the algorithm for functioning of the microprogram controller.

4. Development of the unit's skeleton diagram.

5. Choice of the microinstruction format and development of the structure of the microprogram controller.

6. Microprogram implementation of the algorithm for functioning of the controller.

7. Simulation of the unit's operation and debugging of microprograms in real time.

Two fundamentally different methods exist for organizing systems based on mini- and microcomputers--the centralized and the decentralized.

The distinctive feature of the first is the effective combining of the equipment of various functional units in the general equipment of the processor and the assignment of logic control functions to the central processor, and the input/output controllers perform various uncomplicated functions.

The second method presumes an architecture for the computer of which the distributed processing of information is characteristic, when logical control is distributed over the entire system for the purpose of the simultaneous non-overlapping execution of various functions.

The floppy disk storage controller which has been developed for a UVK with a "common bus" interface, in terms of the structure of equipment and functions performed, meets the organization requirements of the distributed data processing principle.

8831
CSO: 1863/114

UDC 681.3:61

MINI- AND MICROCOMPUTER HARDWARE

Kiev TEKHNICHESKIYE SREDSTVA MINI- I MIKRO-EVM in Russian 1981 (signed to press 19 Aug 81) pp 2, 75

[Annotation and table of contents from collection of articles "Mini- and Microcomputer Hardware", edited by Ukrainian SSR Academy of Sciences Corresponding Member B.N. Malinovskiy, Institute of Cybernetics, Ukrainian SSR Academy of Sciences, 500 copies, 80 pages]

[Text] Questions are discussed, relating to the design of controllers for controlling floppy disk storages and for displaying and processing information, which are effective in mini- and microcomputers, as well as aspects of the design of minicomputers and of testing minicomputers, and to the hardware implementation of a fast Fourier transform algorithm.

CONTENTS

8831
CSO: 1863/114

UDC 62-50

MINIMIZATION OF SIZES OF FLAT MAGNETIC DOMAINS IN DESIGNING STORAGE DEVICES WITH INCREASED INFORMATION DENSITY

Moscow PROBLEMY UPRAVLENIYA V TEKHNIKE, EKONOMIKE, BIOLOGII in Russian 1981 (signed to press 13 Aug 81) pp 209-216

[Article by S. I. Kasatkin and V. S. Semenov]

[Excerpts] A storage device with flat magnetic domains (PMD) is a magnetic film deposited on a glass base. Low-coercive channels and a high-coercive array are formed in the film. Magnetic domains that are the information carrriers are generated, impelled and erased in the channels. In some versions of domain storage, additional magnetic layers are deposited on the base, which allows improving a number of characteristics, but in turn complicates the manufacturing technology. The magnetic film is combined with one or two layers of conductors, either formed with foil dielectric or deposited on the film itself. In some storage versions, coils are needed to create an external magnetic field. All this is bonded together, screened and placed in a standard case.

Development of magnetic domain storage is currently split in several directions. Most promising are the devices that make use of magnetically hard bands, the tension effect of the domain side wall and the method of advancement suggested by Broadbent [1]. Used for forward motion in the proposed device is a modified Broadbent method that allows domain motion in both directions. The shortcoming of this method compared to the others is the need for two layers of conductors. The advantage is the simpler topology of the low-coercive channels.

A main problem to be solved in developing any storage unit is that of increasing information density. In domain storage devices, increasing information density runs into both technological difficulties and physical limitations. At this stage of storage devlopment, the technological problems are not matters of principle, though they do present definite difficulties. Therefore, maximum information density is now governed by physical factors. The main parameters governing information density can be considered, first, the minimal width of a stable domain which specifies the minimal width of a low-coercive channel, and second, the minimal length of the flat magnetic domain for a given channel width that governs the minimal width of the conductors. The following theory has been suggested for calculating these values.

General Description of Storage Matrix

A storage matrix is a magnetic section bonded with two-sided foil, on which two layers of conductors are deposited. All this is covered by a screen and placed in a standard case.

Matrix capacity	32K bits
Registers	4
Length of augend registers	8K bits
Maximum clock frequency	50 kHz
Maximum data transmission rate	200K bits/s
Maximum power consumption ($f = 50$ kHz)	5 W
Maximum current	600 mA
Read signal (bipolar)	1-2 mV
Case dimensions	30 x 48 mm^2
Contacts	20
Base dimensions	30 x 45 mm^2

8545
CSO: 1863/77

UDC 681.324

STATE OF ART FOR FIBER OPTIC APPLICATIONS IN COMMUNICATIONS

Riga AVTOMATIKA I VYCHISLITEL'NAYA TEKHNIKA in Russian No 1, Jan-Feb 82 (manuscript received 28 Oct 80) pp 34-42

[Article by Ye. D. Bulatov, Yu. V. Grigor'yev, I. V. Kalmykov, V. G. Lomanov, Ye. A. Otlivanchik, A. M. Prokhorov, N. D. Simachev and I. N. Sisakyan: "The Use of Fiber Optic Communications Lines and Integral Optic Elements in Computer Complexes and Networks"]

[Text] A characteristic of computer technology today is that an ever-growing volume of data is being transmitted over great distances with high speed and reliability. In many cases the use of wire and coaxial lines as communications channels no longer meets all the requirements of transmission lines and results in an increase in equipment to monitor errors and relay the transmitted data.

In recent years considerable attention has been devoted to the use of fiber optic communications lines at all levels of data transmissions in computer systems. This is linked to significant advances that have been made in the field of designing low-loss fiber optic cables, and semiconductor emitters and photoreceivers [1-3].

Fiber optic communications lines have a number of unquestioned advantages over wire lines: high noise suppression; ideal galvanic isolation; absence of electromagnetic inductance; resistance to radiation; small dimensions and weight; high heat resistance; and, security of transmitted data. The large pass bands of fiber optic communications lines make it possible to use multiplexing and time or spectrum compaction. Data transmission speeds in this case reach tens of gigabits per second.

Leading companies in the United States, Japan, West Germany, and France have developed and are producing, in addition to powerful equipment for multichannel telephone and television communications, fairly compact fiber optic communications lines to replace wire lines. They are designed for distances of up to 1-5 kilometers and data transmission speeds of up to 50 megabits per second (see Table 1 [4]).

The present article considers the structure of current fiber optic communications lines and prospects for their use in computers, systems for data collection and control of experiments and industrial processes. It also evaluates possible

Table 1.

Company	Model	Type	Maximum Line Length, m	Transmission Speed	Frequency of Error	Voltage and Current Transmitter	Voltage and Current of Receiver	Dimensions of Transmitter and Receiver, mm	Approximate Price $
ITT Electrical Products	T-614/ 615	Simplex	1,500	0-5 Mbauds	10^{8}	+5 V, 200 mA	±5 V, ±75 ma	48 × 20 × 8	750
3 M Co.	3,550	Duplex	100	100 Kbauds- 10 Mbauds	10^{-10}	+5V, 300 mA ±7-±15V	10-25 mA	68 × 32 × 10	695
Plessy	OHL-400	Simplex	2,000	0.1-30 Mbauds	10^{-10}	+5 V, 300 mA ±12 V	150 mA	105 × 28.5 × 33	1,950
TI	TXED455 C025-489	Simplex	50	0-67 Mbauds	10^{-13}	+2 V, 100 mA	5 V	9 × 15 × 12	188
RCA	C-86008E	Simplex	1,500	0-20 Mbauds	10^{-9}	5 V, 250 mA	±5 V, 30 mA	44 × 51 × 25	850
Valtec	TTK	Duplex	1,000	0-10 Mbauds	10^{-9}	115 V	6WT	165 × 114 × 51	1,200
Hewlett-Packard	HFBP-0010	Simplex	100	0-10 Mbauds	10^{-9}	5 V, 125 mA	5 V, 120mA	43 × 16 × 8	570
Siemens	D-10	Duplex	5,000	0-10 Mbauds	10^{-10}	2 wt		125 × 158 × 51	2,500
Nippon Electric	Neolink-10D	Simplex	500	0-10 Mbauds	10^{-9}	5 V, 300 mA	5 V, 100mA	43 × 17 × 17	193
Meret	MDL-4211	Simplex or Duplex	2,000	0-2 Mbauds	10^{-10}	±5 V, 10 mA	± - ±15 V, 30 - 10 mA	13 × 19 × 25	800
Galileo	DL-2	Simplex	1,000	10 Kbauds- 10 Mbauds	10^{-9}	4 wt, 110 V	85 wt, 110 V	68 × 107 × 152	950

modifications of the architecture of computer complexes that use high-speed communications networks with fiber optic communications lines.

First let us consider the capabilities of fiber optic communications lines and their elements (see Figure 1 below). Systems with fiber optic communications

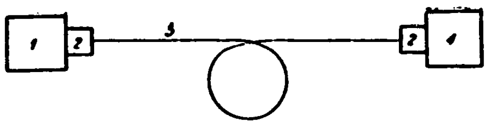


Figure 1. Fiber Optic Communications Line and Its Elements.

Key: (1) Emitter;
(2) Coupler:
(3) Optical Transmission Medium (Fiber Cable);
(4) Receiver.

lines can use cables with one or several fiber strands and cable with a bundle of fibers as the optical transmitting medium. The properties of the cables are determined by the type of fiber. Table 2 below gives averaged specifications of the most widely used types of fibers created in laboratories and by industry.

Table 2

Type of Fiber	Diameter of Light Guide Strand in Microns	External Diameter in Microns	Losses in Decibels per Kilometer	Dispersion in Nanoseconds per Kilometer
Single-Mode	5-10	125-150	0.2-3	0.01-0.1
Multimode	50-60	125-150	0.3-3	1-20
With Large Core	200-600	to 1,000	5-20	30-100

Because of their significant losses (100-1,000 decibels per kilometer), plastic fibers and bundles of fibers are not given. They usually find application in short fiber optic communications lines (up to 50-100 meters) and are gradually being supplanted by glass fibers with large cores. From the standpoint of distance, speed, and reliability of data transmission, losses per unit of length, dispersion, and diameter of the fiber are the determining factors in selecting and designing the cable (if we do not consider cost). For sources of emission and photoreceivers the 0.8-0.9 micron range can be considered industrially adopted at the present time. Nonetheless, use of the 1.3-1.6 micron range, which has been intensively studied in recent years, offers significant advantages for transmitting data at high speed and over large distances [5]. In this range losses in the fiber may be 0.2-0.3 decibels per kilometer, which is significantly lower than losses in the 0.8-0.9 micron field. Material and waveguide dispersion in the 1.3-1.7 micron field have opposite signs. The parameters of the light guide are thus optimized to have a low total dispersion for the required length of the wave. A transmission distance of 62.3 kilometers without relays at a speed of 32 mbits per second [6] and 15 kilometers at a speed of 1.6 Gbits per second [7] have already been obtained on the 1.3 micron wavelength. Even greater advantages are expected with use of the 1.5 micron wave, where attenuation in the fiber is close to the minimum of 0.2 decibels per kilometer [8].

The principal sources of emissions for optical fiber communications lines are semiconductor light diodes that emit from the surface and the end, superluminescent light diodes, and injection lasers. In the 0.8-0.95 micron range they are fabricated on the basis of GaAlAs. The service life of series-produced light diodes and superluminescent light diodes has been raised to $5 \cdot 10^4$ and more hours, and laboratory models have been increased to 10^6 hours. The main advantage of light diodes with large emitting surfaces is better linearity of the characteristic of emissive power from the pumping current. But they are inferior to other types of emitters with respect to the important indicator of the pass band. At high transmission speeds in systems with code-pulse modulation of signals, superluminescent light diodes and semiconductor lasers fabricated by a similar technology on the basis of double heterostructures are used. In terms of emissive power, connection to the fiber, efficiency, cost, reliability, and speed, superluminescent light diodes have fairly good properties for most applications. As an example the following indicators can be given: emissive power from the end of the superluminescent light diode in the 0.8-0.9 micron range with a pumping current of 100-300 mA is a few mWt, while the modulation frequencies can be hundreds of megahertz. Overall, with respect to level of introduced power and speed superluminescent light diodes are inferior only to injection lasers whose modulation frequency can be several gigahertz and have an output power of more than 10 mWt at the same pumping currents. Their main shortcomings are a smaller service life (10^3-10^4 hours) and a significant dependence of characteristics on temperature. But these shortcomings refer to the present day and we can expect improvement in their properties.

The narrow spectral width of the emission of ejection lasers makes it possible to receive the least widening of the transmitted pulses owing to dispersion in the fiber. As a result, semiconductor laser diodes are used for high-speed data transmission over great distances. At the same time, in the 1.3-1.6 micron field with very low dispersion in the light guide, it is also possible to use light diode sources for significant transmission distances.

There are reports that a light diode has been made based on InTaAsP/InP with a microlens that supports an output on the order of 0.2 mWt fed to the fiber at a current of 100 mA and a wavelength of 1.3 microns. It is expected to be possible with such an emitter to transmit data at a speed of 30 mbits per second over a distance of 30 kilometers [9].

Miniature neodymium lasers with pumping from a light diode can be considered a promising source of emissions for the 1.06 and 1.35 wavelengths [10] but such lasers require an external modulator. Gas lasers can also be used as emitters because they provide a fairly simple connection with integrated optical modulaters. In this case the frequency of modulation of the continuous emission of a laser is several gigahertz.

In general the transition to high-speed data transmission systems is linked to the development of elements of integrated optics.

Solid state photoreceivers used in optical fiber communications sytems should have high sensitivity on the wavelength of the emission source, high speed, and a low noise level. Silica pin photo diodes with a sensitivity up to 0.6 amperes per watt of optical emission power striking the photo diode have been used most widely in the range up to 1.1 microns. A significant increase in sensitivity, by roughly an order, is achieved in avalanche photo diodes that combine detection of optical signals with internal amplification of the photocurrent. Internal amplification occurs by avalanche multiplication of media in the domain of a strong electrical field. At the same time, avalauche photo diodes require comparatively high supply voltages, stabilization of voltage, and temperature stabilization of the amplification factor.

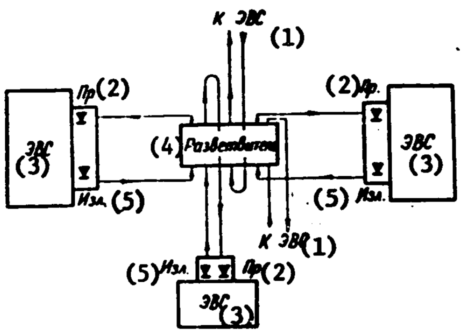
At the present time, photo diodes of both types have been developed with speeds of less than one nanosecond [1-3]. Thanks to their small dimensions they match well with fiber light guides and electronic units. In the 1.0 micron and higher range where silica instruments do not operate, avalanche and pin photo diodes made of germanium are used, and in recent times semiconductor combinations of groups III and V have been used. They have quite high speed (rise time on the order of several dozens of picoseconds) and better noise characteristics than germanium.

The connecters which interlink the emitters and receivers with the fiber and the connection of individual segments of fiber are important elements of a fiber optic communications line. For large-diameter fiber (50-600 microns) and for fiber bundles such connecters are manufactured in series abroad. They are plastic or metal and insure interlinking with losses of 1-3 decibels [11, 12]. Multichannel connecters have also been built. Connecting fibers with a cord diameter of 5-15 microns is much more complex and requires precision fabrication of the connecter elements. Nonetheless, there are reports of connecters with insertion losses that do not exceed 0.5 decibels [13].

The components of optical fiber communications lines we have considered are the primary ones because they must be used in relay systems that employ intermediate conversion of optical signals into electrical signals and back as well as in systems where only optical signals are used for data transmission. In relay systems the various units may be connected to the communications grid electrically or optically through the relays themselves with intermediate processing and restoration of the signal. For data input and output at any point of purely optical systems there must be elements that distribute the optical energy over different channels (fibers) or, conversely, consolidate several channels, including those with different wavelengths, into one. These elements are usually called "splitters" [razvetviteli], directed branchers [otvetviteli], and so on. Among the various splitter designs are the "star" type which distribute the input signal equally to all outputs, and the T-shaped brancher. Depending on the design the star-type splitter can work in a pass mode, in which case it has M inputs and M outputs, or a reflection mode, in which case it has M equal inputs, each of which can serve as an input and an output. The former (pass mode) is essential when associating units interconnected by two light guides, one for transmission and the other for reception (see Figure 2). The reflective splitter makes it possible to connect each

Figure 2. The EVS — Computer System Element (Processor, Storage, Special Processor, Intelligent Terminal, and Peripherals)

Key: (1) To EVS;
(2) Transmitter;
(3) EVS;
(4) Splitter;
(5) Emitter.



unit with a single light guide that both receives and transmits data (see Figure 3 below). In this case data input and output to the light guide from the

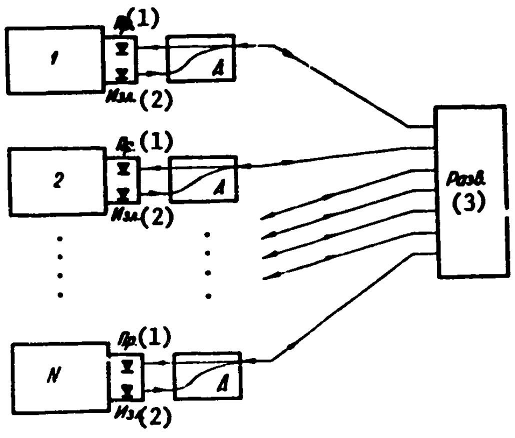


Figure 3. 1-N — Computer System Elements, Д — Directed Splitter-Duplexer, and Star-Type Splitter with N Inputs.

Key: (1) Transmitter;
(2) Emitter;
(3) Splitter.

unit being connected must be accomplished through the duplexer (D in Figure 3) or by means of a semiconductor structure that combines the functions of emitter and receiver for different biases [14].

The directed T-type brancher makes it possible to take a certain part of the power off the main line and to feed signals from different sources to the line. The brancher, which is formed by fusing a single-mode fiber with a multimode fiber, feeds emissions from the single-mode fiber to the pipeline with losses at a level of 0.5 decibels [15]. With this brancher-duplexer a duplex data transmission line can be constructed on one right wire [16].

Directed T-type branchers that both input and output emissions from the pipeline are obtained by fusing fibers of similar corss-sections and by various other means. These branchers make it possible to connect peripheral equipment to the optical line running from the central processor (see Figure 4). The proportion

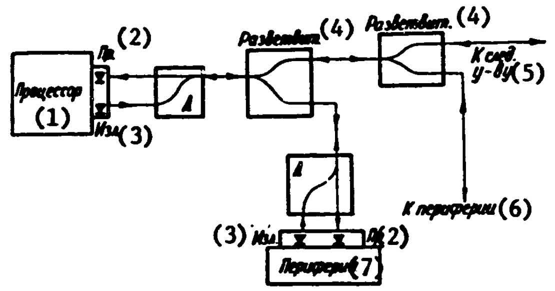


Figure 4. "Star" Type Connection

Key:	(1)	Processor;
	(2)	Transmitter;
	(3)	Emitter;
	(4)	Splitter;
	(5)	To Next Unit;
	(6)	To Peripherals;
	(7)	Peripherals.

of power that is taken off can be varied from one brancher to another in order to even out the signals coming to different elements of the system. In those cases where peripheral units must exchange data among themselves, and also when elements of distributed computing complexes are interconnected, a symmetrical directed brancher with four inputs can be used [17]. The symmetrical brancher and the star-type M × M brancher make it possible to associate units both by type of two-way common line (see Figure 5 below) and in a circle (see Figure 6 below). In the latter case it may be necessary to take measures to prevent information from circulating around the circle.

Whe selecting the final variation of communications -- with a two-way line, circle, or star-type splitter, the total losses introduced into the channels, including also losses in the connections of the light line with the splitters, should be evaluated. For many associated units the radial system with a star

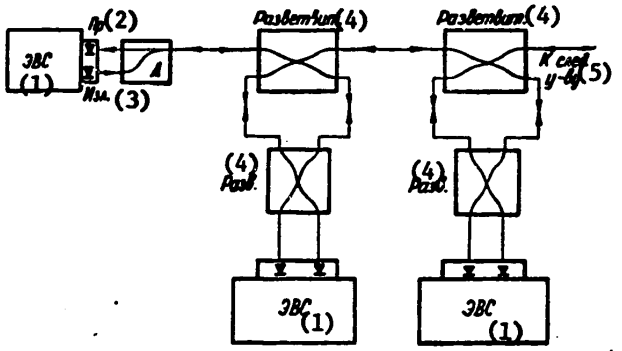


Figure 5. "Common Line" Type Connection.

Key: (1) EVS;
(2) Receiver;
(3) Emitter;
(4) Splitter;
(5) To Next Unit

form has smaller losses, but it requires a greater length of fiber cable to connect all the units. In the actual system of a distributed computer complex it may prove advisable to use a combination of different types of splitters, and where significant distances are involved intermediate relays may also be desired.

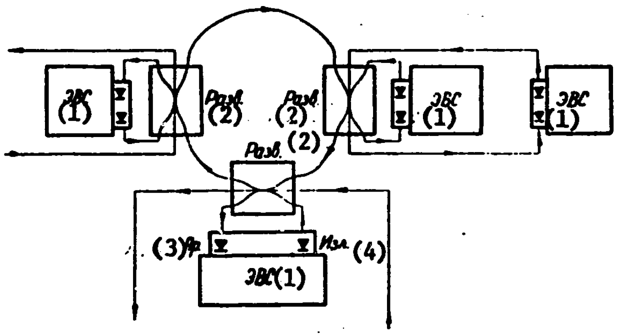


Figure 6. The EVS — Computer System Element (Processor, Storage, Special Processor, Intelligent Terminal, and Peripherals)

Key: (1) EVS;
(2) Splitter;
(3) Transmitter;
(4) Emitter.

In the best models of splitters built today for multimode fiber with a core diameter of 60-600 microns, the induced (excess) optical losses for connecting 3-20 channels do not exceed 0.5-5 decibels [11]. Prospects for building splitters for single-mode fiber systems are based chiefly on the advances of integral optics. At the present time, integral optic waveguide structures are being effectively developed. They perform various functions, including the functions of splitters, channel switchers, modulators, and multiplexer. Laboratories have built and are studying integral optical modulators; matrixes of electrooptical switch-splitters with up to 4 × 4 channel; multiplexer optical integral circuits which combine, in a common waveguide output, the emission of several laser diodes with different wave lengths and independent modulation; emission input-output units for integral-optical structures and for connecting them to a single-mode fiber [3, 18]. It is the integral optical instruments that make it possible to achieve maximum data transmission speed for fiber optic communications lines with single-mode fiber and to switch channels rapidly in fiber communications networks to transmit data along the required routes.

Evaluating the state of the current and prospective basic elements of fiber optic communications lines, we may identify the following three types:

1. fiber optic communications lines with data transmission speeds to the level of megabits per second. Depending on their length, these lines can use fiber bundles (up to 100-200 meters), cables with large-diameter fiber (for distances up to a few kilometers), and also series-produced light and photo diodes and phototransistors;

2. fiber optic communications lines for data transmission speeds up to dozens of megabits per second, based on multimode fibers of different diameters, superluminescent light diodes, pins, or avalanche photodiodes. These lines can be up to several kilometers long;

3. fiber optic communications lines with speeds from 100 megabits per second to tens of gigabits per second and up to dozens of kilometers long. They are based on single-mode fiber, injection lasers, and avalanche photodiodes, and in many cases use integral optical modulators.

The use of laser diodes in lines of the first and second type can increase their length to 10 and more kilometers. But this causes a substantial rise in cost and complexity of design because of the need for automatic monitoring of the level of laser emissions. The classification given above is arbitrary because during selection of the system component one must take account of the type of compaction, modulation, coding, and synchronization, and then look at the sources and receivers of emission, modulators, and branchers, type of fiber, and matching units. Work [19] proposes that this problem be solved beginning from data transmission speed, the band occupied by the channel, the number of channels, the reliability of data transmitted, the reliability and complexity of the hardware, cost, and power consumption.

Other factors in addition to these that must be considered in designing systems are flexibility of the system, the possibility of increasing carrying capacity, and connecting in hardware that is in the design stage, in other words further development of the systems.

The most obvious use for fiber optic communications lines is as communications channels in systems for collection of data and control of experiments or industrial processes. In this case they are used to solve problems related to the remoteness of different units from one another, large differences in potentials among these units, and high noise levels [20-25]. The main advantages of using fiber optic communications lines in these systems are: galvanic isolation of their elements and high noise suppression in communications lines running great distances. An example of a successful application is development of the sequential CAMAC branch on fiber optic communications lines [20]. This makes it possible to use a single-strand optical cable instead of a 10-strand electrical cable to maintain data transmission speeds of up to five megabytes per second and complete electrical isolation of all crates included in the branch. It is entirely feasible to realize this system at the present time using elements produced by industry.

Another example of effective use of a fiber optics communications line is transferring the parallel input-output interface of YeS computers, which has 34 physical lines, to an optical fiber sequential interface realized with two light guides. The application of fiber optic communications lines in this case gives high transmission speed (60 megabits per second) and noise suppression and allows a significant increase in distances between computers and peripheral units and simplification of communications in the system.

There is considerable interest in using fiber optic communications lines in the main channels of computer networks and to organize local networks on the sequential common line principle. In these cases the speed of data transmission may reach several megabits per second.

In all these cases, naturally, significant difficulties arise during development of the necessary "framing" (high-speed code convertors, various types of switching and logical elements, and the like).

In conclusion, the authors would like to stress that advances in the fields of physics and technology have led to the development of optical fibers and integral optical elements that not only have a significant effect on the character and quality of communications lines, but can also result in a substantial change in the architecture of computer systems.

BIBLIOGRAPHY

1. S. Miller, and A. Chynoweth, "Optical Fiber Telecommunications," New York, "Acad. Press," 1979, 720 pages.

2. H. Kressel, "Semiconductor Devices for Optical Communications," Berlin, "Springer-Verlag," 1980, 280 pages.

3. T. G. Dzhallorenzi, "Studies and Technology of Optical Communications System. Fiber Optics," TIIER, 1978, Vol 66, No 7, pp 29-72.

4. G. Khindin, "What Developers Should Know About Ready-To-Use Fiber Optic Lines," ELEKTRONIKA, 1978, Vol 51, No 26, pp 35-42.

5. Ye. M. Dianov, "Prospects for the Use of the 1-1.6 Micron Wave Band for Fiber Optical Communications," KVANTOVAYA ELEKTROTEKHNIKA, 1980, Vol 7, No 3, pp 453-464.

6. T. Ito, K. Nakagawa, K. Ishihara, Y. Ohmori, and K. Sugiyama, "Transmission Experiments in the 1.2-1.6 m Wavelength Region Using Graded-Index Optical Fiber Cables," in "Tech. Dig. Topical Meet. Optical Fiber Communication, OSA,(Washington, D.C.). Paper TuB1," March 1979, pp 6-8.

7. T. Kimura, and H. Kanbe, "Long Wavelength Single-Mode Fiber Transmission System," in "Topical Meeting on Integrated and Guided Wave Optics," Nevada, Incline Village, January 28-30, 1980.

8. T. Miya, Y. Terunuma, T. Hosaka, and T. Miyashita, "Ultimate Low-Loss Single-Mode Fiber at 1.55 m," ELECTRON LETT., February 1979, Vol 15, pp 106-108.

9. R. C. Goodfellow, A. C. Carter, I. Griffith, and R. R. Bradley, "GaInAsP/InP Fast, High-Radiance, 1.05-1.3 m Wave Length LED's with Efficient Lens Coupling to Small Numerical Aperture Silica Optical Fiber," IEEE TRANS. ELECTRON. DEVICES, August 1979, Vol Ed.-26, pp 12-15-1220.

10. J. P. Boudin, M. Neubauer, and M. Rondot, "On the Design of Neodymium Miniature Lasers," IEEE J. QUANTUM ELECTRON., November 1978, Vol QE-14, pp 831-839.

11. P. Ormond, "Fiber Optic Components," EDN MAGAZINE, 1979, Vol 24, No 6, pp 37-96.

12. "The PRW Connector," ELEKTRONIKA, 1979, Vol 52, No 14, pp 8-9.

13. N. Shimuzu, and H. Tsuchiya, "Single-Mode Fiber Connectors," ELECTRON. LETT., September 1978, Vol 14, 19, pp 611-613.

14. "Diode ER955, Thomson-CSF," preliminary data sheet.

15. B. Kowasaki, and K. Hill, "Efficient Power Convolver for Multiplexing Multiple Sources to Single Fiber Optical Systems," APPL. PHYS. LETT., 1977, Vol 31, 740 pages.

16. B. S. Kowasaki, K. O. Hill, D. C. Johnson, and A. A. Tenne-Sens, "Full Duplex Transmission Link over Single-Strand Optical Fibers," OPT. LETT., September 1977, Vol 1, No 3, pp 107-108.

17. "Specially-Made Directed Brancher," ELEKTRONIKA, 1978, Vol 51, No 22, pp 4-6.

18. E. Konvell, "Integral Optics," UFFN, 1978, Vol 126, No 4, pp 639-656.

19. I. V. Kalmykov, A. M. Prokhorov, N. D. Simachev, and N. D. Sisakyan, "FIAN Preprint,' No 158, SER. KVANTOVAYA ELEKTRONIKA, Moscow, 1979, 24 pages.

20. Ye. D. Bulatov, A. A. Danilenko, I. V. Kalmykov, V. G. Lomanov, Ye. A. Otlivanchik, and I. N. Sisakyan, "Automation of Physics Experiment Based on a Fiber Light Diode CAMAC Sequential Branch," TEZ. DOKL. NA KONF. PO AVTOMATIZATSII EKSPERIMENTAL'NYKH ISSLEDOVANIY, Kuybyshev, 1978, p 86.

21. M. I. Belovolov, M. M. Bubnov, A. N. Gur'yanov, G. G. Devyatykh, Ye. M. Dianov, V. I. Pelipenko, A. M. Prokhorov, and I. N. Sisakyan, "Study of Fiber Optic Systems for Communication among Blocks of the Computer," KVANTOVAYA ELEKTRONIKA, 1977, Vol 4, No 11, pp 2,456-2,459.

22. Ye. D. Bulatov, Ye. B. Zhilin, G. I. Zatsepin, I. V. Kalmykov, V. S. Korzinkin, V. G. Lomanov, Ye. A. Otlivanchik, A. M. Prokhorov, and I. N. Sasakyan, "Optical Communications Channel for Computer Systems," TEX. DOKL. NA KONF. PO AVTOMATIZATSII... op. cit., p 85.

23. Ye. D. Bulatov, I. V. Kalmykov, V. G. Lomanov, Ye. A. Otlivanchik, A. M. Prokhorov, and I. N. Sisakyan, "Universal Fiber Optic Communications Line Channel between Computers Based on the CAMAC System," PREPRINT FIAN, No 160, Moscow, 1977, 6 pages.

24. L. Tomasetta, "GaAs Optical Electronic Devices for Signal Processing Applications," PROC. OF THE SOC. OF PHOTO-OPTICAL INSTR. ENG., 1979, Vol 176, pp 111-114.

11,176
CSO: 1863/120

UDC 681.324-192

MATHEMATICAL PROCEDURES FOR EVALUATING COMPUTER SYSTEM RELIABILITY

Riga AVTOMATIKA I VYCHISLITEL'NAYA TEKHNIKA in Russian No 1, Jan-Feb 82 (manuscript received 4 Jan 80, after revision 6 Jun 80) pp 67-71

[Article L. I. Kul'bak, D. I. Karaban' and S. S. Prokhorenko: "Indicators for Evaluating the Reliability of Multiprocessor Computing Systems"]

[Text] 1. Requirements of indicators for evaluating the reliability of multiprocessor computing systems (MCS's). The basic features of MCS's are modular construction and accessibility of any processor module to any memory module, as a result of which the system has greater vitality. This is expressed in the ability of the MCS, when particular components fail (with the exception of the functionally essential equipment, the "nucleus"), to continue working at lowered productivity, not losing the ability to work entirely. Another important characteristic of MCS's is that the productivity of the system depends on the number of processor modules, storage modules, and channels for communication with peripheral units.

The indicator for evaluating the reliability of MCS's should take into account their enumerated features and must coincide in the particular case (when a lowering of productivity is not acceptable) with the indicator for evaluating the reliability of a standard computer system so that it will be possible to make a comparative evaluation of the reliability of different MCS's and standard computer systems.

2. Mathematical models of the quality of functioning of the MCS. Suppose an MCS consists of n elements. The state of each element i (i=1, 2, ..., n) is described by the function

$$X_i(t) = \begin{matrix} 1, \text{ if element } i \text{ at moment } t \text{ is operable;} \\ 0, \text{ if element } i \text{ at moment } t \text{ is not operable.} \end{matrix}$$

The state of the MCS in the general case can be described by the vector:

$$\bar{Z}(t) = \left\| \begin{matrix} X_1(t) \\ \vdots \\ X_n(t) \end{matrix} \right\|.$$

We will designate the productivity of the MCS at moment t as $\pi_z(t)$; then $\pi_z(t) = [\bar{Z}(t)]$.

Because state $\bar{Z}(t)$ of the system changes randomly in time, process $[\bar{Z}(t]$ of the change in productivity is random. It can be considered as the aggregate of random functions $\{\pi_z(t)\}$, which show the change in productivity of the MCS for all possible changes in its state $Z(t)$ during the entire period of use, or as the aggregate of random quantities dependent on parameter t.

The random process $\pi[\bar{Z}(t)]$ is a general mathematical model of the quality of functioning of the MCS.

3. Indicators of the quality of functioning of the MCS. It is advisable to adopt the simplest function that characterizes the random process of change in MCS productivity as the indicator of quality of functioning of the MCS at moment t. This function is the mathematical expectation of random function $\pi_z(t)$ as the average for a set of observations of random process $[\bar{Z}(t)]$ at moment t, $\pi(t) = M\{\pi[\bar{Z}(t)]\}$.

The average value of the quality of functioning of the MCS is interesting. It can be expressed in the form $\pi_{cp}=\sum_{j=1}^{N}\pi_j P_j$, where π_j is the productivity of the is the productivity of the MCS in state j; P_j is the probability of state j of the MCS; and, N is the number of MCS states differing productivity.

The state of the ideal (in the sense of failure-free) MCS is described by the vector

$$\bar{Z}^0(t)=\begin{Vmatrix} 1 \\ \vdots \\ 1 \end{Vmatrix}.$$

Therefore, for the ideal MCS $\pi_{z^0}(t)=M\{\pi[\bar{Z}^0(t)]\}=\pi_{max}$, where π_{max} is the maximum productivity of the MCS.

It is convenient to make a comparative evaluation of the states of one MCS using a reactive indicator of MCS productivity in the form $\alpha(t)=\pi(t)[\pi_{max}]^{-1}$, $\alpha_{cp}=\pi_{cp}(\pi_{max})^{-1}$.

4. Indicators for evaluating the reliability of the MCS. The following indicators best meet the requirements formulated in section 1 above for evaluating MCS reliability: working time until productivity drops below level $\alpha_{\tau p}$ (analogous to working time until failure), designated $T_{\alpha_{\tau p}}$; coefficient of readiness for work at a productivity level not lower than $\alpha_{\tau p}$ (analogous to readiness coefficient), designated as $K_{\alpha_{\tau p}}$; probability of maintaining productivity not lower than level $\alpha_{\tau p}$ during period of time t (analogous to probability of trouble-free operation), designated as $P_{\alpha_{\tau p}}(t)$; average time of restoration to a level not lower than $\alpha_{\tau p}$ (analogous to average restoration time), designated as $T_{B\alpha_{\tau p}}$; and, average level of productivity (no analog) α_{cp}.

5. Mathematical model of the process of change in MCS states. The standard MCS consists of four types of elements: nucleus, processor modules, internal memory modules, and channels for communication with peripheral units.

Suppose that the states of the MCS differ from one another by level of productivity α. Then the number of MCS states is determined by the number of combinations of failure-free MCS elements that support the different levels of productivity.

Let us make the following assumptions: the failures of MCS elements are independent; the flows of failures and restorations of MCS elements are elementary; the monitoring of the state of MCS elements is continuous and reliable; restoration of the MCS is restricted, and the first element to be restored is the one that supports restoration of the greatest MCS productivity; and, the initial state of the MCS is a state that assures productivity $\alpha = 100$ percent without redundancy.

On these assumptions the process of change in the states of the MCS will be a discrete markovian process with continuous time described by a system of linear differential equations.

6. Calculated ratios for evaluating the reliability of the MCS. We will adopt the following designations: $H_{\alpha_{T\rho}}(a, b)$ is the mathematical expectation of number of times MCS productivity drops below level $\alpha_{T\rho}$ in time interval $a \leqslant t \leqslant b$; $\omega_{\alpha_{T\rho}}(t)$ is the parameter of the flow of drops in productivity below level $\alpha_{T\rho}$.

It is apparent that $H_{\alpha_{T\rho}}(a, b) = H_{\alpha_{T\rho}}(t_0, b) - H_{\alpha_{T\rho}}(t_0, a)$, where t_0 is an arbitrary start of the count.

By analogy with working time until failure, working time until productivity drops below level $\alpha_{T\rho}$ is defined as follows [1]:

$$T_{\alpha_{T\rho}}(a, b) = \frac{b-a}{H_{\alpha_{T\rho}}(t_0, b) - H_{\alpha_{T\rho}}(t_0, a)}.$$

Using property $H_{\alpha_{T\rho}}(t_0, b) = \int_{t_0}^{b} \omega_{\alpha_{T\rho}}(t)\,dt$, we receive

$$T_{\alpha_{T\rho}}(a, b) = (b-a)\left[\int_{t_0}^{b} \omega_{\alpha_{T\rho}}(t)\,dt - \int_{t_0}^{a} \omega_{\alpha_{T\rho}}(t)\,dt\right]^{-1}.$$

With the assumptions in section 5 above, $\omega_{\alpha_{T\rho}}(t) = \omega_{\alpha_{T\rho}} = \text{const}$, then $T_{\alpha_{T\rho}}(a, b) = T_{\alpha_{T\rho}} = \omega^{-1}_{\alpha_{T\rho}}$.

From the set of states of the MCS β we single out the subset of states corresponding to a level of productivity not lower than $\alpha_{T\rho}$, and designate it $\beta_{\alpha_{T\rho}}$. In the subset $\beta_{\alpha_{T\rho}}$ we single out the subset of MCS states from which it is possible to make a direct transition to MCS states with lower levels of productivity, and we designate it $\beta^*_{\alpha_{T\rho}}$.

Considering the designations we have adopted, $\omega_{\alpha_{T\rho}} = \sum_{i \in \beta^*_{\alpha_{T\rho}}} \omega_{\alpha_{T\rho} i}\, q_{\alpha_{T\rho} i}$, where $\omega_{\alpha_{T\rho} i}$ is the intensity of output from state i of subset $\beta^*_{\alpha_{T\rho}}$ to MCS states with productivity $\alpha < \alpha_{T\rho}$; $q_{\alpha_{T\rho} i}$ is the hypothetical probability of finding the MCS in state i of subset $\beta^*_{\alpha_{T\rho}}$ on the condition of finding the MCS in the states of subset $\beta_{\alpha_{T\rho}}$.

It follows from the definition of $q_{\alpha_{\tau p}i}$ that $q_{\alpha_{\tau p}i} = h_{\alpha_{\tau p}i}\left(\sum_{j\in\beta_{\alpha_{\tau p}}} h_{\alpha_{\tau p}j}\right)^{-1}$, $h_{\alpha_{\tau p}i} = P_{\alpha_{\tau p}i}(P_{\alpha=100\%})^{-1}$, where $P_{\alpha_{\tau p}i}$ is the stationary value of the probability of finding the MCS in state i of subset $\beta_{\alpha_{\tau p}}$, $i\in\beta_{\alpha_{\tau p}}$.

After a number of substitutions we obtain

$$T_{\alpha_{\tau p}} = \left[\sum_{i\in\beta^{*}_{\alpha_{\tau p}}} \omega_{\alpha_{\tau p}i} h_{\alpha_{\tau p}i}\left(\sum_{j\in\beta_{\alpha_{\tau p}}} h_{\alpha_{\tau p}j}\right)^{-1}\right]^{-1}. \qquad (1)$$

The coefficient of MCS readiness to work with the level of productivity not lower than $\alpha_{\tau p}$ can be defined as the probability of finding the MCS at an arbitrary moment in time (without considering planned shutdowns) in a state with productivity not lower than $\alpha_{\tau p}$. Following the definition and assumptions of section 5 above, $K_{\alpha_{\tau p}} = P_{\alpha=100\%}\sum_{i\in\beta_{\alpha_{\tau p}}} h_{\alpha_{\tau p}i}$.

It follows from the condition of normalization $\sum_{i\in\beta} P_{\alpha_{\tau p}i} = 1$ that $P_{\alpha=100\%} = \left(\sum_{i\in\beta} h_{\alpha_{\tau p}i}\right)^{-1}$.

Considering that latter,

$$K_{\alpha_{\tau p}} = \sum_{i\in\beta_{\alpha_{\tau p}}} h_{\alpha_{\tau p}i}\left(\sum_{j\in\beta} h_{\alpha_{\tau p}j}\right)^{-1}. \qquad (2)$$

With the assumptions in section 5 above $K_{\alpha_{\tau p}} = T_{\alpha_{\tau p}}(T_{\alpha_{\tau p}} + T_{s\alpha_{\tau p}})^{-1}$, from which $T_{s\alpha_{\tau p}} = T_{\alpha_{\tau p}} \times (1 - K_{\alpha_{\tau p}})K^{-1}_{\alpha_{\tau p}}$.

If $P^{*}_{\alpha_{\tau p}i}(t)$ is the probability of finding the MCS in state i of subset $\beta_{\alpha_{\tau p}}$ at moment i on the condition of zero intensities of return to subset $\beta_{\alpha_{\tau p}}$ after leaving it, the following is obvious: $P_{\alpha_{\tau p}}(t) = \sum_{i\in\beta_{\alpha_{\tau p}}} P^{*}_{\alpha_{\tau p}i}(t)$.

In engineering practice the more convenient formula [2] is $P_{\alpha_{\tau p}}(t) = \exp[-i(T^{1}_{\alpha_{\tau p}})^{-1}]$, where $T^{1}_{\alpha_{\tau p}}$ is the average working time until the first drop in MCS productivity below level $\alpha_{\tau p}$.

From the well-known expression in reliability theory it follows:

$$T^{1}_{\alpha_{\tau p}} = \sum_{i\in\beta_{\alpha_{\tau p}}} \int_0^{\infty} P_{\alpha_{\tau p}i}(t)\,dt$$

We will designate

$$C_{\alpha_{\tau p}i} = \int_0^{\infty} P^{*}_{\alpha_{\tau p}i}(t)\,dt = \lim_{s\to 00}\int_0^{\infty} \exp(-st)\times P^{*}_{\alpha_{\tau p}i}(t)\,dt,$$

then

$$P_{\alpha_{\tau p}}(t) = \exp\left[-t\left(\sum_{i\in\beta_{\alpha_{\tau p}}} C_{\alpha_{\tau p}i}\right)^{-1}\right]. \qquad (3)$$

The average level of productivity is

$$\alpha_{cp}=\sum_{i\in\beta}\alpha_i P_i=\sum_{i\in\beta}\alpha_i h_i\Big(\sum_{j\in\beta}h_j\Big)^{-1}. \qquad (4)$$

7. The algorithm for calculating indicators for evaluating MCS reliability. It is recommended that indicators for evaluating MCS reliability calculated by the following algorithm.

7.1. One of the possible methods (analytic, by modeling, and the like) determines the states of the MCS by combinations of element failures that maintain productivity $\alpha \geq \alpha_{TP}$ ($\beta_{\alpha_{TP}}$), and those states to which a direct transfer can be made from subset $\beta_{\alpha_{TP}}$, that is, the set of states β. The states of the subset $\beta^*_{\alpha_{TP}}$ are singled out from the subset $\beta_{\alpha_{TP}}$.

7.2. A marked graph of the MCS states is compiled, with the number of the state and productivity of the MCS in this state (α_i) put at the nodes and the intensities of transitions from one state to another marked on the arcs.

7.3. Using the generally accepted technique [2], a system of differential equations is compiled from the graph.

7.4. To determine H_{α_i} $(i \in \beta_\alpha)$, $P_{\alpha_i}(t)=P_{\alpha_i}\cdot\frac{dP_{\alpha_i}(t)}{dt}=0$ is adopted in the initial system of equations; then all the equations are divided termwise by $P_{\alpha=100\%}$. The result is a system of N algebraic equations relative to h_{α_i}, whose roots are used in formulas (1), (2), and (4) to determine the indicators $T_{\alpha_{cp}}$, $K_{\alpha_{cp}}$, and α_{cp}.

7.5. The indicator $T_{\alpha_{TP}}$ is computed according to formula (1), while $K_{\alpha_{TP}}$ is computed by formula (2) and α_{cp} by formula (4).

7.6. To determine $C_{\alpha_{TP}i}$ in the initial system of differential equations, we take the intensities of the return to the states of subset $\beta^*_{\alpha_{TP}}$ to be equal to zero and eliminate equations for states where $\alpha<\alpha_{cp}$.

In the resulting system we go from the originals to representations of a Laplace transform for an initial state of $P_{\alpha=100\%}(0) = 1$, $P_{\alpha_i}(0) = 0$ $(\alpha_i \neq 100\%,\ i\in\beta)$. We turn s toward zero. We obtain a system of algebraic equations relative to $C_{\alpha_{TP}i}$. From this resulting system of equations $C_{\alpha_{TP}i}$ $(i \in \beta_{\alpha_{TP}})$ are determined and used in formula (3) to determine the indicator $P_{\alpha_{TP}}(t)$.

Example of calculating indicators to evaluate MCS reliability. The possibility of evaluating the reliability of an MCS which is a functionally distributed system of microprocessors is of some interest. The heightened attention given to these systems can be explained by the swift development of technology and the growing complexity of software. Some of the microprocessor modules of a distributed system are functionally oriented to process control, which was formerly done by software. This makes it possible to raise the overall productivity of the MCS. Suppose that in the MCS the functionally identified microprocessor modules are included in the nucleus of the system, which maintains a parameter for the stream of failures of $\lambda_1 = 26.17 \cdot 10^{-5}$ hr^{-1}, and suppose that the system includes seven microprocessor modules in the execution field with a parameter for the stream of failures of each module of $\lambda_2 = 1.81 \cdot 10^{-5}$ hr^{-1}, nine internal memory modules with a parameter for the stream of failures of each of $\lambda_3 = 2.51 \cdot 10^{-5}$ hr^{-1}, and two channel modules with a parameter for the stream of failures of each of $\lambda_4 = 2.90 \cdot 10^{-5}$ hr^{-1}. The intensities of restoration of all elements are $\nu = 2$ hr^{-1}. The table below gives the states of the MCS and the productivities corresponding to them for calculating the reliability indicators of the MCS where $\alpha_{тр} = 93$ percent.

Table. States of the MCS ($\alpha_{тр} = 93\%$)

Номер состояния (a)	(b) Число работоспособных			Состояние ядра МВС (f)	Относительная производительность (g) α, %
	модулей процессоров (c)	модулей оперативной памяти (d)	каналов (e)		
0	7	9	2	1	100
1	7	9	2	0	0,00
2	6	9	2	1	90,57
3	7	8	2	1	96,98
4	7	9	1	1	80,00
5	7	8	2	0	0,00
6	6	8	2	1	88,27
7	7	7	2	1	93,32
8	7	8	1	1	77,50
9	7	7	2	0	0,00
10	6	7	2	1	82,03
11	7	6	2	1	88,86
12	7	7	1	1	74,60

Key: (a) Number of State; (b) Number of Operable Elements; (c) Processor Modules; (d) Internal Memory Modules; (e) Channels; (f) State of Nucleus of MCS; (g) Relative Productivity α, %.

We compile the initial system of differential equations according to the graph shown in the figure below.

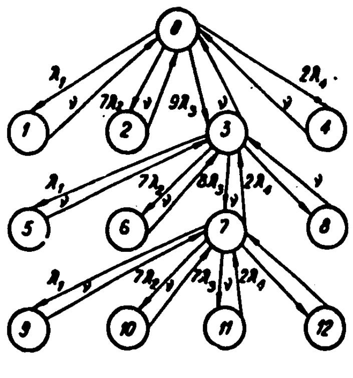


Figure.

45

Following section 6 above, the formulas for calculating the reliability indicators of the MCS under consideration are as follows: $T = 93\% = (1 + h_3 + h_7)\times[\lambda_1 + 7\lambda_2 + 2\lambda_4)(1 + h_3 + h_7) + 7\lambda_3 h_7]^{-1}$, $K_{\alpha=93\%} = (1 + h_3 + h_7)[1 + \sum_{i=1}^{12} h_i]$, $T_{B\alpha=93\%} = v^{-1}$, $P_{\alpha=93\%}(t) = \exp]-t(C_0 + C_3 + C_7)^{-1}]$.

Following section 7 above we obtain the formulas for calculating h_i and C_i: $h_1 = \lambda_1 v^{-1}$, $h_2 = 7\lambda_2 v^{-1}$, $h_3 = 9\lambda_3 v^{-1}$, $h_4 = 2\lambda_4 v^{-1}$, $h_5 = 9\lambda_1\lambda_3 v^{-2}$, $h_6 = 63\lambda_2\lambda_3 v^{-2}$, $h_7 = 72\lambda_3^2 v^{-2}$; $h_8 = 18\lambda_3\lambda_4 v^{-2}$, $h_9 = 72\lambda_1\lambda_3^2 v^{-3}$, $h_{10} = 504\lambda_2\lambda_3 v^{-3}$, $h_{11} = 504\lambda_3^3 v^{-3}$, $h_{12} = 144\lambda_3^2\lambda_4 v^{-3}$, $C_3 = 9\lambda_3]\Lambda(\lambda_1 + 7\lambda_2 + 8\lambda_3 + v] - 9\lambda_3 v]^{-1}$, $C_0 = (1 + vC_3)\Lambda^{-1}$, $C_7 = 8\lambda_3 C_3(\Lambda - 2\lambda_3 + v)^{-1}$, where $\Lambda = \lambda_1 + 7\lambda_2 + 9\lambda_3 + 2\lambda_4$.

As the result of the calculation the following indicators are obtained for the reliability of the MCS: $T_{\alpha=100\%} = 1{,}487$ hours; $K_{\alpha=100\%} = 0.99966$; $T_{B\alpha=100\%} = 0.5$ hours; $T_{\alpha=93\%} = 2{,}018$ hours; $K_{\alpha=93\%} = 0.99975$; $T_{B\alpha=93\%} = 0.5$ hours; $P_{\alpha=100\%}(t) - \exp(-t/1{,}487)$; $P_{\alpha=93\%}(t) = \exp(-1/2{,}240)$; $\alpha_{cp} = 99.97\ \%$.

BIBLIOGRAPHY

1. "Nadezhnost' v tekhnike. Terminy i Opredeleniya. GOST 13377-75," [Reliability in Engineering. Terms and Definitions. State All-Union Standard No 13377-75], 21 pages.

2. Ovcharov, L. A., "Prikladnyye Zadachi Teorii Massovogo Obsluzhivaniya" [Applied Problems of Mass Service Theory], Moscow, "Mashinostroyeniye", 1969, 324 pages.

11,176
CSO: 1863/120

UDC 681.3.01:621.372.5

DESIGN TECHNIQUE FOR DIGITAL SHAPING FILTER WITH FIXED POINT AND MAXIMAL DYNAMIC RANGE

Kiev KIBERNETIKA I VYCHISLITEL'NAYA TEKHNIKA: DISKRETNYYE SISTEMY UPRAVLENIYA in Russian No 53, 1981 (signed to press 16 Jun 81, manuscript received 20 Feb 80) pp 10-17

[Article by A. A. Petrovskiy and A. Ye. Leusenko, Minsk Radioengineering Institute]

[Text] Much attention is now being paid to the design of automated vibration test control systems (ASUV) on digital principles [1, 2], which in contrast to analog systems for this purpose allow with high accuracy forecasting the spectral characteristics of vibration processes and efficiently implementing control and full automation of the process of tests according to a specified program, including according to several.

Digital shaping filters are being used extensively in these ASUV to obtain random vibration processes with specified spectral properties. In doing so, they are implemented as specialized devices with program control [2]. In the majority of cases, simple specialized devices, including digital filters in particular, are built on the basis of representation of binary numbers with fixed point. Inherent to them is overflow of the word length when addition is performed because of the limited dynamic range of the filter [3]. That is why the problem of designing the factors of a program-controlled digital shaping filter in which overflow is eliminated is important.

Discussed below is a technique for designing a shaping filter that allows preventing overflow. In doing so, the maximal absolute value of the input signal may be equal to the upper bound of the filter's dynamic range.

Filter Transfer Function. Implementation of the digital filter by cascade or parallel connection of the sections of the first and second orders provides a higher precision of the position of the poles than the direct and canonical forms. With that, the cascade form yields the least mean noise power at output [4]. Therefore, the digital shaping filter is implemented by series connection of the elementary filters of the first and second orders.

It should be noted that to prevent overflow in the first place, the variation range of the filter factors has to be determined so that they can be represented with a fixed point with the limits of the digital filter's dynamic range, if it is assumed that each register with fixed point represents a fraction with a sign, the modulus of which is less than one.

As a rule, in problems of synthesis operations are performed with the rational function [5]

$$H^2(\omega) = H(-j\omega)\,H(j\omega) = \frac{c_0 + c_1\omega^2 + \cdots + c_n\omega^{2n}}{k_0 + k_1\omega^2 + \cdots + k_m\omega^{2m}}, \qquad (1)$$

with $n \leqslant m$.

It is known [6] that the spectral density at output of a linear system equals the product of the spectral density at input by the square of the modulus of the system transfer function. Without restricting the generality of the discussions, let us assume a signal with a spectral density of one (white noise). Then

$$S_y(\omega) = H^2(\omega) = H(-j\omega)\,H(j\omega).$$

For the function $S_y(\omega)$ to meet the bounds of the transfer characteristic, the conditions [5] must be met. First, that $S_y(\omega)$ be an even fractional-rational function ω with real factors: $S_y(\omega) = C(\omega)/K(\omega)$.

Second, that the degrees of the polynomials $C(\omega)$ and $K(\omega)$ meet the relationship $n \leqslant m$. And third, that the polynomials $C(\omega)$ and $K(\omega)$ be non-negative on the entire real semiaxis ω, i.e. $C(\omega) > 0,\ \omega \in [0, \infty)$ and $K(\omega) > 0,\ \omega \in [0, \infty)$.

After defining a function $S_y(\omega)$, that meets the limits presented, one can find the transfer function of the shaping filter $H(j\omega)$, i.e. the problem of factorization must be solved. The factorization algorithms are presented in detail in [5, 7]. Computation of $H(j\omega)$ by method [5] is easier. According to [5], we have

$$H^2(\omega)\big|_{\omega^2=-p^2} = \frac{\bar{c}_0 + \bar{c}_1 p^2 + \cdots + \bar{c}_n p^{2n}}{\bar{k}_0 + \bar{k}_1 p^2 + \cdots + \bar{k}_m p^{2m}} = H^2(p), \qquad (2)$$

where

$$\bar{c}_0 = c_0,\quad \bar{c}_1 = -c_1,\ \ldots,\ \bar{c}_{2i} = c_{2i},\quad \bar{c}_{2i+1} = -c_{2i+1},\ \ldots;$$
$$\bar{k}_0 = k_0,\quad \bar{k}_1 = -k_1,\ \ldots,\ \bar{k}_{2i} = k_{2i},\quad \bar{k}_{2i+1} = -k_{2i+1},\ \ldots$$

Then, following the factorization algorithm and considering that the shaping filter is in the form of a cascade connection of elementary filters of the first and second orders, we finally derive the filter transfer function in the form of

$$H(p) = \frac{c_n}{\bar{k}_n}\,\frac{\prod\limits_{i=1}^{n_1}(h_{0,i} + h_{1,i}p + p^2)\ \prod\limits_{i=1}^{n_2}(p + \gamma_i)}{\prod\limits_{i=1}^{m_1}(d_{0,i} + d_{1,i}p + p^2)\ \prod\limits_{i=1}^{m_2}(p + \delta_i)}, \qquad (3)$$

where

$$h_0 = \begin{cases} \alpha^2+\beta^2,\ \alpha<0, & \text{if the roots of the numerator of (2) } \alpha+j\beta \text{ and } \alpha-j\beta \text{ are complex-conjugate;} \\ \gamma_1\gamma_2,\ \gamma_1<0,\ \gamma_2<0, & \text{if the roots of the numerator of (2) } \gamma_1 \text{ and } \gamma_2 \text{ are real;} \end{cases}$$

$$h_1 = \begin{cases} -2\alpha,\ \alpha<0, & \text{if the roots of the numerator of (2) are complex-conjugate;} \\ -(\gamma_1+\gamma_2),\ \gamma_1<0,\ \gamma_2<0, & \text{if the roots of the numerator of (2) are real;} \end{cases}$$

$$d_0 = \begin{cases} \bar{\alpha}^2+\bar{\beta}^2,\ \bar{\alpha}<0, & \text{if the roots of the denominator of (2) } \bar{\alpha}+j\bar{\beta} \text{ and } \bar{\alpha}-j\bar{\beta} \text{ are complex-conjugate;} \\ \delta_1\delta_2,\ \delta_1<0,\ \delta_2<0, & \text{if the roots of the denominator of (2) } \delta_1 \text{ and } \delta_2 \text{ are real} \end{cases}$$

$$d_1 = \begin{cases} -2\bar{\alpha},\ \bar{\alpha}<0, & \text{if the roots of the denominator of (2) are complex-conjugate; and} \\ -(\delta_1+\delta_2),\ \delta_1<0,\ \delta_2<0, & \text{if the roots of the denominator of (2) are real.} \end{cases}$$

The transition to the transfer function of the digital shaping filter is effected by means of the bilinear Z-transform [4]

$$p = R[(1-z^{-1})/(1+z^{-1})], \tag{4}$$

where $R = 2/\Delta t$, Δt is the qunatization period.

Substituting (4) in (3) results in the transfer function of the elementary filter of the first order being transformed to the form

$$H(z) = \frac{(R^2+h_1R+h_0)+2(h_0-R^2)z^{-1}+(R^2-h_1R+h_0)z^{-2}}{(R^2+d_1R+d_0)+2(d_0-R^2)z^{-1}+(R^2-d_1R+d_0)z^{-2}} = \frac{R^2+h_1R+h_0}{R^2+d_1R+d_0}\,\frac{1+a_1z^{-1}+a_2z^{-2}}{1+b_1z^{-1}+b_2z^{-2}} \tag{5}$$

Also, a_1 and b_1 are less than two, and a_2 and b_2 are less than one. This follows from all factors h_0, h_1, d_0 and d_1 being greater than zero based on expression (3) and the inequalities

$$\begin{gathered} 2(R^2+h_0+h_1R) > 2(h_0-R^2), \\ 2(R^2+d_0+d_1R) > 2(d_0-R^2), \\ R^2+h_0+h_1R > R^2+h_0-h_1R, \\ R^2+d_0+d_1R > R^2+d_0-d_1R. \end{gathered}$$

Then to implement the digital filter based on representation of binary numbers with fixed point, the factors of the numerator and denominator of (5) must be divided by two. One can show that in this case, the equations describing the operation of the digital filter take the form:
for the direct form

$$(1/2)\,y_n = \sum_{i=0}^{N} a_i x_{n-i} - \sum_{i=1}^{M} b_i y_{n-i}, \tag{6}$$

and for the canonical

$$(1/_2)v_n = x_n - \sum_{i=1}^{M} b_i v_{n-i}, \qquad (7)$$

$$y_n = \sum_{i=0}^{N} a_i v_{n-i}.$$

Here x_n is the input signal and y_n is the output signal of the filter.

Analysis of the dynamic range of the factors of the digital filter of the first order

$$H(z) = \frac{(R+\gamma)+(\gamma-R)z^{-1}}{(R+\delta)+(\delta-R)z^{-1}} = \frac{R+\gamma}{R+\delta}\,\frac{1+qz^{-1}}{1+lz^{-1}}$$

shows that the parameters q and l are less than one. This follows from (3) and the inequalities $\gamma + R > \gamma - R, \delta + R > \delta - R.$

Finally, the transfer function of the digital shaping filter implemented in cascade form takes the form

$$H(z) = C_0 \prod_{i=1}^{L} \left(\frac{a_{0,i} + a_{1,i}z^{-1} + a_{2,i}z^{-2}}{b_{0,i} + b_{1,i}z^{-1} + b_{2,i}z^{-2}} \right) \prod_{i=1}^{P} \left(\frac{1+q_i z^{-1}}{1+l_i z^{-1}} \right). \qquad (8)$$

In this expression, all factors in absolute value are less than one (a_0 and b_0 for all cascades equal 0.5). Thus, they can be represented by binary digits in the form with fixed point in the digital filter.

Selection of Scaling Factors. If x_{max} is the maximum absolute value of the input signal, and y_n and h_k are the output signal and pulse characteristic of the filter, then [8]

$$|y_n| \leqslant x_{\max} \sum_{k=0}^{\infty} |h_k|,$$

since $|y_n| < 1$, then

$$x_{\max} < 1/\sum_{k=0}^{\infty} |h_k| \qquad (9)$$

is the upper bound of the input signal, within which there is no overflow in the digital filter.

Selection of scaling factors by formula (9) is complicated and is not always warranted since it yields overstated results and summing the series in (9) is rather difficult. In the process, the dynamic range of the input signal is reduced.

A more convenient method of scaling the digital filter implemented in cascade form can be found, after requiring the fulfillment of the following conditions:

$$\begin{aligned} K_L^* &= \sup\left|\prod_{i=1}^{L} H_i^*(\omega)\right| = 1, \\ K_{L-1}^* &= \sup\left|\prod_{i=1}^{L-1} H_i^*(\omega)\right| = 1, \\ &\vdots \\ K_N^* &= \sup\left|\prod_{i=1}^{N} H_i^*(\omega)\right| = 1, \\ &\vdots \\ K_2^* &= \sup\left|\prod_{i=1}^{2} H_i^*(\omega)\right| = 1, \\ K_1^* &= \sup|H_1^*(\omega)| = 1, \quad -\pi \leqslant \omega \leqslant \pi, \end{aligned} \tag{10}$$

where K_N^* is the maximum factor of transfer of N cascades.

Here formula (8) is written in the form $H(z) = C_0 \prod_{i=1}^{L} H_i(z)$, where $L = K + P$ is the number of cascades, C_0 is a constant that for simplicity may be assumed equal to one, $H_i(z)$ is the unnormed transfer function of the elementary filter, and

$$H_i^*(z) = \frac{g_{0,i} + g_{1,i}z^{-1} + g_{2,i}z^{-2}}{b_{0,i} + b_{1,i}z^{-1} + b_{2,i}z^{-2}}$$

is the transfer function of the elementary filter with which there is no overflow.

Fulfillment of the conditions (10) leads to the fact that the response of the digital filter when fed a sinusoidal signal of unit amplitude on resonant frequencies $\omega_L, \omega_{L-1}, \ldots, \omega_N, \ldots, \omega_2, \omega_1$ will not exceed one, i.e. the restriction $|y_n| < 1$ is met when x_{max} is as close to one as desired.

With regard to conditions (10), the technique for scaling the factors of the filter (8) is met the following way.

We find the maximum factor of transfer of the first elementary filter $K_1 = \sup|H_1(\omega)|$ and determine the parameters of the transfer function $H_i^*(z)$: $q_{0,1} = a_{0,1}/K_1$, $g_{1,1} = a_{1,1}/K_1$, $g_{2,1} = a_{2,1}/K_1$, with $K_1^* = 1$ on frequency ω_1. Then we compute the maximum value

$$K_2 = \sup|H_1^*(\omega) H_2(\omega)|$$

and find the factors of the second cascade $H_2^*(z)$: $g_{0,2} = a_{0,2}/K_2$, $g_{1,2} = a_{1,2}/K_2$,

$g_{2,2} = a_{2,2}/K_2$, with $K_2^* = 1$ for the resonant frequency ω_2. It should be noted that it is possible that $K_2' = \sup|H_2'(\omega)| > 1$ for ω_2, but in this case too overflow will not occur since the input signal on frequency will be attenuated by the first filter cascade. Similarly, we determine

$$K_{L-1} = \sup\left|\prod_{i=1}^{L-2} H_i^*(\omega) H_{L-1}(\omega)\right|,$$

and

$$g_{0,L-1} = a_{0,L-1}/K_{L-1},\quad g_{1,L-1} = a_{1,L-1}/K_{L-1},\quad g_{2,L-1} = a_{2,L-1}/K_{L-1}$$

$$K_L = \sup\left|\prod_{i=1}^{L-1} H_i^*(\omega) H_L(\omega)\right|,$$

$$g_{0,L} = a_{0,L}/K_L,\quad g_{1,L} = a_{1,L}/K_L,\quad g_{2,L} = a_{2,L}/K_L.$$

In considering this technique for scaling filter factors, we should single out the cases when one of the maximum transfer factors $K_N < 1$. With that, $g_{0,N}$, $g_{1,N}$ and $g_{2,N}$ may become greater than one and the dynamic range of representation of the digital filter factors will be exceeded. To prevent overflow of the word length in this case, it is necessary to select K_N from the condition

$$\frac{1}{a_{1,N}} > \frac{1}{K_N}, \qquad (11)$$

which is derived with regard to the fact that $a_{1,N}$ is the maximum factor of the numerator of the transfer function of the Nth cascade, and the absolute value of the digital filter factors is less than one. In the process, to ensure the maximum possible transfer factor of the filter cascades, obviously, it is necessary to take $K_N = a_{1,N}$. Then

$$g_{0,N} = a_{0,N}/a_{1,N},\quad g_{1,N} = a_{1,N}/a_{1,N} = 1,\quad g_{2,N} = a_{2,N}/a_{1,N}.$$

Selection of the scaling factors of the digital filter implemented in cascade form by (10) and (11) shows that the direct form of building the elementary filter is better than the canonical. It has one summing unit (6), the output of which is the filter output. The canonical form of implementation of the filter has two summing units (7), and with that the values of v_n may be greater than one, since the factors b_i are not normed, which leads to overflow. To prevent it, fulfillment of condition (9) is required, if we put y_n equal to the output value of the first unit v_n, and h_k to the pulse characteristic of this unit. Thus, it is necessary to introduce additional scaling of the input signal of each cascade.

In implementing the cascade form of the digital filter, where the output values of the preceding section are inputs to the following, and the elementary filters are in direct form, we can avoid the shortcoming of increased storage costs compared to the canonical implementation. In this case, N + 2 storage cells are required, where N is the number of storage cells with the canonical implementation of the filter (fig. 1).

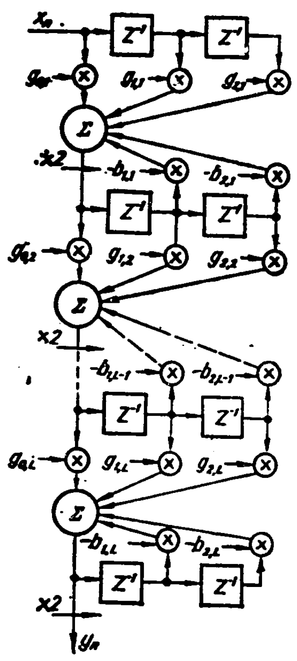


Fig. 1.

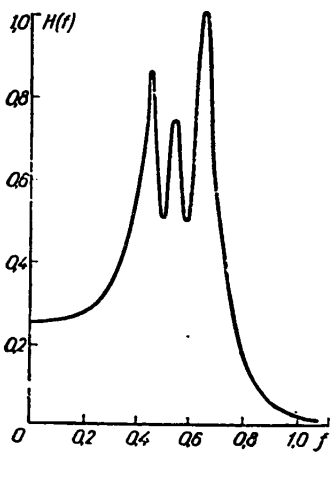


Fig. 2.

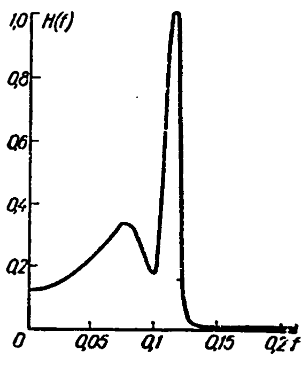


Fig. 3.

Program for Filter Design and Engineering Implementation. The DISHAF (digital shaping filter) program, written in FORTRAN-IV, has been repeatedly tested and was developed from the technique described above for computing the factors of digital shaping filters. The formal procedure for determining the scaling factors, put into the program, is as follows.

1. The maximum transfer factors $\bar{K}_1$, $\bar{K}_2$, ..., $\bar{K}_N$, ..., $\bar{K}_{L-1}$, $\bar{K}_L$ of one, two, ..., L filter cascades are found.

2. The factors of the transfer function $H_1^*(z)$ are determined:

$$g_{0,1} = a_{0,1}/K_1, \quad g_{1,1} = a_{1,1}/K_1, \quad g_{2,1} = a_{2,1}/K_1, \quad K_1 = \bar{K}_1$$

(maximum transfer factor of first two cascades now equals $K_2 = \bar{K}_2/\bar{K}_1$, and $K_1^* = 1$).

3. The factors of the numerator of the second cascade are divided by K_2, i.e. $g_{0,2}$, $g_{1,2}$ and $g_{2,2}$ are found (maximum transfer factor of first three elementary filters, connected in series, is now $K_3 = \bar{K}_3/(K_2\bar{K}_1) = \bar{K}_3/\bar{K}_2$, and $K_2^* = 1$).

4. The factors of the numerator of the third cascade are divided by K_3, i.e. $g_{0,3}$, $g_{1,3}$ and $g_{2,3}$ are found and so on to the last cascade, the numerator factors of which are multiplied by $\bar{K}_{L-1}/\bar{K}_L$. In the process, condition (11) has to be checked at each stage of the computation.

Function (1) is determined by using mathematical programming. In the DISHAF program, the method of fractional-rational approximation is used, and the formal procedure for it is presented in detail in [5].

Shown in figs. 2 and 3 are the amplitude-frequency characteristics of digital shaping filters synthesized by using this procedure. The factors for these filters are given in tables 1 and 2 respectively.

Table 1. Factors of Bilinear Z-Transform for Cascade Digital Shaping Filter of 10th Order When $b_0 = 0.5$

Cascade Каскад	g_0	g_1	[illegible]	b_1	b_2
1	0,2419	0,0095	0,1749	—0.1373	0,3923
2	0,6325	0,3393	0,5588	0,3964	0,4095
3	0,2074	—0,1015	0,0367	0,2032	0,4457
4	0,5154	0,6371	0,1347	0,0856	—0,0189
5	0,4296	0,8593	0,4296	0,6637	0,1637

Table 2. Factors of Bilinear Z-Transform for Cascade Digital Shaping Filter of 4th Order When $b_0 = 0.5$

Cascade Каскад	g_0	g_1	g_2	b_1	b_2
1	0,2105	—0,0725	0,1702	—0,4264	0,2736
2	0,1000	0,1601	0,0600	—0,2044	0,4765

It should be noted that this procedure is very effective when constructing program controllable digital shaping filters as part of digital ASUV [automated vibration test control systems] [2], when the filter factors change at each stage of control and the danger of overflow occurs. In this case, the digital filter is a specialized computing device. Binary numbers are represented with fixed point in two's complement. Elementary filters of the second and first orders are implemented in a form corresponding to direct programming.

Key:
1. code from computer
2. BVPK [paraphase code gates unit]
3. STA [address counter 1, 2]
4. DSh [decoder 1, 2, 3, 4]
5. OZU [storage unit 1, 2, 3, 4]
6. PU [control console]
7. T [trigger]
8. control from computer
9. BU [control unit]
10. GBSh [white noise generator]
11. RG [register 1, 2, 3, 4]
12. M2 [modulo 2 adders unit]
13. SM [expansion unknown]
14. BA [analysis unit]
15. DU [expansion unknown]
16. BR [buffer register]
17. output

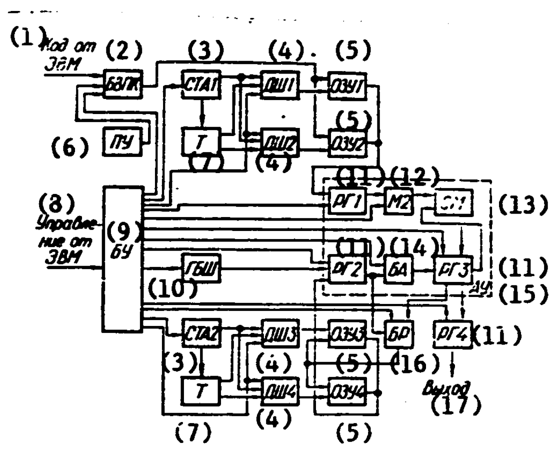


Fig. 4.

Fig. 4 shows a structural diagram of a digital shaping filter with program control by a control computer, that by the procedure presented above computes the factors of the transfer function $H^*(z)$ for a specified spectral density $S_y(\omega)$. Here the filter factors $g_{0,i}$, $g_{1,i}$, $g_{2,i}$ and $b_{1,i}$, $b_{2,i}$ are stored in storage units OZU1 and OZU2 respectively. All storage units are made with K155RU1 integrated microcircuits. Inputs values x_n and output y_n are written to OZU 3 and OZU 4. Using this type of storage requires representation of information to be written in paraphase code. Therefore, the paraphase code gates unit (BVPK) is designed to write to OZU1 and OZU2 the values of the filter factors transferred from the computer that are shaped by using a white noise generator (GBSh).

The buffer register (BR) with the circuit for forming the paraphase code is designed for short-term storage of the values of the input and output codes for their subsequent writing to storage units 3 and 4. The circuit for generating the addresses of storage units 1 and 2 consists of the binary address counter STA1, the trigger (T) and the two decoders DSh1 and DSh2. The circuit for generating the addresses of storage units 3 and 4 has a similar structure. It consists of STA2, T, DSh1 and DSh2. Arithmetic unit operation is based on the method of multiplication from the high-order bits in two's complement [9]. The hardware is implemented with the K155IR1 and K155IM3 microcircuits. Here M2 is the unit of modulo two adders, and BA is the unit for analysis of the next bit of the multiplier. In the process, operation of the cascades is effected sequentially in time in one arithmetic unit and synchronized by the control unit (BU). The output value y_n is stored for the period of quantization in register RG4, the output of which is connected to a digital-to-analog converter.

Manual entry of the factors of the transfer function $H^*(z)$ is provided for in this specialized processor from the control console (PU) to storage units 1 and 2; it is also used for making a test check of the specialized processor. In automatic mode, the device operates under control of signals from the BU [control unit].

In conclusion, it should be noted that this specialized processor can operate in a "pair" with a program controlled digital nonrecursive filter with an arbitrary transfer characteristic [10]. Achieved in doing so is a wide range of retuning by frequency and high precision in generating the spectrum. This procedure for designing a digital shaping filter can be used extensively in digital simulation of normal stationary random processes on mini and micro computers, when fixed-point operation is required to achieve a high rate of information processing.

BIBLIOGRAPHY

1. Chegolin, P. M.; Kuntsevich, V. M.; Tunik, A. A.; Konchak, V. S.; Poyda, V. N. and Borisov, V. F., "Automated System for Control of Vibration Tests on a One-Component Vibration Stand," US i M, No 2, 1978, pp 115-118.

2. Leusenko, A. Ye. and Petrovskiy, A. A., "Computational System for Generation and Control of Matrix of Spectral Densities of Three-Dimensional Random Process," in "Tr. VII Vsesoyuz. soveshch. po probl. upr" [Transactions of the 7th All-Union Conference on Control Problems in Moscow], Minsk, Vol 3, 1977, p 106.

3. "Introduction to Digital Filtration," edited by R. Bogner, A. M. Konstantinidis, Moscow, Mir, 1976, 216 pages.

4. Gold, B. and Reider, C., "Digital Processing of Signals," Moscow, Sov. radio, 1973, 336 pages.

5. Lanne, A. A., "Optimal'nyy sintez lineynykh elektricheskikh tsepey" [Optimal Synthesis of Linear Electrical Circuits], Moscow, Svyaz', 1969, 292 pages.

6. Pugachev, V. S., "Teoriya sluchaynykh funktsiy i yeye primeneniye k zadacham avtomaticheskogo upravleniya" [Theory of Random Functions and Its Application to Problems of Automatic Control], Moscow, Fizmatgiz, 1960, 883 pages.

7. Rozanov, Yu. A., "Statsionarnyye sluchaynyye protsessy" [Stationary Random Processes], Moscow, Fizmatgiz, 1963, 284 pages.

8. Oppenheim, A. and Weinstein, D., "Effect of Final Register Length in Digital Filtration and Fast Fourier Transform," TIIER [PROC. IEEE], No 8, 1972, pp 41-64.

9. Kartsev, M. A., "Arifmetika tsifrovykh mashin" [Arithmetic of Digital Machines], Moscow, Nauka, 1969, 575 pages.

10. Petrovskiy, A. A., "Some Problems of Building a Stationary Random Process Generator Based on Digital Filter," in "Materialy seminara 'Vopr. vibroispytaniya, metody vibroispytaniy, formirovaniye i analiz sluchaynykh vibratsiy'"

[Materials from the Seminar "Problems of Vibration Testing, Methods of Vibration Tests, Generation and Analysis of Random Vibrations], Moscow, TsNIII i TEI [expansion unknown], 1979, pp 32-34.

8545
CSO: 1863/73

UDC 681.3.62.52

ANALOG-DIGITAL AUTOMATIC JAMMING SIGNAL SPATIAL-TIME PROCESSING SYSTEM

Kiev KIBERNETIKA I VYCHISLITEL'NAYA TEKHNIKA: DISKRETNYYE SISTEMY UPRAVLENIYA in Russian No 53, 1981 (signed to press 16 Jun 81, manuscript received 11 Feb 80) pp 45-48

[Article by A. V. Danil'chenko, G. F. Zaytsev and I. A. Izotov, Kiev Higher Engineering Radio Technical School Production-Technical Association]

[Text] An automatic system for spatial-time processing of jamming signals (fig. 1) is discussed in [3]. It consists of an uncontrollable main and N controllable compensating receiving channels, a common adder and device for automatic regulation of weights. Spatial resolution of jamming signal sources is enabled by using a multielement antenna system in the compensating channels and by using a closed system for correlation-time processing, the added jamming signal is suppressed and the useful signal isolated.

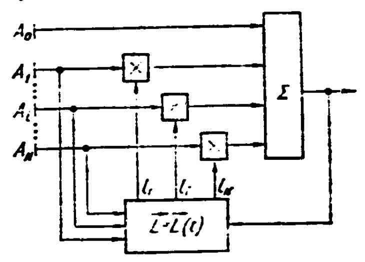


Fig. 1.

The effectiveness of suppression of jamming signals depends on the quality of their spatial resolution (one jamming signal [2, 4] must be processed in each compensating channel) and on the number of compensating channels. Spatial resolution can be enhanced by increasing the number of channels. The correlation-time processing system is a multichannel automatic control system multiply connected on output. Therefore, as the number of channels increases, its dynamic properties deteriorate [1]. And consequently, the quality of suppression of jamming signals by each channel is reduced. Thus, the conditions for high resolution and high operating effectiveness of the correlation-time processing system are contradictory.

Thus, the problem occurs of providing high spatial resolution of jamming signal sources with an acceptable number of channels and the specified effectiveness of jamming signal suppression.

One way of solving this problem is investigated in this article. It consists in the optimal (priority relative to intensity of input signals) control of connecting

$M > N$ elements of the antenna of the compensating channels, the number of which allows obtaining high spatial resolution of the jamming signal sources, to N compensating channels (the number of channels provides a specified level of jamming signal suppression).

To implement optimal control of connecting antenna elements to the channels in the system discussed in [1], it is necessary to add M - N elements of the antenna for the compensating channels, a channel switch and an optimizer for connecting the compensating channels that controls the allocation of channels between the antenna elements.

Let us assume that the time constant τ_C of the channel connection control system is much less than the time constant τ_{AK} of the jamming signal correlation-time processing system. This restriction allows taking the control vector $\vec{L}=(l_1, \ldots, l_N)$ in the correlation-time processing system as a constant independent of time and ignoring the correlation-time processing processes in describing the process of optimizing the spatial filtration process.

With this restriction, the problem of optimal control of connecting the compensating channels is the problem of minimizing the quality functional of the spatial-time processing system of the form

$$Q_{\text{opt}_K} = \min_K Q(x, K, \vec{L} = \text{const}), \quad (1)$$

where $\vec{x} = (x_1, \ldots, x_M)$ is the vector of the input jamming signals and K is a rectangular matrix of commands for connecting the compensating channels with the dimension N x M.

In the system in question, the control quality functional has the form

$$Q = \overline{(\vec{W}^T \vec{x} - \vec{L}^T K W' \vec{x})}, \quad (2)$$

where $\vec{W}^T$ is the transposed column vector of weight factors of jamming signals of the main receiving channel with the dimension M x 1, W' is the matrix of weight factors of jamming signals in the antenna for the compensating channels with the dimension M x M, and

$\overline{\vec{W}^T \vec{x}}$ is the mean value for time τ_{AK} of the composite jamming signal in the main channel:

$$\overline{\vec{W}^T \vec{x}} = \frac{1}{\tau_{AK}} \int_{t-\tau_{AK}}^{t} \vec{W}^T \vec{x}(t)\, dt \quad (3).$$

In expression (2), the product of $\vec{L}^T K W' \vec{x}$ is also the mean value for time τ_{AK} since the devices for correlation-time processing of jamming signals are narrow-band filters with a bandpass proportional to $1/\tau_{AK}$.

To achieve the minimum of the functional (2), it is necessary to maximize the second term in the right part. Maximization of this term is achieved by selecting the optimal values of the vector $\vec{L}$ in the system for correlation-time processing and of the matrix K for connecting the compensating channels in the spatial processing system.

The algorithm for optimization of connecting the compensating channels, which makes use of probability iteration procedures with search optimization, has the form

$$K[n+1] = K[n][1 - \mathrm{sign}(\Delta Q[n] - Q^*[n-1]) - \mathrm{sign}(\Delta P[n] - Q^*[n-1])] +$$
$$+ \|E[s]\|[\mathrm{sign}(\Delta Q[n] - Q^*[n-1]) + \mathrm{sign}(\Delta P[n] - Q^*[n-1])]. \qquad (4)$$

The following designations are used in expression (4):

$$\Delta Q[n] = Q(\vec{x}[n], K[n], \vec{L} = \mathrm{const}) - Q(\vec{x}[n-1], K[n-1], \vec{L} = \mathrm{const}),$$
$$\Delta P[n] = K[n-1]W'\vec{x}[n-1] - K[n]W'\vec{x}[n],$$

where n = 1, 2, ... is the numbers of iterations of the optimization algorithm that are performed with the interval T_n.

The series $\{E[s]\}$ is a series of measuring matrices with the dimension N x M of the form

$$E[s] = \|e_{\alpha,\beta}[s]\|, \; e_{\alpha,\beta}[s] = \begin{cases} 1, & \text{if } \alpha = i, \beta = i + sN, i = 1, 2, \ldots, N, \\ 0, & \text{if } \alpha = i, \beta \neq i + sN, \end{cases} \qquad (5)$$

where N is the number of compensating channels and s = 0, 1, ..., S is the numbers of iterations of the process of measuring the jamming signals that are performed with the interval T_s and the number of the measuring matrix in the corresponding iteration. The number of measuring matrices (S + 1) = M/N, if M is a multiple of N or is equal to the next larger integer to M/N if M is not a multiple of N. The intervals T_n and T_s are associated by the relation $T_n = (S + 1)T_s$.

The current threshold value of jamming signal intensity is denoted by $Q^*[n-1]$ in [4] and the function

$$\mathrm{sign}\, x = \begin{cases} 1, & \text{if } x > 0, \\ 0, & \text{if } x < 0. \end{cases}$$

The functional diagram of the signal spatial-time processing system in question is shown in fig. 2. The optimizer for connecting the compensating channels consists of the jamming signal analysis unit (BAPS), the measuring matrix generation unit (BIM) and the channel switch control unit (BUPK). The jamming signal analysis unit implements algorithm (4) and is used to detect changes in the jamming signal vector $\vec{x} = (x_1, \ldots, x_M)$ that require searching for a new optimal connection of the compensating channels to the antenna elements. The unit for generating measuring matrices E [s] is used to measure the intensities of all jamming signals

Key:
1. PK [channel switch]
2. BUPK [channel switch control unit]
3. BIM [measuring matrix generation unit]
4. BAPS [jamming signal analysis unit]

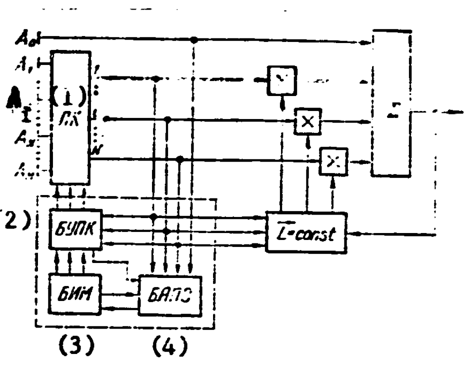


Fig. 2.

affecting the system by using the measuring matrices E [s]. The channel swicth control unit is used to isolate N of M strongest jamming signals and generate optimal matrices for commands to connect compensating channels K [n].

In accordance with expressions (4) and (5), the search for the optimal connection of compensating channels begins when $\mathrm{sign}\,(\Delta Q\,[n] - Q^*\,[n-1]) = 1$ or $\mathrm{sign}\,(\Delta P\,[n] - Q^*\,[n-1]) = 1$ and consists of the following operations:

measurement of all M jamming signals in series of N signals in parallel by using the measuring matrices E [s],

ordering of M jamming signals by their intensity and rejecting of M - N signals of low intensity, and

generation of the matrix K [n] of commands for connecting the compensating channels.

Thus, the process of optimizing the connection of compensating channels reduces to finding the maximal projection of an M-dimensional vector of jamming signals on N-dimensional space. The command matrix generated in the optimization process has the form

$$K\,[n] = \|k_{\alpha,\beta}\,[n]\|;\ k_{\alpha,\beta}\,[n] = \begin{cases} 1, & \text{if}\ \ \alpha = i,\ \beta = j_i; \\ 0, & \text{if}\ \ \alpha = i,\ \beta \neq j_i, \end{cases}$$

where i = 1, 2, ..., N and j_i = 1, 2, ..., N is the number of the antenna element selected for connection to the i-th compensating channel.

Let us consider the operation of algorithm (4). In the initial state n = 0, the compensating channels are disconnected from the antenna, and in the process

$\Delta Q\,[0] = W^*\,[0] \geq 0,\quad Q^*\,[n-1] = 0$ and $\mathrm{sign}\,(\Delta Q\,[0] - Q^*\,[-1]) = 1,$

hence K [1] = $\{E\,[s]\}$, i.e. the search for the optimal connection of the compensating channels begins.

If between the iterations n-1 and n, there began to operate a jamming signal with an intensity greater than Q* [n-1], then $\Delta Q[n] > Q^*[n-1]$, $\Delta P[n] < Q^*[n-1]$, and in the process K [n + 1] = $\{E[s]\}$.

If between the iterations n-1 and n, one or more jamming signals being processed by the compensating channels ceased affecting the system, then
$\Delta Q[n] < Q^*[n-1]$, $\Delta P[n] \geqslant Q^*[n-1]$ and K [n + 1] = $\{E[s]\}$.

All possible alternatives of variation with the course of time of the jamming situation are reduced to the situations discussed above and cause similar responses in the system for optimization of spatial processing.

Thus, this system allows achieving in each compensating channel the processing of one jamming signal, i.e. spatially resolving the jamming signal sources, when the number of jamming signals $N_n \leqslant N$; when $M \geqslant N_n > N$, the system not only resolves the jamming signals, but also optimizes the process of their suppression by priority processing of N jamming signals maximal in intensity; and the engineering implementation of the system is relatively simple. The amount of apparatus increases largely by adding additional elements of the antenna for the compensating channels.

BIBLIOGRAPHY

1. Meyerov, V. M., "Sintez struktur sistem avtomaticheskogo regulirovaniya vysokoy tochnosti" [Synthesis of Structures for Systems of Automatic Regulation of High Precision], Moscow, Nauka, 1967, 423 pages.

2. "Teoreticheskiye osnovy radiolokatsii" [Theoretical Principles of Radar], ed. by Ya. D. Shirman, Moscow, Sov. radio, 1970, 560 pages.

3. Woodrow et al., "Adaptive Antenna Systems," TIIER [PROC. IEEE], No 12, 1967, pp 78-95.

4. Shirman, Ya. D., "Statistical Analysis of Optimal Resolution," RADIOTEKHNIKA I RADIOELEKTRON., No 8, 1961, pp 1237-1246.

8545
CSO: 1863/73

MULTIPROCESSOR SYSTEMS

UDC 681.3

CONFIGURATION OF MULTIPROCESSOR COMPUTER SYSTEMS

Leningrad ARKHITEKTURA MNOGOPROTSESSORNYKH VYCHISLITEL'NYKH SISTEM in Russian 1981 (signed to press 19 Mar 81) pp 2-6, 16-17, 78-94, 103

[Annotation, table of contents, foreword and excerpts from chapter 3 from book "Configuration of Multiprocessor Computer Systems", edited by V. I. Timokhin, Izdatel'stvo Leningradskogo universiteta, 10,000 copies, 104 pages]

[Text]

ANNOTATION

The basic principles of designing multiprocessor computer systems are outlined in the handbook. The most investigated types of hardware structures of MVS [Multiprocessor computer systems] and problems of organizing control of computer processes in them are considered. Modern Soviet and foreign computer systems that implement these principles are described. The book is intended for students of senior courses of electronics and radio engineering vuzes. It may be of interest to specialists in the field of computer technology and computer software.

CONTENTS

FOREWORD

This book is a textbook for students of higher educational institutions that are training specialists in development of computers, their software and application in different fields of science and technology. Its basis is the section of the course of lectures "Digital Computers and Computer Systems," read at the Faculty of Automatics and Computer Equipment, LETI [Leningrad Electrotechnical Institute imeni V. I. Ul'yanov (Lenin)], which is devoted to multiprocessor systems.

The most investigated types of multiprocessor computer systems* and general problems of organizing the management of computer processes in them are considered in the textbook. Main attention is devoted in the book to structures of multiprocessor hardware. Descriptions of specific computers are supplemented by general data on the configuration of multiprocessor computer systems.

As a whole the book should be regarded as an introduction to systematic study of the problems of the configuration of multiprocessor systems. It may also be of interest to specialists in the field of computer technology and computer software.

A large class of systems that contains several processors performing calculations by dependent and independent programs simultaneously are called multiprocessor computer systems. The idea of developing a multiprocessor structure initially appeared upon development of reliable high-speed specialized computers and proceeded from the natural desire to provide simultaneous fulfillment of "independent" parts of the calculation process to achieve the required operational processing. Investigations in this field provided an impetus to development of new structures and principles of organizing control [2, 3, 12].

*Along with the term "multiprocessor computer system," the concept of multiprocessor computer system and the abbreviated term--multiprocessor--are used extensively in the literature. These three terms are subsequently used interchangeably.

Sufficient experience has now been accumulated in design of multiprocessor computer systems. The concept of multiprocessor configuration was formulated.

The variety of structures of multiprocessor computer systems, organization of parallel calculating processes in them and the characteristic features of using these systems comprise a new field of knowledge about computers in modern science. The range of problems considered in the textbook determined its structure. The basic concepts are outlined in Chapter 1 and the main components of the configuration of modern computers are given and multiprocessors are classified. Chapter 2 is devoted to a description of the basic types of structures of multiprocessor computer systems and to comparative analysis of the characteristics and the characteristic features of use. Organization of the Soviet El'brus-1 multiprocessor computer system in whose structure a large number of original solutions is realized, is considered in detail in Chapter 3. The complexity of organizing parallel calculating processes in these systems is shown on the example of this system. Chapter 4 is devoted to organization of calculating processes in multiprocessors. A detailed discussion of systems organization of the operation of multiprocessors, besides analyzing the structure of their hardware, would require consideration of a wide range of problems related to describing the functional conditions of multiprocessor computer systems, methods and software for them. Therefore, only the basic principles of organizing the management of calculating processes in the indicated system are reflected in the proposed book. Problems related to the specifics of designing multiprocessor computer systems are also not touched on in the textbook. The appearance of microprocessors and computer systems based on them is a significant step in the direction of developing multiprocessor structures. Most of the structures considered in the book and developed for "large" systems are reflected in the configuration of multimicroprocessors [13].

Multimicroprocessors with common information mainline and with matrix organization of communications, regular computer systems based on microprocessors (vector and matrix systems), conveyor and other structures of multimicroprocessor systems are designed according to general principles of organizing the hardware of these systems, outlined in Chapter 2. This is also true of organizing control in the given systems. At the same time there are undoubtedly characteristic features in organization of hardware and software components of multiprocessor configuration that require special consideration.

The material of the given book is an introduction of the configuration of multiprocessor computer systems. The authors recommend that readers continue more extensive study of the considered problems in the following basic directions: analysis of the configuration of specific systems, organization of calculating processes in multiprocessor computer systems and characteristic features of multimicroprocessor configuration.

The textbook was compiled by a collective of authors under the scientific editorship of Candidate of Technical Sciences V. I. Ti·okhin. Chapter 1 was written by the scientific editor and co-authored with Candidate of Technical Sciences Ye. A. Metlitskiy (section 1) and O. S. Kozlov (section 2) and sections 3-5 were also written by O. S. Kozlov. Sections 1 and 2 in Chapter 2 were written by Ye. A. Metlitskiy and V. I. Timokhin, sections 3-6 were written by Ye. A. Metlitskiy and A. V. Ekalo and section 7 was written by O. S. Kozlov. Candidate of Technical Sciences A. V. Ekalo is theaauthor of Chapters 3 and 4.

The authors request that all comments and desires on the contents of the book be sent to the address: 197022, Leningrad, Ulitsa prof. Popova, No. 5, Chair of Software and Computer Application.

Table 1.

Computer System	Year of Manufacture	Productivity, million ops/s	Storage Capacity, thousand words
UNIVAC-21	1950-1960	0.008	1
IBM-709	1958	0.2	4- 32
BESM-2	1955-1960	0.015	2
BESM-6	1960-1965	1	32- 131
CDC-6600	1970-1975	12	65- 512
IBM-370/195	1970-1975	7	131- 524
AMDAHL-470/6	1972-1975	7-8	131-1,000
El'brus	1972-1978	1.5-12	164-1,152

Improvement of the characteristics of electronic circuits and components during the past 20 years can be evaluated on the average by a coefficient whose value varies by 10 units per decade. When the latest advances are used in the field of designing the configuration of modern computer systems due to development of effective algorithms and an increase of their productivity and storage capacity, it is possible to achieve an approximately threefold increase in the speed of performing basic operations (instructions) even in large computers.

Thus, performing addition operations on one of the first computers, the MARK-1 (1944), occupied 33 ms, while in 1964 the CDS-6600 expended 300 nanoseconds on the same operation, i.e., the speed of computers had increased 10^6 times during this period. Data that characterize the increase in the productivity and storage capacity of computers are presented in Table 1.

CHAPTER 3

ORGANIZATION OF EL'BRUS-1 MULTIPROCESSOR COMPUTER SYSTEM

1. General Structure of El'brus-1 Multiprocessor

The El'brus-1 multiprocessor computer complex (MVK) is the first powerful general-purpose Soviet multiprocessor computer system. The given computer complex is a typical multiprocessor with cross-switching according to communications topology. However, the presence of a number of interesting features makes more detailed consideration of the structural organization of this system feasible. The contents of this chapter are based on materials of [1, 14], in which the main technical data, configuration characteristics and some concepts that are the basis of the adopted structural solutions are presented.

The El'brus-1 computer complex was the fruit of integral development of hardware and software. A significant increase in the efficiency of the calculating processes of the multiprocessor computer complex was achieved by using a hardware stack, hardware startup of procedures, non-address instruction system and dynamic distribution of high-speed registers.

The multiprocessor computer complex is characterized by essentially unlimited storage capacity and complete input-output autonomy. The organizational structure of the system permits programming to be simplified considerably. The dynamic distribution of resources, context protection of memory, work of several users with common data and recursive use of procedures are automatic in the considered computer complex. The instruction system and the structural organization of the multiprocessor computer complex are designed for programming in high-level language (ALGOL-60, FORTRAN, PL-1, ALGOL-68, SIMULA-67, COBOL and PASCAL), which permits efficient translation of programs. The programming system also includes the El'brus autocode--a procedures-oriented machine-independent language comparable in capabilities to high-level languages. The autocode permits well-structured programs to be constructed.

The overall structure of the El'brus-1 multiprocessor computer complex is presented in Figure 44, where UMB and UML are magnetic drum and magnetic tape control, MB and ML are magnetic drums and magnetic tapes, USD is plug-in disk control, SD is plug-in disks, ATsPU is alphanumeric printers, PK is punch card input-output devices, PL is papertape input-output devices, ATsD is alphanumeric displays, GD is graphic displays and GP is graph plotters.

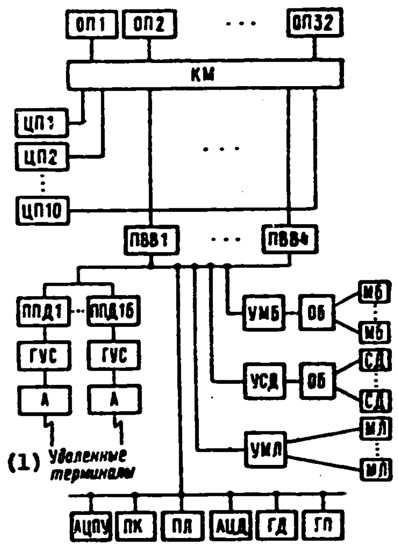


Figure 44.

Key:
1. Remote terminals

Depending on the makeup, the system may include from 1 to 10 central processor modules (TsP), from 4 to 32 internal storage modules and from 1 to 4 input-output processor (PVV) modules. Each central processor and input-output processor module is connected to all internal storage modules through the cross-switching (KM) modules. The cross-switching module is a 4 X 14 commutator and connects four internal storage modules to all central processor and input-output processor modules. The number of cross-switch modules depends on the internal storage capacity and may vary from one to eight. According to configuration, the productivity of the system may vary from 1.5 (one central processor) to 12 million operations per second (10 central processors) and internal storage capacity may vary from 576 (four internal storage modules) to 4,608 Kbytes (32 internal storage modules). The time of executing basic operations in each central processor is characterized by the following values:

adding numbers with fixed point	520 ns
adding numbers with floating point	780 ns
multiplying 32-digit numbers	780 ns
multiplying 64-digit numbers	1,300 ns
logic operations and operations with fields	520 ns

The logic part of the central processor is based on integrated circuits. The internal storage is constructed on ferrite cores. Each of its modules contains eight memory units of 4 K 36-digit words each. These units are standard replacement components. Four internal storage modules connected to a single .

cross-switching module operate with overlapping of access cycles. The cross-switching modules are designed on integrated circuits with switching time an order less than that of central processor circuits. The buffer and external memory, input-output devices and communications lines with users of the system are connected to the system through input-output processors. The given processors receive assignments for exchange through the common field of the internal storage from any central processor and the interaction process itself with the communications channels proceeds without the participation of the central processor. The external storage is connected to the input-output processor channels by special control devices (UU). Several external devices can be connected to one control device through the switchboards of the external devices (exchangers OB). The complex works with communications lines by means of a data transmission processor (PPD), which has its own instruction system and internal storage (up to 16 K words each). The communications line is connected to the data transmission processor through adapters (A) and group integration devices (GUS). The data transmission processor is started by a central processor (through the input-output processor) and then controls data transmission and the operating mode of the line independently by its own program. The multiprocessor computer complex may contain up to 16 data transmission processors --four each per input-output processor.

All the system components operate in parallel, independently of each other, and are dynamically distributed by the operating system for servicing routine tasks. All modules of the same type are identical and independent from the viewpoint of switching the power supply. Each module of the system, including the cross-switching modules, has 100 percent hardware verification and if a single error appears during processing (transmission), it emits a signal on the malfunction of the given module. The operating system cuts out the malfunctioning module from operation by this signal through a hardware-realized reconfiguration system. The cutout module goes into the repair configuration, during which it is repaired by using test-diagnostic programs and special hardware, after which it can be connected by the operating system to operating configuration. This system permits redundancy to be achieved at the level of modules of the same type and ensures given reliability and viability of the entire complex.

2. Organization of Information Processing in Central Processor

Processing of data in the central processor is organized by the principle of stack access to the storage and hardware realization of the stack [14]. The internal language of the multiprocessor computer complex is similar to reverse Polish notation and is a sequence of the names of operands placed in the stack and operation codes performed on operands located in the top of the stack. References to operands or even references to procedures that calculate the value of the required operand may also be located at the top of the stack.

The stack mechanism is used extensively for dynamic distribution of memory for local facilities of program units and procedures. Thus, local variables and files in languages with block structure (for example, ALGOL-60) occur on the basis of their description at the beginning of the block (procedure) and are retained until emerging from it. In this regard the memory should be allocated for local data upon entry to the block and should be freed upon output of it.

Stack discipline is also used effectively to store control information upon conversions to deeper levels of procedure embedding and for storage of information about the address environment of the task during interruptions and switches from task to task. The efficiency of using this stack with procedure transitions is determined by the fact that output is achieved first from the procedure which was started last.

The stack principle of memory access made it possible to realize an economic method of addressing--addressing on the dictionary level, i.e., by the number of the level of embedding of the block (procedure) and by shift--in the El'brus-1 multiprocessor computer complex.

The dictionary level determines the basis of the corresponding procedure while the shift level determines the index with respect to the base. Thus, addressing to local procedure entities is accomplished with respect to its base.

The local variable address is an address pair consisting of the number of the dictionary level and the shift. The procedure base determined by its dictionary level is the address of the beginning of the so-called region of direct-address data (PAD) (Figure 45), corresponding to this procedure. The common region of the direct-addressed data corresponds to each begun but uncompleted procedure in the stack. The presently existing procedure has its own address context or region of name access. This region includes the names of local entities of a given procedure and the names of global entities contained in the encompassing procedures (blocks) according to the block structure of the program.

A base table corresponding to the address environment of the current procedure is used to convert the address pair of the operand to the internal storage address. The base address table is stored in the processor on base registers (BRg) to increase the speed of address conversion. The contents of the base registers should be corrected with each startup of the procedure and completion of it. Descriptors in which the base addresses of access regions of the corresponding levels, the dimensions of these regions and the protection status are indicated, are located in these registers. The address environment of the procedure being executed is determined by the current dictionary level (LU) and by the contents of the base register.

Connecting information is formed and used in the central processor to store the procedure transitions, the block structure of protected procedures and other information required to make the return. This information is located at the beginning of the region of direct-addressed data of the corresponding procedure and occupies two words. The connecting information contains the base address encompassing the procedure, the dimensions of the direct-addressed data region of a given procedure, the base address of a started procedure, the return instruction address, the dictionary level and some additional information related to the operating mode of the started procedure.

The procedure is started in three phases. A place is initially allocated in the directed-addressed data region for storage of the connecting information.

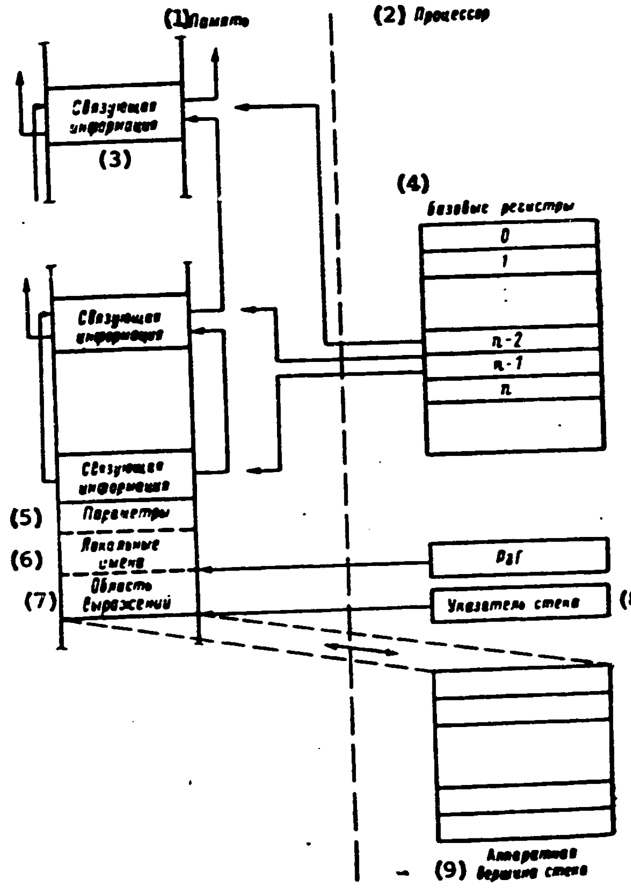


Figure 45.

Key:

1. Memory	6. Local names
2. Processor	7. Region of expressions
3. Coupling information	8. Stack indicator
4. Basic registers	9. Hardware top of stack
5. Parameters	

The actual parameters are then transferred to the started procedure. The parameters are arranged immediately behind the connecting information and occupy the beginning of the PAD region. The parameters are transferred by reading the values of parameters or their addresses to the top of the stack in the case of parameter transmission by name. And finally, the procedure is started. It is accomplished by the start instruction, as the operand of which the procedure mark is used. The mark contains the transition instruction address, the dictionary level and the base address of the encompassing procedure. The start instruction forms the connecting information.

Thus, two lists are formed in the stack. The first (static chain) contains the base addresses of the regions of permissible names for each procedure and serves to form the contents of the base register and the second (the dynamic chain) reflects the sequence of the start of procedures and is used to emerge from them. The base address of the encompassing procedure contained in the marker may indicate the beginning of the PAD region of the encompassing procedure located in the stack or the descriptor file. In the second case the address context of the started procedure is determined by the descriptor file, which may describe the arbitrary regions in the memory.

The entry instruction to the procedure carries the base address of the started procedure to the corresponding base register, examines the static chain or descriptor file, beginning with the address in the marker and, if necessary, corrects the contents of the base register so that they addressed the entire name access region. Control is then transferred to the instruction indicated in the marker. In similar fashion, return to the started procedure is carried out by means of the output instruction and all the required data are taken from the connecting information. After input has been executed, the started procedure itself allocates a place in the stack for its own local names, which are arranged in the direct-addressed data region immediately after the parameters. When calculating expressions, the operands read from the memory by instructions of the "read value" type are loaded into the top of the stack. The results of operations are also left there. The real depth of this part of the stack is shallow. At the same time access to operands in the top of the stack determines the operating speed of the processor when calculating expressions. Therefore, the top of the stack in which nonaddressed intermediate results are placed is realized in the fast registers and is located in the processor. This is the so-called hardware top of the stack (AVS).

The number of registers in the hardware top of the stack can be selected as sufficiently large so that there is no need to pump them to the memory because of overflow. However, during procedure transitions and interruptions, the contents of the hardware top of the stack should be stored in the memory and they should be recovered upon returns. This information is located after the region of the current procedure names, forming its region of expressions. The stack indicator (US) contains the address for pumping the next element from the hardware top of the stack. Only elements from the region of current procedure expressions can be located in the hardware top of the stack. The boundary between the name region and the expression region is established after initiation of local variables by means of a special instruction. The boundary register (RgG) indicates the beginning of the expression region of a current procedure (see Figure 45).

The instruction system of the central processor is nonaddress type. It has a number of advantages over the traditional instruction system of third-generation computers. The most essential of them is elimination of fast registers from the address instruction code to store operands and intermediate results when calculating expressions. The instructions of third-generation computers contained the addresses of the fast registers in explicit form. Therefore, their effectiveness is determined to a considerable degree by the programming method. This is related to medium- and high-productive machines in which individual operations are performed simultaneously. The degree of parallellism in them is sensitive to the distribution of fast registers and accordingly computer productivity is considerably dependent on the extent to which the program fully utilizes the available potential capabilities of parallelling.

The designation of the registers of the top of the stack is performed dynamically in the El'brus-1 multiprocessor computer complex by the hardware method, which ensures greater parallellism than with static distribution. The designation of the registers of the hardware top of the stack is made during execution of instructions. The instruction unit selects and decodes the instruction. The instruction is then fed in stack format to the dynamic register distribution unit (BDRR), which translates it to three-address format and then transmits it to the servo devices. The three-address format is convenient from the viewpoint of storing the initial operands and repeating the operation upon breakdown. Two lists are formed in the dynamic register distribution unit: the list of free registers and a list of occupied registers or the address stack. The first contains addresses of free registers and the second contains the addresses of occupied registered arranged by the stack principle, i.e., the order of the addresses in it determines the stack order of registers of the hardware top of the stack. Two lists are used when forming the instruction of the dynamic register distribution unit.

All data in the El'brus-1 multiprocessor computer complex are identified by tags (features) that determine their type (entire, essential, set, descriptor, indirect word (address), procedure marker and so on) and format (32 digits, 64 digits, 128 digits). The presence of a tag permits one to eliminate from the instruction system superfluous information that indicates which hardware is used to process the information (control algorithm with floating or fixed point, control algorithm for working on integers and so on). When data are fed to the central processor, the apparatus identifies them and according to their type and format dynamically rearranges the algorithms for proce sing these data.

Tags perform simple and complete protection of programs and data. Unlike protection by keys where it is applied to rigidly cut regions of the memory, tags protect data rather than the memory.

The address region of the operating procedure is limited by the descriptors and markers available to it. Only those regions of names whose descriptors are in the base registers at a given moment are accessible in the operating procedure stack. The contents of the base register can be changed only by procedure transition operations. In this case the new context is wholly determined by the transition marker.

Data addressing outside the stack is possible only through the descriptors and indirect words, which does not permit access to regions not described by the address information located in the procedure context. The mathematical memory is separated and the address information is formed only by the operating system. All words that contain an address are identified by apparatus by tags and are protected. The user cannot arbitrarily change the contents of these words (an interruption is issued). "Context protection" of data is achieved by this, during which protection is provided with accuracy up to one word.

And finally, the use of tags permits semantic verification of calculations, since a set of permissible operands is determined for each operation. This considerably simplifies and reduces program debugging.

The most frequently used instructions of the El'brus-1 computer complex has a length of one, two or three bytes. Multibyte instructions are usually direct loading instructions. The operands for the next operation are the top elements of the hardware top of the stack. Instructions can begin from any byte in the word and can be transferred to another word. The most frequently used instruction is "read value." It has two formats (single- and two-byte).

The single-byte format is used for addressing to local names and current procedure parameters. Three digits comprise the operating code and five digits comprise a shift. The dictionary level of the current procedure is used to determine the base. If a five-digit shift is insufficient for addressing to local variables or if one must gain access to global variables, the two-byte format is used. Three digits also determine the operating code while 13 digits determine the address pair. The address pair is the only type of address contained in the instruction code. The address pair is formed as a result of program translation and is actually a name encoded in an optimum manner. This name is converted to a memory address only at the moment an instruction is executed and is determined by the contents of the base registers. Accordingly, addresses in the program code are not dependent either on the position of the program itself in the memory or on the stack position. All variables are arranged in the stack or in the data file described by descriptors. There is only one unmodified program code in the program segments, i.e., all programs in the El'brus-1 computer complex are repeatedly input programs and do not require loading.

Repeated entry permits one to have only a single copy of the program for different tasks. Thus, for example, the same translator program can be used simultaneously by different tasks. Each task has an individual stack and data file but a single common program code. It should be noted that realization of repeated entry of programs in third-generation computers is related to additional expenditures of programmer labor. Taking into account that a place is allocated in the stack when starting each procedure for local data and that the programs are repeated input types, recursive starting of the procedure in no way differs from starting any other procedure.

Special instructions are contained in the central processor that permit automation of access to the elements of multidimensional files. It is assumed that the multidimensional files are arranged sequentially in the mathematical memory. Access to the file elements is gained through the file descriptor, which consists of two words. The first word is a purely standard descriptor containing the address of the beginning of the region, the size of the file, the format of the element and the status of protection. The second word contains the products of index boundaries. The instruction "Read file element" is designated for single access to the file element. Its operands are the element indices and descriptor address contained in the top of the stack. The instruction is executed in the following manner: the resulting index is calculated (in linear representation of a multidimensional file) and is verified for its limit.

If the index does not exceed the size of the file given in the descriptor, the address of the element is determined and reading is accomplished. Since calculation of the resulting index occupies a comparatively long time, the special instruction "read file element in cycle," which optimizes the described circuit, is used upon access to the file elements during the cycle. Cycle counters--one each for each index--are used as indices in this instruction. The address of the file element is calculated for the next cycle and access to the internal storage is gained when executing the instruction "read file element in cycle" simultaneously with reading the current element. The file element called for retention is stored in the zero-access memory of the central processor.

3. Synchronization of Calculating Processes

With respect to apparatus, the calculating process is usually represented by the stack in the internal storage. If a processor (active process) operates in a stack, hardware representation of the process is applied to the base registers, the stack indicator, zero-access storage and other hardware of the processor. The process can be in the status of waiting for some event (passive process) and is represented with respect to hardware only by the data in the internal storage.

The adopted structure of organizing the calculating process offers broad capabilities of parallelling calculations [1]. Thus, along with the fact that an independent stack can be established for each task or complex of jointly translated programs, each procedure can be realized in a separate stack.

The same type modules of a multiprocessor computer complex, specifically central processors, are general-purpose, i.e., any central processor operates in any stack or performs calculations for any task. Moreover, since the system is multiprocessor, there can be several processes working on common data simultaneously in the active state.

One of the pieces of hardware in the El-brus-1 computer complex to provide synchronization of the operation of the central processors and their general-purpose distribution is the use of "semaphores" (Figure 46). Such operations as "open semaphore" and "close semaphore" are entered in the instruction system. The work of the program with common data is preceded by its access to a common cell with a different program, called a semaphore, to close it for this time. If the semaphore is open (a zero in the corresponding digit of the cell), the semaphore is closed (a one is written in this digit), otherwise the process must wait until it is open, i.e., it becomes passive. The processor is freed of executing this process and carries the address of the stack to the semaphore cell. Thus, a line of several wait stacks can be formed on one semaphore.

After work with the data has been completed, the process that closed the semaphore opens it, setting a zero in the corresponding digit of the cell. If the address of the wait stack or a line of stacks is in the address field of the semaphore, the process that opens the semaphore places the stacks in the line of finished processes. If any central processor is freed of work (the process

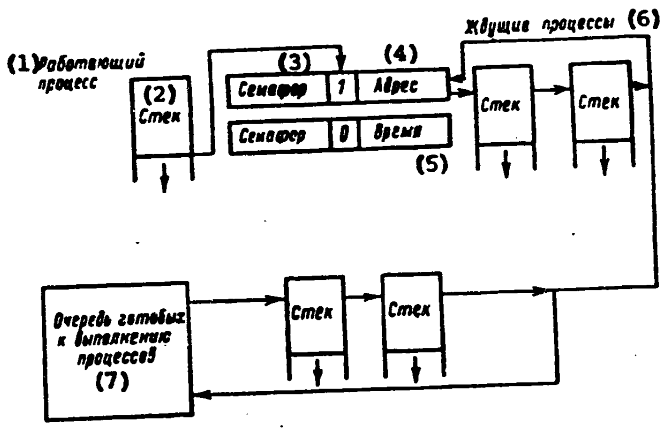


Figure 46.

Key:

1.	Working process	5.	Time
2.	Stack	6.	Waiting processes
3.	Semaphore	7.	Line of processes ready for execution
4.	Address		

on which it was working is converted to the passive process or is completed) it gains access to the line of finished processes and begins to execute the process, standing first in the line of finished processes.

Work of the process is synchronized in similar fashion upon access of it to data located in the external storage. Forming the task for the input-output processor, the central processor closes the semaphore. This semaphore is opened after execution of the corresponding task by the input-output processor. Before gaining access to the indicated data, the process gains access to the semaphore and if the corresponding digit is not a zero, it "hangs" on this semaphore. It is converted to the line of finished processes only after execution of the corresponding task by the input-output processor at the moment the semaphore is opened.

Thus, the passive state of the process can be caused by waiting:

a) completion of access of another process to common data;

b) exchange of necessary data with internal and external storage;

c) freeing of central processor (line of finished processes).

It must be noted that most procedures of the operating system are executed in the stacks of those processes which caused them. Along with this, there are independent processes of the operating system whose stacks are subordinate to common discipline. The place of the process in the line of finished processes is determined by its priority, i.e., it can be changed.

4. Organization of Memory

The efficiency of using the internal storage and accordingly of the external storage is largely dependent on the basic principles of its distribution established in the hardware and operating system. It is obvious that the method of memory distribution which meets the following requirements is the most effective [1]:

a) a data bit with greatest probability of being used at a subsequent moment should be called to the internal storage;

b) the required data bit should be called at the last moment, while the internal storage should have the capability of being freed at any moment after access to it;

c) the data inside the memory should be redistributed or reset to the external storage with the next call;

d) large files should be arranged in different sections of the memory by individual bits.

Taking these circumstances into account, the exchange bit of internal storage with the external storage of arbitrary length (with discreteness up to a word) is used in the El'brus-1 computer complex and the length is determined by the descriptor that describes this data file on the basis of the algorithm. To meet the requirements formulated in item "d" to simplify organization of reset and reuse of the memory, all the data, with the exception of program codes, are loaded into the mathematical memory. The mathematical memory is divided into pages of 512 words each. The operating system determines one or several pages of the mathematical memory (depending on the file capacity) for each data bit described by a descriptor and the first word of the file is arranged in the first word of the page. Thus, if the file is less than 512 words, the page is unfilled and if the file capacity is greater than 512 and less than 1,024 words, two pages are allocated, one of which will be unfilled and so on. If information is located in the physical memory, its capacity, allocated for data, is determined only by the requirements of the descriptor; therefore, packing of the physical memory does not suffer from underutilization of the mathematical memory.

The variables in the procedure code segments are addressed by means of address pairs, while the mathematical address of the value is found during the run by indexing the corresponding base register by shifting the address pair. Rapid conversion of the mathematical address to an equivalent physical address is provided by 32 associative registers of each central processor. If the desired pair in them is not found, an apparatus search is made by the table of user pages and the associative registers are loaded.

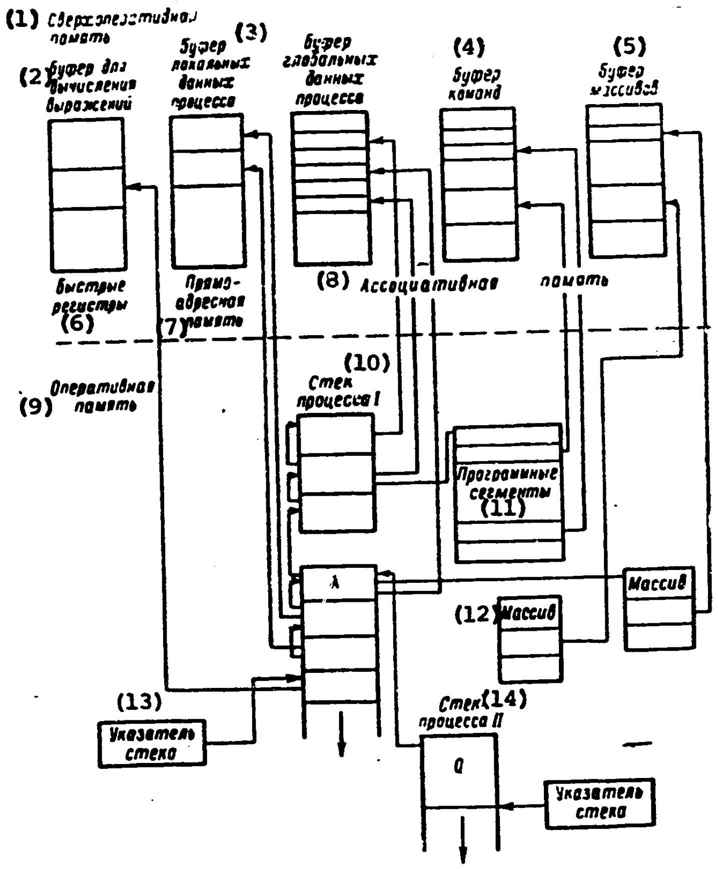


Figure 47.

Key:

1.	Zero-access storage	8.	Associative memory
2.	Buffer for calculation of expressions	9.	Internal storage
3.	Local data buffer of process	10.	Stack of process 1
4.	Instruction buffer	11.	Program segments
5.	File buffer	12.	File
6.	Fast registers	13.	Stack indicator
7.	Direct-access memory	14.	Stack of process 2

Let us consider in more detail the characteristics of organizing the zero-access memory.

The zero-access memory (SOP) distributed by processors and organized on the functional principle (Figure 47) is used in the El'brus-1 computer complex. According to the functional designation, the SOP is divided into five parts:

1) data of the top of the stack being processed in the processor--virtual fast registers that execute stack discipline;

2) local data of process 1--continuous section of the stack memory from stack indicator to data (procedure A) common to process II (procedure Q). Direct addressing to the zero-access memory is possible due to the continuity of the represented section of memory;

3) common data of two processors are represented by the associative memory;

4) the buffer of instructions to be executed, used to reduce the number of accesses to the internal storage during frequently repeated procedures (specifically to accelerate access of instructions in the cycle) is executed on the associative principle;

5) the buffer memory designed for advance hardware pumping of data files for maximum acceleration of working with them is executed on the associative principle.

Design of a local zero-access memory in multiprocessor systems is related to the following difficulties. Upon access to common data of at least two processors, a situation is possible when old data are located in the zero-access memory of the processor while restored data are located in the internal storage. This can be avoided only provided that the writing processor will at least erase the data in the zero-access memory of other processors with each writing to the internal storage. The latter invariably leads to a significant reduction of the information processing speed in the central processor.

Considering each part of the zero-access memory of the El'brus-1 computer complex from the viewpoint of the indicated complex situation, one can note that the memory of the fast registers, local data and instruction buffer are free of this defect. The first and second are due to the locality of data and the third is due to the fixed nature of the instruction codes during their execution. A conflict situation may appear in the associative memory of local data and the file buffer. Upon access to common data, the file buffer can be erased without effect, whereas complete resetting of the global data buffer slows down the calculating process. Therefore, the write time is fixed in it when writing information in these registers simultaneously with data. The opening time is written in its registers when the semaphore is opened. The reliability of data in each global data buffer cell is evaluated clearly by these times.

The method of designing the zero-access memory, adopted in the El'brus-1, is economical since it permits one to reduce the number of buffer cells and makes it possible to transform many of them to direct-address cells (without associative nature), thus reducing the volume of equipment and the access time. Moreover, the adopted organization of the zero-access memory permits parallelling of its operation, which considerably increases the efficiency of using the SOP and also eliminates conflict situations inherent to the given memory designed by traditional methods.

5. Input-Output Subsystem

All control functions of the external memory and terminal devices can be realized through communications lines by one of the central processors. However, the specifics of information processing (only logic processing of low-digit data) makes the use of its equipment inefficient. Therefore, special processors--input-output processors that control external devices and data transmission processors that control terminal equipment through communications lines--are introduced in the El'brus-1 computer complex to perform these functions [1].

The input-output processor controls strictly regulated types of entities, the number of switching versions of which is small. This made it possible to execute all functions of the processor and switching of entities by the hardware method. In this case there is no need for instructions in the local memory and a small zero-access local memory is used for low-digit data processing.

The central processor in the El'brus-1 computer complex is relieved to the maximum of interruptions by external devices by transferring dispatcher functions to the input-output processor in which they are realized by the hardware method. The work of the operating system performed by the central processor is completed by compiling a declaration for access to the external device in which are indicated the information address within the entity, the internal storage address where information must be placed (or from where it must be taken) and its volume. This application of the central processor is placed in line to the descriptor that describes the given device. The descriptor is found in the table by the index corresponding to the ordinal number assigned to this device in the system. The table of devices is stored in the internal storage. After the external device completes the next assignment, the input-output processor makes the corresponding marks in the declaration, removes this declaration from the line to the device and places it in the line of operations completed, after which the input-output processor takes the next declaration in line and organizes execution of it.

The list of executed declarations is processed by the operating system in the central processor. If the declaration is executed without comment, the operating system opens the corresponding semaphore and performs the required operations. In this case if the input-output processor indicated defects in execution of the assignment (it exercised hardware verification and so on), the operating system makes a decision on a second execution of the assignment.

The same external device can operate through any of four input-output processors. Selecting the switching path is determined with respect to hardware in the following manner. When the N first declaration is fed to the device, any input-output processor free at a given moment processes it. The input-output processor is considered free when it does not execute a declaration to change the operating modes of devices. Its work with devices in the multiplex mode does not hinder initial processing of declarations. Subsequent declarations to the device N are processed in the same input-output processor, since the signal on completion of work with the external device comes to the in output processor that started it. The external control device and the corresponding switching channel are selected as it is freed. Individual external devices such as punch cards, printing and magnetic tape that have less influence on the viability of the system do not have special commutators and are rigidly connected to the input-output processor.

The input-output system adopted in the El'brus-1 that realizes with respect to hardware all the control functions of the external devices, including dispatching, redundancy and selection of the working channel, permits a considerable saving of the operating life of the internal storage and a reduction of the number of interruptions of the central processor.

Unlike the input-output processor, the data transmission processor controls a number of entities of different types, the methods of switching of which are essentially unlimited. This leads to the fact that hardware realization of the control function by means of rigidly commutated programs of the data transmission processor places considerable restrictions on the operation of the system in a branched network of terminal devices that function through data transmission lines. The program method of adapting the data transmission processor to the entities controlled by it has been adopted in the El'brus-1 complex and measures have been implemented so that no program corrections are required with variation of switching of the entities and if a new device is introduced, the change of program would be limited to adding a program module that describes the new entity introduced to the system. These principles could be realized due to the following organization of data transmission processor hardware.

Entities operating in the system (including modems, data transmission lines and so on) were described in the handbook of program modules. The connections of the entities are presented in a special table. The operating system formulates a program from the program modules of the entities that ensure interaction with the required terminal device, taking into account its connection to the system, upon access to one of the terminal devices based on the connection table. Thus, if switching of entities changes, changes must be made from one of the terminals to the entity connection table. Since formulation of the operating program and access to the references are of different types and since the formulated working program is used repeatedly when working with a terminal, all the references and tables are arranged in the internal storage, while a local memory is allocated to the transmission processor to store the formulated working programs and their operands. Similar processors in traditional computer systems have no flexible software of their own; therefore, the operating system of the complex performs adaptation functions. This is usually related to complete or partial generation of it.

BIBLIOGRAPHY

1. Burtsev, V. S., "Printsipy postroyeniya mnogoprotsessornykh vychislitel'nykh kompleksov 'El'brus'" [Design Principles of El'brus Multiprocessor Computer Complexes], Moscow, 1977.

2. Golubev-Novozhilov, Yu. S., "Mnogomashinnyye kmopleksy vychislitel'nykh sredstv" [Multimachine Computer Complexes], Moscow, 1967.

3. Yevreinov, E. V. and Yu. G. Kosarev, "Odnorodnyye universal'nyye vychislitel'nyye sistemy vysokoy proizvoditel'nosti" [Homogeneous High-Productive Universal Computer Systems], Novosibirsk, 1966.

4. Zhirov, V. F., "Matematicheskoye obespecheniye i proyektirovaniya struktur EVM" [Software and Design of Computer Structures], Moscow, 1979.

5. ZARUBEZHNAYA RADIOELEKTRONIKA, No 5, 1976.

6. Kagan, B. M., "Elektronnyye vychislitel'nyye mashiny i sistemy" [Computers and Systems, Moscow, 1979.

7. Kattsan, G., "Vychislitel'nyye mashiny sistemy 370" [Computers of the 370 System], Moscow, 1974.

8. Korolev, L. N., "Struktury EVM i ikh matematicheskoye obespecheniye" [Computer Structures and Their Software], Moscow, 1978.

9. Mednik, S. and J. Donovan, "Operatsionnyye sistemy" [Operating Systems], Moscow, 1978.

10. "Mnogoprotsessornyye vychislitel'nyye sistemy" [Multiprocessor Computer Systems], Moscow, 1975.

11. "Mul'tiprotsessornyye vychislitel'nyye sistemy" [Multiprocessor Computer Systems], edited by Ya. I. Khetagurov, Moscow, 1971.

12. "Mul'tiprotsessornyye sistemy i parallel'nyye vychisleniya" [Multiprocessor Systems and Parallel Calculations], edited by F. G. Enslow, Moscow, 1976.

13. Prangishvili, I. V., "Mikroprotsessory i mikro-EVM" [Microprocessors and Microcomputers], Moscow, 1979.

14. Pshenichnikov, L. Ye. and Yu. Kh. Sakhin, "Arkhitektura mnogoprotsessornogo vychislitel'nogo kompleksa 'El'brus'" [Configuration of the El'brus Multiprocessor Computer Complex], Moscow, 1977.

15. RADIOELEKTRONIKA ZA RUBEZHOM, No 4, 1976.

6521
CSO: 1863/97

UDC 681.3

STRUCTURE OF MULTIPROCESSOR COMPUTER HARDWARE

Leningrad ARKHITEKTURA MNOGOPROTSESSORNYKH VYCHISLITEL'NYKH SISTEM in Russian 1981 (signed to press 19 Mar 81) pp 32-41, 70-77

[Excerpts from chapter 2 from book "Configuration of Multiprocessor Computer Systems", edited by V. I. Timokhin, Izdatel'stvo Leningradskogo universiteta, 10,000 copies, 104 pages]

1. Multiprocessor Systems With Common Bus

Multiprocessors with common bus include systems in which all the functional units are connected to a common bus, a connect bus, which is a set of lines used to transmit information signals. The structure of this system is presented in Figure 15. Information is transmitted from one unit to another in this system in batches, each of which should contain, besides data subject to transmission, control information as well, specifically, the address of the unit where the data are being sent. There are no conflicts in the considered structure between several batches coming to some unit simultaneously. The bus contains and transmits only one batch at each moment. All other information sources should await freeing of it. Accordingly, information is transmitted in systems with common bus by bus time-sharing methods.

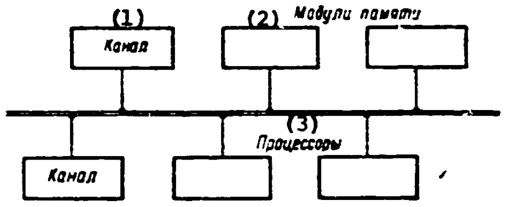


Figure 15.

Key:
1. Channel
2. Memory modules
3. Processors

This structure of the system and the method of exchanging information between its functional units envision the presence of special hardware for each module connected to the bus. The module should contain the required hardware to form the appropriate response upon identification of the unit address in the data pack. This response may include, for example, connection to the bus to receive the information part of the pack if the address in it corresponds to the unit address.

The structure of a multiprocessor system with common bus is determined by the type of bus. The diagram shown in Figure 15 assumes the use of a two-way bus, i.e., one that permits transmission of information signals in both directions. The use of one-way buses in which signals are transmitted only in one direction is possible. The structure of such a system is shown in Figure 16. A special unit--the bus modifier, which, upon receiving signals from the bus of one direction, shapes and transmits them to a bus of another direction--plays a special role in the given system. Comparing the two structures of multiprocessor systems with a common bus, we note that the difference is actually included in realization of one two-way or two one-way interfaces.

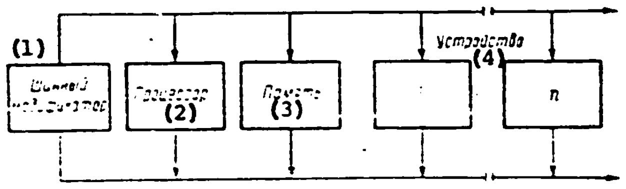


Figure 16.

Key:

1. Bus modifier	3. Memory
2. Processor	4. Devices

Let us consider SM-3 and SM-4 computer complexes (VK)* as an example of a single-bus system. SM-3 and SM-4 machines are related to minicomputers. Like most representatives of this class, they have a low word capacity (16 bits) and the basis of their instruction system is operations on symbols and numbers with fixed point. The structure of the hardware and systems software of these complexes reflect the characteristic features of the configuration inherent to minicomputers.

All SM-3 devices, including the processor and memory, are connected to a common information transmission mainline (Figure 17). This mainline (common bus) is a set of 56 lines (wires) of which 16 are used to transmit data and the remaining ones are used to transmit addresses, synchronous signals, control signals and so on. The given set comprises the nucleus of the interface and is the only method of information transmission in the system.**

*The given complexes are produced in the USSR and are included in the International Small Computer System, used in the socialist countries.

**Additional lines, the number of which is unlimited, can be used in the system. These lines have no effect on the functional characteristics of the interface and are therefore not considered.

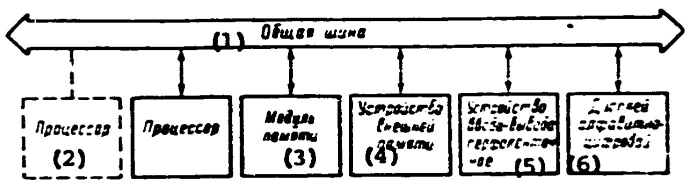

(1) Общая шина
Процессор (2)
Процессор
Модуль памяти (3)
Устройство внешней памяти (4)
Устройство ввода-вывода перфоленты (5)
Дисплей (6)

Figure 17.

Key:

1.	Common bus	4.	External memory device
2.	Processor	5.	Papertape input-output device
3.	Memory module	6.	Alphanumeric display

The universal method of connecting all devices is used in the SM-3 (SM-4) computer complex, which permits a unified algorithm for their interaction to be constructed and accordingly makes it possible to use unified integration hardware. An important element of the configuration of SM-3 (SM-4) machines is the common scheme of addressing the internal storage cells, the internal registers of the processor and the registers of the external devices. A diagram of the address distribution of the SM-3 computer complex is presented in Figure 18.

This "general-purpose" addressing system has a number of significant advantages. The processor can perceive the registers of the external devices as internal storage cells. Thus, no special input-output instructions are required for access to the device and the entire set of address instructions from the instruction system can be used for this purpose. Transmission of data from one device to anotehr (for example, from an external device to the memory) is regarded as transmission from one address to another.

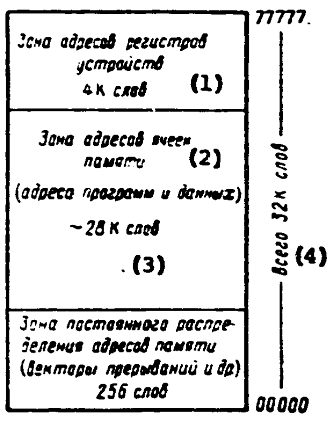

77777
Зона адресов регистров устройств 4 К слов (1)
Зона адресов ячеек памяти (2)
(адреса программ и данных)
~28 К слов
(3)
Зона постоянного распределения адресов памяти (векторы прерываний и др) 256 слов
Всего 32 К слов (4)
00000

Figure 18.

[Key on following page]

[Key continued from preceding page]:

1. Address zone of device registers, 4 K words
2. Address zone of memory cells (program and data addresses), ~28 K words
3. Zone of permanent distribution of memory addresses (interrupt vectors and so on), 256 words
4. Total of 32 K words

Interaction of devices connected to a common bus, as already noted, is organized by time-sharing of bus utilization. Two devices, one of which performs the role of controlling the operation of the bus and the other of which performs the role of the controlled device of bus operation, always participate in any exchange operation. It is obvious that only one device can control a bus at the same moment. A processor, external storage device, video terminal and so on can be used as the devices that control the operation of the bus. Only the internal storage is always the controlled device.

When some device presents a request to control the operation of the bus, it must exchange data with the memory by using direct communications (data transmission outside the processor) or it must interrupt a current program to convert to realization of the interrupt service program by the processor with program control of exchange.

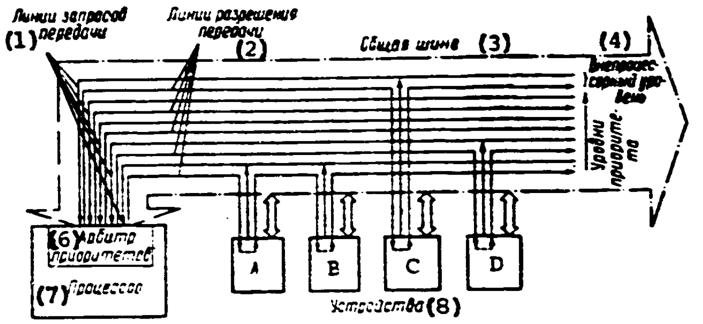


Figure 19.

Key:

1. Transmission request lines	5. Priority levels
2. Transmission authorization lines	6. Priority arbitrator
3. Common bus	7. Processor
4. Extra processor level	8. Devices

The bus distribution between devices is controlled by means of the established priority structure of the devices and the interrupt system. The priority distribution of devices on an SM-3 bus is illustrated by Figure 19, where four priority levels of program interruptions and one level corresponding to exchange outside the processor are indicated.

All the devices emerging as those that control the bus are connected to one of the interrupt request levels. The priority of the levels increases from bottom to top. Thus, devices that present a request to the "direct access request" line (exchange outside the processor) have the highest priority.

The priority of devices connected to a single line is established according to their distance from the processor: the farther from it, the lower the priority. This is provided by sequential propagation of signal over interrupt authorization lines. If a device was not prohibited to a bus, it receives and relays the corresponding interrupt authorization signal to the next device on the same line. The first device on the line that interrogated the bus receives and interlocks the subsequent propagation of the interrupt authorization signal. Thus, the following priority of devices determined by their connection to the corresponding line and position in the system (as distance increases)--C, D, A, B--is established in the diagram in Figures 19.

Taking into account that the processor itself is one of the devices that interrogates the bus, its priority in the system should also be determined. Unlike all the remaining devices, the priority level of the processor is not fixed but is established and changed by program.

The considered configuration of a system with common bus has all the advantages inherent to systems of this type. They are distinguished by high flexibility and simplicity of addition or removal of modules, convenience of organizing data protection by bus monopolization, relatively low cost and so on.

However, several disadvantages can be noted in systems with common bus. Due to the fact that all information flows move in only one direction, the wait time when exchanging data packs as the system is expanded and the load on the bus increases may be impermissibly long. In systems with common bus, even with equal number of processors and storage devices, they are unable to function simultaneously due to time-sharing of the data transmission paths. The relatively low reliability of the system should also be noted since there is always the danger of failure due to bus malfunction.

Some of the noted disadvantages of multiprocessor systems with common bus can be corrected if several buses rather than a single bus are used in the system. As an example, let us consider a system with several buses designed on the basis of the SM-4 computer complex (Figure 20).

The characteristic features of this system are the presence of a common internal storage field, common access to external storage devices, the capability of making the processor, memory and peripheral devices redundant and providing conditions for parallel joint operation of all processors. All this is provided by special intrasystems communications hardware--an interprocessor communications adapter that performs cross-communications of processors and a bus switch for switching common bus 3.

The structure of the multiprocessor system in Figure 20 approaches a cross-switched system in topology.

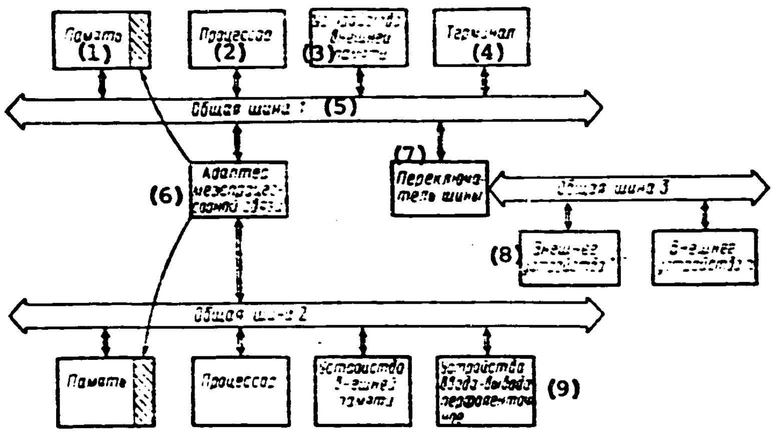


Figure 20.

Key:

1. Memory	6. Interprocessor communications adapter
2. Processor	7. Bus switch
3. External storage device	8. External device
4. Terminal	9. Papertape input-output device
5. Common bus	

2. Cross-Switched Systems

Multiprocessor systems designed on the principle of communicating between modules by means of a "rectangular array" of connect lines, which can make contact at any intersection point, are called cross-switched systems (Figures 21). This organization of the system permits contact to be established between any two units of the system for the entire time of information exchange. Unlike time-shared switching realized in systems with a common bus, the considered method of communications switching is frequently called space-shared switching.

The cross-switch is "non-interlocked" in the sense that transmission through it cannot be prohibited due to the absence of transmission paths. There is the capability of establishing several information transmission paths simultaneously in this system. At the same time one should bear in mind that the switch can be blocked if one of the connected devices is already occupied.

One of the early structures in which the cross-switched principle was realized was the system which has been called a "polymorphous computer" (Figures 22) [12]. Computer modules that include processor and memory units can communicate with peripheral devices through a central switchboard. An attempt was made in the given system to organize connections directly between processors and cross-access to the memory by short-circuiting the corresponding number of

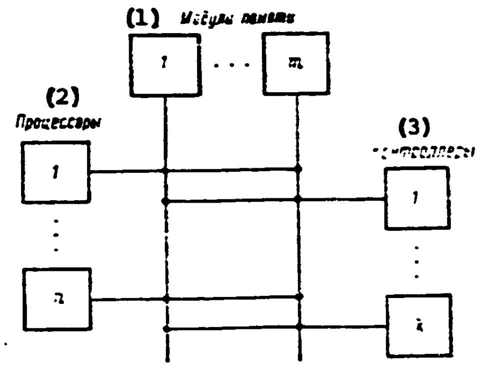


Figure 21.

Key:
1. Memory modules
3. Processors
3. Controllers

intersections. The complexity of this method of communications between processors and memory units and the low utilization efficiency of equipment (the processor and memory of one module having a single communications bus "interfere" with each other) determine the disadvantages of the "polymorphous computer" structure compared to the structure of the system presented in Figure 21.

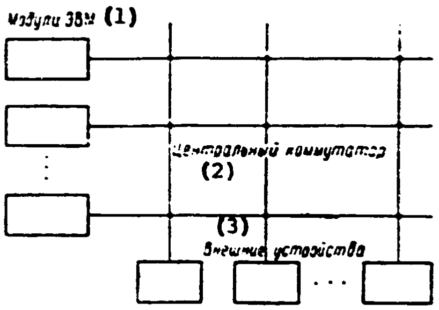


Figure 22.

Key:
1. Computer modules
2. Central switchboard
3. External devices

Multiprocessor cross-switched systems, having somewhat lower flexibility than systems with a common bus, nevertheless permit comparatively simple introduction of new modules if the switching matrix has adequate capacity. The matrix is completely isolated from the other functional units and can also be designed on the modular principle, which permits expansion of it. However, due to the complexity of switchboard functions, its structure may be considerably complicated.

An additional switching matrix of input-output devices can be introduced in the system to provide greater flexibility and to increase possibilities for expansion. This switchboard is connected to the central switchboard through the input-output control processors (Figure 23), and in this case the input-output devices can be connected to any channel. The considered structure of multiprocessor systems is used in large computer systems of the Burroughs Company (United States).

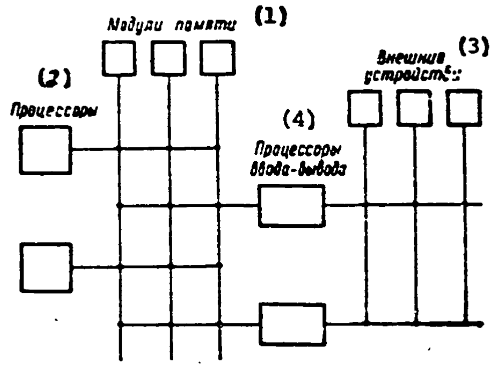


Figure 23.

Key:
1. Memory modules
2. Processors
3. External devices
4. Input-output processors

An original version of organizing multiprocessor configuration has been proposed for the Multi-Interpreter system of the Burroughs Company, in which a group of processor units with microprogram control was introduced. By reloading the microprogram memory units, the same modules are used as central processors or as input-output controllers. Because of this, all the processors, memory modules and peripheral devices are connected to a common switching matrix (Figure 24).

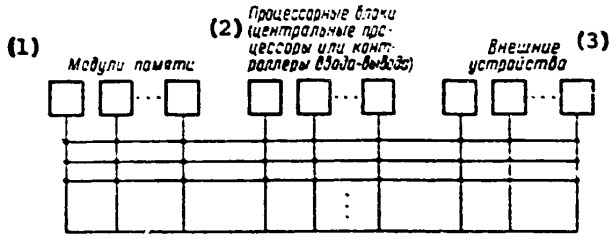

(1) Модули памяти
(2) Процессорные блоки (центральные процессоры или контроллеры ввода-вывода)
Внешние устройства (3)

Figure 24.

[Key on following page]

[Key continued from preceding page]:

1. Memory modules
2. Processor units (central processors or input-output controllers)
3. External devices

Besides the foreign computers mentioned, the El'brus-1 high-production computer system [8] and the SM-2 computer complex--one of the models of the SM EVM [International Small Computer System] are multiprocessor cross-switched systems. Chapter 3 will be devoted to the El'brus-1 system.

Let us consider in more detail the structure of the SM-2 computer complex (Figure 25). The modular structure of the switchboard is used in this system. Eight-channel (KMR-8) and four-channel (KMR-4) switchboards provide intrasystems communications between devices of the given computer complex. A common shared switchboard used to realize total matrix switching of each processor and the direct computer access channel (KPDP) with each internal storage device (UOP) and input-output matching device (SVV) that performs the role of controller, is designed on their basis.

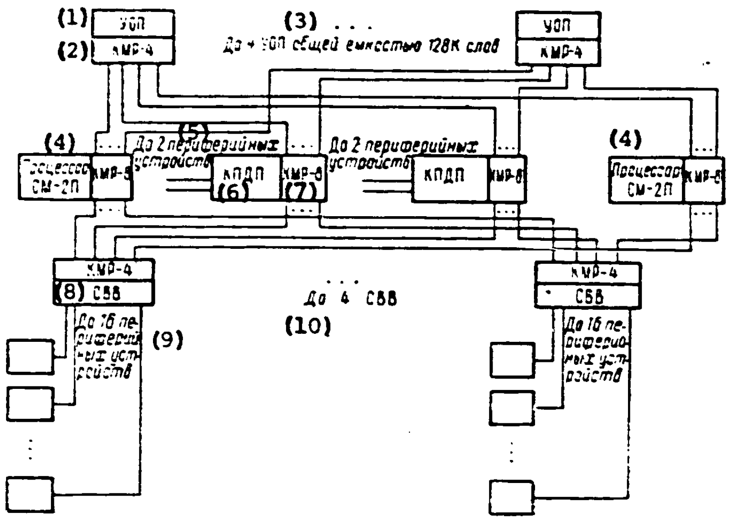

(1) УОП
(2) КМР-4
(3) . . . До 4 УОП общей емкостью 128К слов
УОП
КМР-4
(4) Процессор СМ-2П
КМР-8
(5) До 2 периферийных устройств
КПДП (6)
КМР-8 (7)
До 2 периферийных устройств
КПДП
КМР-8
(4) Процессор СМ-2П
КМР-8
КМР-4
(8) СВВ
До 16 периферийных устройств
(9)
. . . До 4 СВВ
(10)
КМР-4
СВВ
До 16 периферийных устройств

Figure 25.

Key:
1. Internal storage device
2. Four-channel commutator
3. Up to four internal storage devices with total capacity of 128 K words
4. SM-2P processor
5. Up to two peripheral devices
6. Direct memory access channel
7. Eight-channel commutator

[Key continued on following page]

[Key continued from preceding page]:

8. Input-output matching device	10. Up to four input-output matching devices
9. Up to 16 peripheral devices	

The direct memory access channel is a device that provides rapid information exchange between the internal storage device and the peripheral devices. It performs input-output operations independently of the processor. The mutual influence of these devices is manifested only in the attempt at simultaneous access to the same memory module. In this case priority is offered to the channel, while operation of the processor is delayed for one access cycle to the memory. Information can be exchanged with devices connected directly through the direct memory access channel at a speed up to 1,100,000 bytes per second. The channel can service one device connected directly through the direct memory access channel or no more than four input-output devices connected through the input-output matching device simultaneously. In the latter case the exchange speed is lower and comprises up to 550,000 bytes per second.

The input-output matching device has 16 outputs per interface, which permits up to 16 peripheral devices to be connected through one unit. The input-output matching device is connected by the KMR-4 to the processors and direct memory access channel.

Consideration of the structure of the SM-2 computer complex again permits one to note the main advantages of multiprocessor cross-switched systems in which information exchange is possible over several data transmission paths simultaneously. In this case the effective transmission speed can be higher than, for example, in a time-shared system with a common bus since contact is established between the interacting modules for the entire information exchange time. Because of this system of organizing communications, there are no problems in parallel operation of processors. The interfaces of individual units are simplified in a multiprocessor cross-switched system since data addressing and resolution of conflicts occurring upon access to a single module from several sources are accomplished by switching matrix logic.

The occurrence of conflicts in the switching matrix is at the same time the main cause of a reduction in the efficiency of cross-switched multiprocessors. Delays of access to the memory caused by the fact that it is used by other processors or input-output devices reduce the speed of the processors and accordingly of the system as a whole.

7. Specialized Multiprocessor Computer Systems Based on Standard Modules

The classification of multiprocessor computer systems presented in [12] does not encompass the large class of specialized multiprocessor computer systems which are used extensively in different automated and automatic control systems.

The structure of the hardware of specialized multiprocessor computer systems is effected to a significant degree by the requirements on these systems, determined by their designation and by their operating conditions.

Let us consider specialized computer systems designed on a set of standard modules. These systems have found application as onboard multiprocessor computer systems in aircraft and spacecraft.

Expansion of the range of problems solved onboard aircraft and spacecraft of different designation by using digital computer systems would inevitably require an increase in the number of autonomously operating onboard computers or the use of large multiprocessor computer systems that contain one or several storage modules.

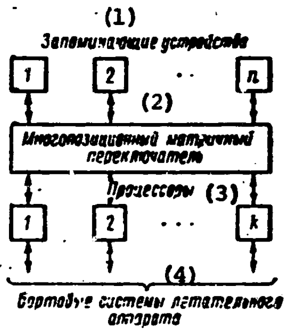

(1)
Запоминающие устройства
1
2
n
(2)
Многопозиционный матричный переключатель
Процессоры (3)
1
2
k
(4)
Бортовые системы летательного аппарата

Figure 40.

Key:

1. Storage devices
2. Multiposition matrix switch
3. Processors
4. Airborne systems

A number of autonomous computers is used mainly during the first stages to solve the problem of airborne system control. However, the practice of development and operation of airborne control systems soon adequately determined the considerable advantage of the second method and made it possible to reduce the overall dimensions and mass of the system and power consumption. At the same time realization of this direction in practice became possible due to successful development of the theory of organization and control of computer processes and the theory of multiprogram and multiprocessor information processing during the second half of the 1960s.

One of the possible versions of structural organization of an airborne multiprocessor computer system (BMVS) is illustrated by Figure 40.

The airborne multiprocessor computer system contains a number of modular units (processors P1, P2, ..., Pk) of the same type which can be connected to several storage modules by means of a multiposition matrix switch.

An increase in the reliability of the given airborne multiprocessor computer system is achieved by using reserve processors, which perform the volume of calculations in case of failure of the main processors or which perform the more important tasks according to given priority by redistribution of tasks among operating processors.

This structural organization of the hardware of an airborne multiprocessor computer system permits computer equipment of the system to be developed with given requirements on capacity, calculating load and productivity according to specific tasks and phases of flight.

Introduction of standard plug-in modules (processors and storage devices) simplifies the repair and maintenance of airborne multiprocessor computer systems during operation. Having available reserve components on a common basis, one can design "self-repairing" assemblies and units. "Self-repair" is accomplished by means of special hardware and software (reconfiguration equipment). The need for manual repair of such an airborne multiprocessor computer system occurs if the main or reserve components fail or if there is interference that causes the main or reserve modules to fail. The advantages of automatic repair consists in increasing the reliability of airborne multiprocessor computer systems and a significant reduction of the maintenance and repair time.

A malfunction and failure detection logic, switching to the reserve module group and also a catalog of malfunction features are provided in the malfunction detection programs to ensure the capability of "self-repair" and repair of airborne multiprocessor computer systems.

Automatic repair is accomplished by special programs after detection of the failure. The nature and location of the malfunction are determined on the basis of data on the failure by examining the catalog.

One of the first airborne multiprocessor computer systems was the aircraft computer system AADC (United States) [15]. The absence of standardization and unification in development of the airborne multiprocessor computer system leads to the appearance of different modules differing by software, which in turn complicates checking, repair and operation of the computer systems.

The main requirement in development of the general-purpose airborne digital computer AADS was to develop standard equipment modules and software on the basis of which airborne computer systems of different configuration could be designed with wide range of basic specifications. All models of the developed family of airborne computer systems should be compatible with respect to software. The programs for all models of these systems are compiled in the same high-level language and are translated to an entity code corresponding to a specific configuration, while program compatibility is observed only from bottom to top, i.e., the programs written for any model are suitable for a model with higher calculating capabilities. For example, the program for a small airborne computer system can be used in an airborne multiprocessor computer system, but not vice versa.

Configurations that include 16- and 32-digit small airborne computer systems, simplex processors and multiprocessor organization can be developed on the basis of the set of standard modules.

When developing the AADS airborne computer system, the main tasks were as follows:

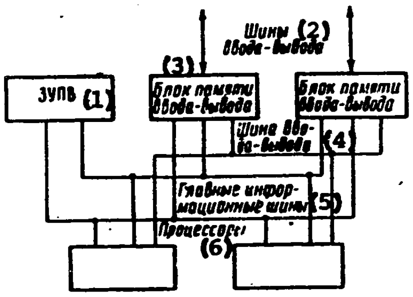


Figure 41.

Key:

1. Random-access storage device	4. Input-output bus
2. Input-output buses	5. Main information buses
3. Input-output memory unit	6. Processors

development of high and medium degree of integration of a family of functional modules based on integrated circuits which provide reduction of the time and labor expenditures on design, manufacture and maintenance of airborne computer systems of future aircraft;

producing airborne computer systems of different complexity based on a minimum number of standard modules that meet the requirements placed on United States military aircraft computer systems during the 1980s;

achieving a high level of reliability of these airborne systems by using highly reliable electronic components, built-in monitoring devices and introduction of hardware redundancy;

development of airborne computer systems with storage capacity and productivity that exceed the values required for a specific application, which permits a considerable reduction of programming expenses;

creation of a library of standard software modules which can be used in airborne computer systems of different aircraft when they are performing different tasks.

A two-processor airborne computer system designed on a set of standard modules of the AADS system is presented in Figure 41. One of the basic components in functional organization of the airborne multiprocessor computer system is the transmission buses shown in the figure. These buses, having redundancy, provide two-way information exchange between all functional modules. The evenness of the message code is monitored in all buses in the exchange mode and message transmission is repeated if an error is detected. If a double error is determined in the message code, an interrupt signal is automatically generated.

The data processing components (processors) receive data through the information buses from the input-output memory units, process the data and return them for storage or output to other subsystems.

A random access storage device (ZUPV) with page organization is provided for program and data storage. A page of information containing 256 words is transmitted to the processor by the hardware to which the programmer does not have access.

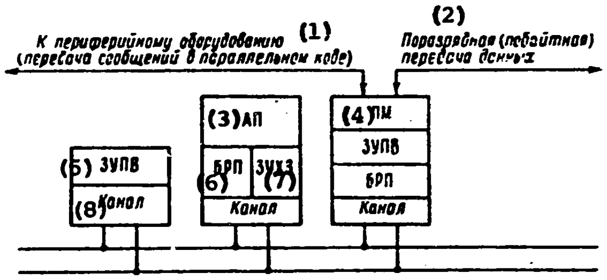

(1) К периферийному оборудованию (передача сообщений в параллельном коде)
(2) Поразрядная (побайтная) передача данных
(3) АП
(4) ПМ
ЗУПВ
БРП
Канал
(5) ЗУПВ
(8) Канал
(6) БРП
(7) ЗУХЗ
Канал

Figure 42.

Key:
1. To peripheral equipment (message transmission in parallel code)
2. Digit (byte) data transmission
3. Arithmetic processor
4. Peripheral module
5. Random access storage device
6. Program distribution unit
7. Task storage device
8. Channel

A block diagram of an airborne multiprocessor computer system with configuration of a simplex processor that includes all the basic components of a standard set of modules is presented in Figure 42. The channel provides for integration of all components of the system and contains logic elements required for operation of the buses. In the general case it integrates two buses.

All the components of the airborne multiprocessor computer system are connected to buffer devices through secondary buses, each of which ensures data transmission in one direction. These 16-digit buffer devices integrate the secondary buses to the main information bus. During data transmission, the transmitting unit transmits the next message only after confirmation of the correctness of reception of the previous message has been received through the feedback line. If an error is detected, the incorrectly received message is transmitted a second time.

The information exchange control functions are distributed between separate channels. This design version permits expansion or reduction of the number of channels without any equipment modification. However, a controller that performs centralized control can be put into operation if needed. In this case the operating reliability of the device is increased with a reduction of the flexibility of channel use since the total number of contacts is reduced.

The data processing element is a general-purpose program controlled processor which performs logic and arithmetic operations required when solving sequentially occurring problems. This module is the main one in a simplex processor and in an airborne multiprocessor computer system.

The processor consists of a program distribution unit (BRP), arithmetic processor (AP) and task storage device (ZUKhZ). The capacity of the storage device can reach 65 K 36-digit words. The capacity of the task storage device of a simplex processor is equal to 4 K words. The length of a word comprises 36 digits, of which 32 are information digits and 4 are allocated for byte verification for evenness. The required pages of programs to be fulfilled are requested from the ZUPV and are entered in the task storage device by the entire page.

The functions of the program distribution unit include selection of instructions and operands and also performance of such operations as conditional jump and information exchange with other modules through the main information bus. Arithmetic and logic operations are performed by the arithmetic processor. Sharing of arithmetic and control operations ensures a high degree of parallelism in operation.

The productivity of the considered processor for a program is equal to two million operations per second, 20 percent of which consists of multiplication operations and 80 percent of which consists of addition operations.

Several pages of the ZUPV may be required to store the pages of the program related to some specific task. It is not compulsory that these pages be arranged sequentially. Information about the arrangement of pages in the ZUPV related to a given task are contained on the first page.

The first page is transmitted by the operating system to the processor task storage device prior to solution of a specific task. All the required pages are then transmitted independently by the processor upon completion of the task without participation of the operating system. The time required for transmission of a single word through the main information bus in this mode is equal to 150 nanoseconds.

The ZUPV provides only reading of information in the operating mode. It is written in a given device during preparation of the airborne multiprocessor computer system for operation. Moreover, all the information is stored in the ZUPV when the power supply is cut off.

The random access storage device included in the input-output memory unit consists of standard modules with capacity from 8 to 16 K 36-digit words. The operation of this device is controlled by a program distribution unit, similar to that of the processor. Data awaiting processing and those already processed and also data prepared by one program for use in another are stored in this storage device. The read-write cycle of the storage device is equal to 225 nanoseconds in this mode. Depending on operating conditions and the requirements placed on it, a storage device with or without destruction of information when the power supply is cut off can be used.

The task storage device has the same characteristics, but the requirements of information storage when the power supply is cut off are not placed on it.

In a simplex procecessor the random access storage device is designed on the basis of MOP [Metallic oxide semiconductor] components of n type with loss of information when the power supply is cut off. However, a storage battery is provided for the case of failure in the power supply circuits.

The input-output control functions are performed by the peripheral module (PM). Information integration of the airborne multiprocessor computer system with airborne and other systems is provided through this module. The given module also contains a sequential multiplex bus and byte and word transmission buses. The peripheral module has direct access to the memory, which accelerates execution of input-output operations with minimum participation of the operating system.

The AADC airborne computer system differs from most traditional computers by the following properties:

instructions programmed in high-level language of type APL are realized directly by hardware;

the type of information word is designated by three of 36 digits, one of which determines the accuracy of achieving the result. Before being entered into the adder, all the operands on which arithmetic operations are performed are converted to a format corresponding to execution of actions with double accuracy with floating point (64 digits are the mantissa and 8 digits are the exponent). After the operation is performed, the result is converted to the initially established format;

virtual addressing and memory protection is formed. Virtual addresses are converted to physical addresses in the following manner: a 16-digit address is divided into two 8-digit fields. Digits 0-7 of the virtual address indicate one of 256 cells in the page table. The 20-digit address of the first word of the page containing the required word is stored in this cell. Adding digits 8-15 of the virtual address to this address, one can find the real address of the required word. The pages are addressed in similar fashion with the exception of adding digits 8-15 of the virtual address. The entire page is read from the storage device in this case.

The memory is protected on the page level. In this case several digits of the contents of the page table cell determines the accessibility of the pages for reading or writing information.

A number of different configurations of the AADC airborne computer system, of which the main ones are the small airborne computer system, simplex processor and multiprocessor, can be created from a standard set of functional modules.

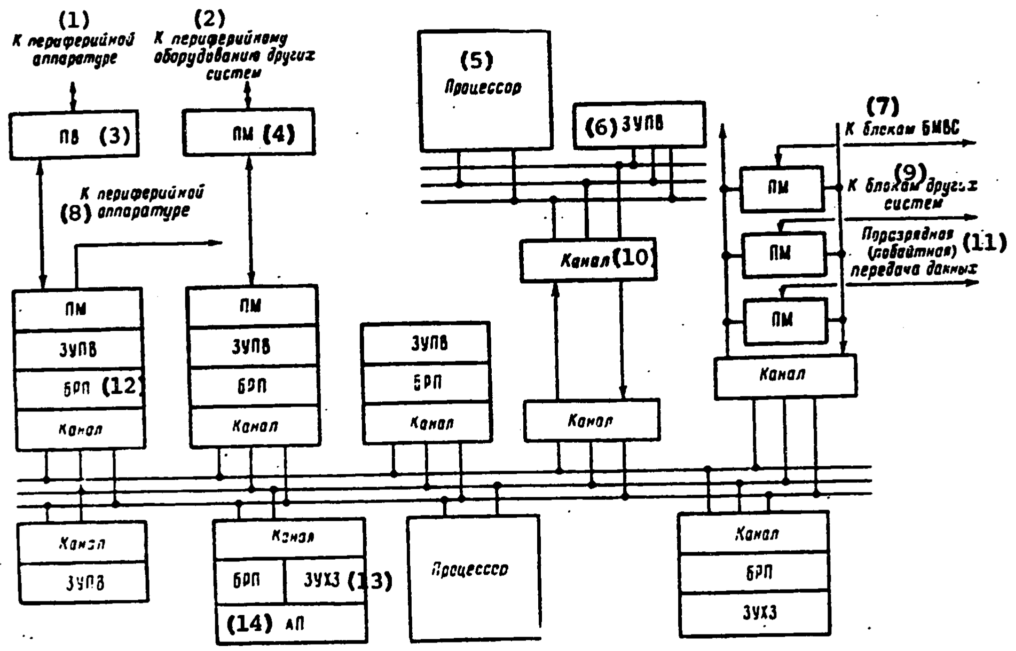

(1) К периферийной аппаратуре
(2) К периферийному оборудованию других систем
ПВ (3)
ПМ (4)
(5) Процессор
(6) ЗУПВ
(7) К блокам БМВС
(8) К периферийной аппаратуре
(9) К блокам других систем
Канал (10)
Поразрядная (побайтная) передача данных (11)
ПМ
ЗУПВ
БРП (12)
Канал
ЗУХЗ (13)
(14) АП
Процессор

Figure 43.

Key:
1. To peripheral equipment
2. To peripheral equipment of other systems
3. Random access
4. Peripheral module
5. Processor
6. Random access storage device
7. To units of airborne multiprocessor computer system
8. To peripheral equipment
9. To units of other systems
10. Channel
11. Digit (byte) data transmission
12. Program distribution unit
13. Task storage device
14. Arithmetic processor

The small airborne computer system performs arithmetic operations with and without regard to characters and has a flexible addressing system which provides direct, indirect and absolute addressing and indexing. The instruction system provides operations on digits, bytes, half-words and complete words. The instruction and data formats of this system correspond to those used in more complex configurations of airborne computer systems. The maximum data input-output speed during parallel operation of all integration devices may reach $3.3 \cdot 10^6$ thousand operations per second with regard to the capability of direct access to the memory. The productivity of this configuration is 500,000 operations per second.

Two-level organization of memory with virtual addressing of the operands and procedures is used in the simplex processor. The processor--a data processing component--performs functions of the central data processing unit.

The multiprocessor, a block diagram of which is presented in Figure 43, is designed for use in ground or shipboard tactical control systems.

More complex configurations of airborne computer systems can also be developed on the basis of the standard set of modules. For example, a decentralized small airborne BMS controlled by a simplex processor corresponds to the requirements placed on aircraft computer systems operating in real time. This configuration of an airborne MVS is characterized by high flexibility and gradual deterioration of parameters in case of failures and has a simple operating system.

CS-4 high-level languages and a special-purpose program and standard macroprogram file are used to write the programs.

6521
CSO: 1863/97

UDC 681.3.06

CONTROL MULTIPROCESSOR WITH DISTRIBUTED OPERATING SYSTEM

Kiev KIBERNETIKA in Russian No 6, Nov-Dec 81 (manuscript received 23 Feb 81) pp 138-139

[Article by Robert Israil'yevich Belitskiy, candidate of engineering science, junior scientific associate, Aleksandr Vasil'yevich Palagin, doctor of engineering science, department chief, and Valeriy Iosifovich Sigalov, candidate of engineering science, senior scientific associate, all with the Institute of Cybernetics, UkSSR Academy of Sciences, Kiev]

[Text] A promising trend in using microprocessors is the use of them as a homogeneous element base in designing high-throughput computing systems. Among the various types of existing and planned multimicroprocessor systems (MMPS) with various functional orientation and architecture [1-3], we can isolate the class of systems intended for control that operate in real time. For these systems, reliability and speed indicators are especially important. The MMPS discussed in this work is one of these systems. In its architectural and structural organization, it belongs to the class of systems with a common (or time-shared) bus and a distributed operating system.

The system is made up of elementary processors, each of which includes a microprocessor, local storage (ZU) and buffer circuits that provide the outlet for the internal bus of the processor that interfaces the cited assemblies with the system common bus. By the common bus, the elementary processors are interfaced with each other, with common storage and with the collectivized input/output devices.

The MMPS, operating under control of the distributed operating system (OS), can function in two modes. In the first mode, used when the microprocessors have sufficient amounts of their own storage, copies of OS are stored in each local storage unit. In the second mode, used when the microprocessors have small amounts of their own storage, a single copy of OS is stored in common storage and time-shared by all system microprocessors. In any mode, each microprocessor's storage holds the program of the current branch, being executed, of the source parallel program and local data, while common storage holds the object code of the source program, jointly usable data and control tables.

The MMPS functions as follows. In the first (preparatory) phase, the parallel program in the microprocessor extended assembler is translated by using a cross-assembler on an instrumental computer. The special object code obtained that contains the microprocessor instructions in machine code and some additional field is loaded into the system common storage. Formed at the same time in common storage are the control tables needed for the distributed OS operation. The control tables consist of three information files: a node table, a list of active nodes and a table of inoperative processors. The node table is a trilevel structure. The first level corresponds to all nodes (i.e. PARBEGIN statements) in the source parallel program, the second to all branches or groups of branches (in the case of parallel loops) subordinate to the corresponding nodes, and the third level contains the addresses of the programs and semaphores of the branches, flags for the parallel loops, names of expected nodes, etc. The list of active nodes contains the addresses of the subtable of the nodes corresponding to the statements PARBEGIN and PAREND, which enclose the branches executable at a given time. The inoperative processor table contains the names of the branches in the top level, the execution of which was not completed because of system processor failure.

The presence of the preparatory phase is conditioned by the specific nature of the use of the MMPS for control problems, the list of which is known in advance (practically, it does not change during system operation).

The system is started after the program and source data are laoded into common storage. In a special case, the program can be placed in common storage in advance, for example when the system is manufactured.

The microprocessors access common storage for programs of accessible branches. After loading the programs found into their own storage, the microprocessors operate in accordance with the programs of their branches, accessing the operating system module (time-shared or their own) in executing the special statements in extended assembler.

The MMPS extended assembler includes special statements that permit description of parallel programs:

PARBEGIN, PAREND [4] are the delimiters of the group of branches executable in parallel;

BEGIN, END are branch delimiters;

PARALLEL, FOR, END [5] are delimiters of branches executable in a parallel loop;

P (semaphore), V (semaphore) [4, 6] are synchronizing primitives;

WAIT $\left(\begin{cases}\text{Boolean variable}\\ \text{real variable = arithmetic expression}\end{cases}\right)$ are wait statements (special cases of the AWAIT statement [7]);

LET $\left(\begin{cases}\text{Boolean variable}\\ \text{real variable = arithmetic expression}\end{cases}\right)$ are assignment statements (generalization of the SIGNAL statement [6]);

INTERRUPT is the statement for restarting a branch or the entire parallel loop;

COMMON, SEMAFOR are descriptors of data and procedures stored in common storage and of binary semaphores; and

TEST, CHECK are descriptors of test branches and groups of test statements.

The functions of the first six statements are evident. The next six statements (synchronization statements) allow the programmer to flexibly describe the interaction between branches, and the system to efficiently implement these interactions in the shortest possible time by the structural facilities included in the elementary processor. These facilities enable associative exchange of information between the elementary processors executing the program branches by the names of these branches. In particular, this permits implementation of synchronization statements without a wait queue [4, 6], simplifies the operating system considerably and speeds up program execution. The P and WAIT statements, just as Hansen's [7], are executed each time the wait variables change right up to the execution of the corresponding conditions. The V and LET statements signal that such changes have occurred.

The INTERRUPT statement (branch name) resets and restarts the processor executing the indicated branch (in the case of a single branch) or group of processors executing a parallel loop with the name specified in the statement. In doing so, if the interrupted branch has subordinate branches, they are interrupted and the processors that were executing them are released for other operations.

As a rule, microprocessors do not have checkability, much less the capability of identifying their own failure. To enhance the validity of multiprocessor system operating results, a known check method can be used [8]: time check by timer signals, self-checking by using tests, hardware redundancy (duplication) or an aggregation of these methods. Using the first method in multiprocessor systems leads to the necessity of including a timer in each processor and consequently, to a considerable increase in the apparatus. Using the second and third methods results in system throughput reduction.

To reduce throughput losses in the system, the capability has been provided to dynamically form pairs of elementary processors backing each other up in executing program branches specified by the TEST descriptor.

In executing a branch with the TEST descriptor, the two processors exchange information only before each output to the common bus, since only through it can a processor disrupt the operation of the other processors and the system as a whole. If a programmer considers it advisable to check not just the statements associated with the output to the external buses, he uses the CHECK descriptor (label name) which enans that the statements following it are checked right up to the statement with the label specified in the descriptor. When a mismatch occurs between the processors, appropriate entries are made in the control tables and the processors test each other. The faulty processor is removed from the system, and the good one continues calculation, after finding another partner.

Thus, the described control multimicroprocessor system is general-purpose. With respect to software, it is oriented to the previously specified concrete algorithm and operates in real time. System input language is assembler, supplemented by a number of special statements that permit representing parallel algorithms, speeding up program execution when combinatorial search procedures are available in it, and ensuring reliable functioning of the system and its dynamic reconfiguration. The microprocessor's operating system is distributed and stored either in each processor's own storage (when it is large enough) or in common storage together with the translated source parallel program, common data and other control information. The programs are processed in parallel dynamically based on data on availability of branches and free processors. Optimization in parallel processing is not performed.

BIBLIOGRAPHY

1. Glushkov, V. M.; Ignat'yev, M. B.; Myasnikov, V. A. and Torgashev, V. A., "Rekursivnyye mashiny i vychislitel'naya tekhnika" [Recursive Machines and Computer Technology], Preprint No 74-57, UkSSR Academy of Sciences Institute of Cybernetics, 1974, 26 pages.

2. Marchuk, G. I. and Kotov, V. Ye., "Modul'naya asinkhronnaya razvivayemaya sistema" [Modular Asynchronous Developable System], Preprint No 86, USSR Academy of Sciences Siberian Branch Computer Center, Novosibirsk, 1978, 48 pages.

3. Enslow, P. H., ed., "Multiprocessors and Parallel Processing," Moscow, Mir, 1976, 383 pages.

4. Dijkstra, E., "Interaction of Serial Processes: Programming Languages," Moscow, Mir, 1972, 87 pages.

5. Kotov, V. Ye., "Theory of Parallel Programming: Application Aspects," KIBERNETIKA, No 1, 1974, pp 1-16.

6. Hoare, C. A. R., "An Operating System: Structuring Concept," COMMUNS ACM, Vol 17, No 10, 1974, pp 549-557.

7. Hansen, P. B., "Structured Multiprogramming," COMMUNS ACM, Vol 15, No 7, 1972, pp 574-578.

8. Rollard, P. R., "Designing Very Reliable Microprocessor Systems," ELEKTRONIKA, No 1, 1979, pp 73-80.

8545
CSO: 1863/91

SOFTWARE

UDC 681.3

APPROACH TO DESCRIBING FUNCTION OF BRANCHING TO NEXT MICROINSTRUCTION IN A MICROPROGRAM

Moscow PROBLEMY UPRAVLENIYA V TEKHNIKE, EKONOMIKE, BIOLOGII in Russian 1981 (signed to press 13 Aug 81) pp 45-49

[Article by B. S. Arutyunyan and G. M. Pogosyants]

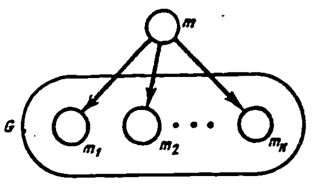
[Text] In automating microprogramming, the problem arises of automatic assignment of addresses to microinstructions when the microprograms are placed in control storage. Naturally, these addresses are far from arbitrary and must correspond to the addressing mechanism selected in a given microprogramming system. It is known that when the current microinstruction is executed, the address of the microinstruction is computed that is fetched by this address from control storage and executed following the current one. There is exceptional diversity in methods of organizing computation of the next address. Attempts have been made to classify these methods [1], but they do not cover the entire diversity of addressing methods since the addressing mechanism is selected during design of the control device in accordance with specific requirements that can vary greatly. In the selection, or more precisely the design, of the addressing method, compromise solutions are often sought since the different requirements of efficiency have to be met, which presupposes a certain element of creativity.

All this to a great extent hinders automating the placement of microprograms in control storage. In fact, writing a program for each inidividual case is very inefficient, but creating a universal program prevents diversity in addressing methods.

In this article, an attempt has been made to overcome this difficulty. A certain regular method for describing the addressing mechanism is proposed (from now on, let us call the is mechanism the addressing function). This description can subsequently be interpreted by the appropriate program.

The idea of this method is based on the concept of the branch group [2]. As is known, the addressing function must correspond to the logical structure of the microprogram, i.e. if from a given microinstruction m there is a branch to a group of microinstructions m_1, m_2, ..., m_k in microprogram M, the addressing function that implements this branch must provide each time the address of that microinstruction to which the branch from m has to be made (see the drawing). Consequently, the addressing function must implement all branches that are encountered

Branch from microinstruction m to group G of successor-microinstructions



in microprogram M. Methods of implementing a specific branch may vary greatly. Analysis of addressing functions encountered in practice has shown that the description of the method for generating the addresses A' of the successor-microinstructions can be limited to the address A of the current microinstruction, some fixed set B of bits in the microinstruction itself (the so-called address bits) and the constants, i.e. the address of each successor m_i of the microinstruction m is described by the formula

$$A'_{m_i} = f_i\,(A_m, B_m, \text{const}), \tag{1}$$

where i = 1, 2, ... k.

FOROS language facilities [3] are used to describe the addressing function. The variables A', A and B are treated as registers with a specified number of bits. Let us cite examples of addressing function descriptions for clarification.

Example 1. Let us consider a certain method of addressing that allows effecting a branch to a group of four microinstructions m_1, m_2, m_3 and m_4. The control storage word address length is 13 bits. The zero through seventh bits in the microinstruction are the address bits. Thus, we can define the registers

$$A'\ (\varnothing:12),\ A\ (\varnothing:12),\ B\ (\varnothing:7).$$

The addresses of the successor-microinstructions are generated the following way: the contents of the bits from the eighth through the twelfth match the contents of those same bits of the address of the predecessor-microinstruction; the contents of the bits from the second through the seventh match the contents of those same bits in the address portion of the predecessor-microinstruction. Also, the following requirement must be met: a) if the zero bit of the address portion contains a zero and a certain condition 1 is met, the zero bit of the address of the next microinstruction gets the value "zero," and when the condition is not met, the value "one"; if the zero bit of the address portion contains a one, then irrespective of condition 1, the zero bit of the address of the next microinstruction receives the value "one"; b) if bit one of the address portion contains a zero and a certain condition 2 is met, bit one of the address of the next microinstruction gets the value "zero", and when the condition is not met, the value "one"; if bit one of the address portion contains a one, then irrespective of condition 2, bit one of the address of the next microinstruction gets the value "one."

Obviously, bits zero and one of the address portion have to be zeros for a branch to occur to the group of four microinstructions.

The addressing function description will look like this:

$$A'(8:12) = A(8:12); \quad M1 : A'(\varnothing:1) = \varnothing\varnothing B;$$
$$A'(2:7) = B(2:7); \quad M2 : A'(\varnothing:1) = \varnothing 1 B;$$
$$B(\varnothing:1) = \varnothing\varnothing B; \quad M3 : A'(\varnothing:1) = 1\varnothing B;$$
$$M4 : A'(\varnothing:1) = 11B;$$

For compactness in this description, those bits of the addresses of the microinstructions of the branch group that are identical with all four microinstructions are described one time. The labels M1, M2, M3 and M4 denote all four microinstructions in the branch group, and the label numbers must match the numbers of the corresponding microinstructions in the symbolic microprogram [2].

Example 2. Let us discuss another addressing method. The control storage word address length is also 13 bits. The address portion contains 12 bits, i.e. we have

$$A'(\varnothing:12),\ A(\varnothing:12),\quad B(\varnothing:11).$$

The branch group microinstruction addresses are generated the following way: the contents of the bits from zero through the second of the address portion are 010; the contents of trigger T1 are assigned to address bit 12; the contents of bits 4 through 11 match the contents of bits 4 through 11 of the address portion; the content of bit 3 is the result of the disjunction of the contents of trigger T2 and bit 3 of the address portion; bits 0 through 2 are generated by register R $R(\varnothing:2)$, the contents of which are moved into the bits mentioned.

It is easy to see that if the content of bit 3 of the address portion is "zero," the branch group may contain a maximum of 32 microinstructions, but if it is "one," then just 16 microinstructions. Let us assume this bit is zero, then the branch function description looks like this:

$$B(\varnothing:2) = \varnothing 1 \varnothing B;$$
$$B(3) = \varnothing B;$$
$$A'(4:11) = B(4:11);$$
$$M1 : A'(12) = \varnothing B;$$
$$A'(\varnothing:3) = 1\varnothing\varnothing\varnothing B;$$
$$M2 : A'(12) = \varnothing B;$$
$$A'(\varnothing:3) = 1\varnothing\varnothing 1B;$$
$$M3 : A'(12) = \varnothing B;$$
$$A'(\varnothing:3) = 1\varnothing 1\varnothing B;$$
$$M4 : A'(12) = \varnothing B;$$

$$A'(\varnothing:3) = 1\varnothing 11B;$$
$$M5 : A'(12) = \varnothing B;$$
$$A'(\varnothing:3) = 11\varnothing\varnothing B;$$
$$M6 : A'(12) = \varnothing B;$$
$$A'(\varnothing:3) = 11\varnothing 1B;$$
$$M7 : A'(12) = \varnothing B;$$
$$A'(\varnothing:3) = 111\varnothing B;$$
$$M8 : A'(12) = \varnothing B;$$
$$A(\varnothing:3) = 1111B;$$
$$M9 : A'(12) = 1B;$$

$$A'(\varnothing:3) = 1\varnothing\varnothing\varnothing B;$$
$$M1\varnothing : A'(12) = 1B;$$
$$A'(\varnothing:3) = 1\varnothing\varnothing 1B;$$
$$M11 : A'(12) = 1B;$$
$$A'(\varnothing:3) = 1\varnothing 1\varnothing B;$$
$$M12 : A'(12) = 1B;$$
$$A'(\varnothing:3) = 1\varnothing 11B;$$
$$M13 : A'(12) = 1B;$$
$$A'(\varnothing:3) = 11\varnothing\varnothing B;$$
$$M14 : A'(12) = 1B;$$
$$A'(\varnothing:3) = 11\varnothing 1B;$$
$$M\,15 : A'(12) = 1B;$$
$$A'(\varnothing:3) = 111\varnothing B;$$
$$M16 : A'(12) = 1B;$$
$$A'(\varnothing:3) = 111B;$$

It is easy to see that all the lines for description of the addressing functions take the form of formula (1), and what is actually described is the structure of the branch group to a degree that is fully sufficient for automatic placement of a symbolic microprogram [4] and the technical method for implementation of the addressing mechanism is disregarded (in example 1, a check of conditions is performed, and in example 2, registers T1, T2 and R are used, but this is not reflected in describing the addressing functions).

After automatic placement of microprograms in control storage, simulation of the operation of the computer device is performed by the conventional facilities of FOROS [5], and here, naturally, the addressing mechanism is also simulated since it is implemented in the control unit, which permits checking the correctness of both microprogram placement and functioning of the unit as a whole.

In conclusion, let us give in full form the format of descrining the addressing function.

```
ADRESS FUNCTION $F$;
$A'(\ldots) = f_1(A(\ldots), B(\ldots), \mathrm{const})$;
        $\vdots$
$A'(\ldots) = f_n(A(\ldots), B(\ldots), \mathrm{const})$;
| $F(U_1)\ F_1$;
| $F(U_2)\ F_2$;
  $\vdots$
| $F(U_m)\ F_m$;
 $F_1 : A'(\ldots) = f_{j_1}(A(\ldots), B(\ldots), \mathrm{const})$;
              $\vdots$
      $A'(\ldots) = f_{j_k}(A(\ldots), B(\ldots), \mathrm{const})$;
$M_1 : A'(\ldots) = f_{j_p}(A(\ldots), B(\ldots), \mathrm{const})$;
              $\vdots$
$M_{i_1} : A'(\ldots) = f_{j_t}(A(\ldots), B(\ldots), \mathrm{const})$;
RETURN;
              $\vdots$
 $F_m : A'(\ldots) = f_{l_1}(A(\ldots), B(\ldots), \mathrm{const})$;
              $\vdots$
      $A'(\ldots) = f_{l_k}(A(\ldots), B(\ldots), \mathrm{const})$;

 $M_1 : A'(\ldots) = f_{l_p}(A(\ldots), B(\ldots), \mathrm{const})$;
              $\vdots$
$M_{i_m} : A'(\ldots) = f_{l_t}(A(\ldots), B(\ldots), \mathrm{const})$;
END;
```

Here in terms of F_1, F_2, ..., F_m are denoted the subfunctions of the addressing function F or the descriptions of the various types of addressing that are encountered in a given microprogramming system; U_1, U_2, ..., U_m designate certain conditions, after checking of which a particular type of addressing is selected that is used for the branch from the current microinstruction; for simplicity, the periods are used to denote the corresponding bits of registers A', A and B.

Naturally, it is not always necessary to use the full format for describing the addressing function.

With this apparatus for describing addressing functions, one can construct a program that interprets a microprogram (written in symbolic language) and a description of the addressing function that is rather simple and entirely standard and which performs automatic placement of microprograms in control storage [4]. The addressing function just has to be described the appropriate way for each microprogramming system developed.

BIBLIOGRAPHY

1. Veytas, V. I. and Zhintelis, G. B., "Typical Fragments of Addressing Structures of Microprogram Control Units," UPRAVLYAYUSHCHIYE SISTEMY I MASHINY, No 4, 1975.

2. Arutyunyan, B. S., "Problem of Analysis of Structure of Microprograms and Function of Branching to Next Microinstruction," VOPROSY RADIOELEKTRONIKI, No 8, 1977.

3. (Landow, I. Ya.), "Using Digital Computers to Design Digital Computers," Moscow, Energiya, 1974.

4. Arlazarova, A. V.; Arutyunyan, B. S.; Yefstifeyeva, T. I. et al., "Microprogram FOROS-ASSEMBLER-MIF," TR. INEUM [Institute of Electronic Control Machines], No 66, 1977.

5. Arlazarova, A. V.; Arutyunyan, B. S.; Yefstifeyeva, T. I.; and Pogosyants, G. M., "Simulating Microprograms in the FOROS System: Materials from the Seminar 'Development, Operation and Evolution of Systems for Computer-Aided Design of REA [Radioelectronic Equipment]'," Moscow, Nauka, 1978.

8545
CSO: 1863/77

UDC 62-50

TECHNIQUE FOR DESIGN AND DEBUGGING OF SOFTWARE FOR PRODUCTION LINES

Moscow PROBLEMY UPRAVLENIYA V TEKHNIKE, EKONOMIKE, BIOLOGII in Russian 1981
(signed to press 13 Aug 81) pp 96-101

[Article by S. M. Golubeva, I. V. Speranskaya and R. I. Shubina]

[Excerpt] Building automated discrete control systems on a base of small computers is prolonged considerably because of the complexity of designing and debugging the applications software (PO), which is governed by the necessity of planning the computing process in real time with the limited resources of computer storage. At the same time, developers of small systems often do not possess sufficient programming skills. In connection with what has been said, it is advisable to equip newly produced small computers with a complex of problem-oriented program and methodological facilities that facilitate development of applications software for classes of single-type systems.

The aim of this work is to create a complex of that type of facilities for production line discrete control systems (SDUPL) and thereby simplify the transition from the SDUPL algorithm to the debugged applications program when the PS-300 [1] is used as the base computer.

8545
CSO: 1863/77

SELECTED ITEMS FROM JOURNAL 'ALGORITHMS AND PROGRAMS', AUGUST 1981

Moscow ALGORITMY I PROGRAMMY in Russian No 8, Aug 81 pp 1-136

[Excerpts]

3193. Methods of debugging devices and systems with use of microprocessor control. Basov, Ye. P.; Golovin, S. S. and Gokhberg, A. G. VOPR. RADIOELEKTRON. SER. EVT, No 4, 1980, pp 3-14. Bibl.: 8 titles. Problems of constructing hardware and software facilities for debugging programs of microprocessor systems, their advantages and disadvantages.

3231. Method for solving problems on arrangements. Bukharayev, N. M.; Litvinov, I. A. and Tagirov, T. S. PRIYEM I OBRAB. INFORM. V SLOZH. INFORM. SISTEMAKH, No 10, 1980, pp 68-77. Bibl.: 3 titles. ALGOL program is described for calculating arrangement of geometric objects by these methods: Monte Carlo, paired rearrangements and constructive. The latter method yields results closer to the optimal value with least machine time. Thus, the Monte Carlo method takes 2.5 hours to solve a problem with a dimension of 10 x 10, while the constructive takes 15 minutes.

3232. LIDA library program for approximation of functions and processing of data. Vasilenko, V. A.; Kovalkov, A. V. and Zyuzin, M. V. ALGOL-BESM-6 version. Novosibirsk, 1981, 39 pages (Preprint No 270, Computer Center, Siberian Branch, USSR Academy of Sciences). Bibl.: 8 titles. The library solves the problems of interpolation and smoothing by splines of any smoothness in a segment, in a region of a type of parallelepiped and in an arbitrary n-dimensional region with chaotically arranged nodes.

3235. Calculation of three-dimensional temperature pattern around a pipeline in frozen ground. Ayzen, A. M.; Aksenov, B. G. and Shokhin, V. F. PROBL. NEFTI I GAZA TYUMENI: NAUCH.-TEKHN. SB. TRUDY. NOVAYA SERIYA/ZAP. SIB. N.-I. GEOL.-RAZVED. NEFT. IN-T, No 49, 1981. "Analysis and Generalization of Expertise on Operations for Oil and Gas in the 10th Five-Year Plan in Western Siberia," pp 62-64. Bibl.: 3 titles. An ALGOL program is described for computing temperature pattern around an oil line with regard to temperature variation along the pipe.
Keywords: methodology, ALGOL, Odra-1204, YeS-1040, temperature pattern, pipelines, oil lines, frozen ground.

3276. ODA Data Processing Management System. Concepts, capabilities. Zabavnikov, V. F. VOPR. RADIOELEKTRON. SER. ASU, No 2, 1980, pp 41-52. Bibl.: 11

titles. The ODA system is oriented to processing data in ASUP [automated production control systems], characterized by a large volume of input and output information and a small number of arithmetic operations and set of operations--logical, moves, conversions of elements and writing of data sets. The system is implemented in the ODA language with use of Russian alphabet letters during design of structures. Operations on elements of records in the system are performed by using user blocks in the ODA, Assembler and PL/1 languages.

3289. Wafer layout algorithm for maximum number of chips. Ivanov, V. V. and Lyangasov, S. I. ELEKTRON. TEKHNIKA, SER. #. MIKROELEKTRON., No 5(89), 1980, pp 50-54. Bibl.: 3 titles.

3301. Copying magnetic tapes from "Elektronika-100/I" format to Unified System format. Bocharov, A. A.; Orlova, T. L. and Popov, M. V. Moscow, 1980, 15 pages (Preprint/USSR Academy of Sciences, Institute of Space Exploration, No 582). Bibl.: 3 titles. A FORTRAN program is described that copies data from amgnetic tapes written in Elektronika-100/I computer format onto magnetic tapes in YeS DOS format.

3306. Facilities for batch debugging in trilevel controllable virtual storage system. Konovalov, N. A.; Kol'tsova, L. I.; Kryukov, V. A. et al. Moscow, 1981, 18 pages. (Preprint/USSR Academy of Sciences. IPM [Institute of Applied Mathematics], No 30). Bibl.: 4 titles. Capabilities of batch debugging of FORTRAN programs in trilevel controllable virtual storage system in terms of input language.

3308. Trilevel controllable virtual storage for BESM-6 computer. Kol'tsova, L. I.; Kryukov, V. A.; Lyubimskiy, E. Z. et al. Moscow, 1980, 21 pages. (Preprint/USSR Academy of Sciences. IPM, No 3). Bibl.: 3 titles. Concepts and procedure for using in FORTRAN programs virtual storage with a size to $4 \cdot 10^6$ words.

3338. Development of "CADCAN-1" software system for computer-aided design of ship cable routings. Berdichevskiy, L. D. and Falin, N. G. MATERIALY PO OBMENU OPYTOM/NTO imeni A. N. Krylov, No 325, 1980. "Problems of Computer-Aided Design of Ship Electrical Equipment," pp 17-23. FORTRAN software system for layout and tightening of main cables introduced during development of ship designs.

3340. Method of computer computation of heat conditions of integrated circuits taking heat removal through leads and package cover into account. Zaks, D. I.; Madera, A. G. and Nagovitsyna, L. F. ELEKTRON. TEKHNIKA. SER. MIKROELEKTRON., No 5(89), 1980, pp 55-59. Bibl.: 8 titles. FORTRAN algorithm for computing heat conditions of integrated circuits based on tri-parallelepiped model, with local sources and drains of heat on two of them. Program computes temperature pattern at 50 points on a chip and on the IC package, taking the effect of leads into account (up to 50). For 12 points on a chip and base of package, with three sources of heat and six leads of package when five versions are computed, computation time is 45 minutes.

3341. Software system for computation and analysis of digital-analog devices of apparatus with use of macromodel of integrated circuits. Zvorykin, L. N. and Mostovoy, D. B. ELEKTRON. TEKHNIKA, SER. MIKROELEKTRON., No 5(89), 1980, pp 36-43. Bibl.: 16 titles.

3343. Criterion and algorithm for selecting optimal method of analog-to-digital conversion. Ilyushin, S. A. VOPR. RADIOELEKTRON. SER. EVT, No 4, 1980, pp 43-53. Bibl.: 2 titles.

3356. Organization of software and information support for computer-aided structural design system. Maleyeva, A. G.; Solov'yev, V. V. and Fionova, L. P. VOPR. RADIOELEKTRON. SER. ASU, No 2, 1980, pp 19-27. Bibl.: 12 titles.

3357. Computation of xenon transient process in heterogeneous reactor (DIXEN program). Malofeyev, V. M. Moscow, 1981, 17 pages (Preprint/ITEF [Institute of Theoretical and Experimental Physics], No 34). Bibl.: 6 titles. FORTRAN program is described for computing transient process occurring under specified conditions of change in capacity, as well as as a consequence of change of fuel channels and other reactor disturbances.

3365. Computer analysis of thermal conditions of hybrid integrated circuits. Petrosyants, K. O. and Ryabov, N. I. ELEKTRON. TEKHNIKA. SER. MIKROELEKTRON., No 5(89), 1980, pp 60-65. Bibl.: 5 titles. FORTRAN program is described for computing stationary heat pattern in substrate of hybrid integrated circuits, created by heat evolution of elements. Three-dimensional equation for heat transfer is reduced to two-dimensional and solved by finite difference method by using the Gauss-Seidel iterative algorithm.

3369. Computer computation of topological parameters of MIS LSI circuits. Romm, G. R. and Shenderovich, Yu. I. ELEKTRON. TEKHNIKA. SER. MIKROELEKTRON., No 5(89), 1980, pp 44-49. Bibl.: 4 titles.

3371. Approximate design of nozzle contour providing minimal loss to friction and dispersion. Sokolov, B. I. Heat processes and properties of working media of flying vehicle engines: MEZHVUZ. SB./Kazan' Aviation Institute, No 3, 1980, pp 77-81. Bibl.: 2 titles. FORTRAN program is described for determining parameters of expanding part of Laval nozzle with free expansion of axisymmetric jet with flat surface of transition through sound velocity. One computation on YeS-1022 computer takes 10-12 seconds.

3372. STRUCTURE software system for X-ray diffraction computations: Description and instructions. Solov'yeva, L. P.; Ovchinnikov, V. Ye.; Ipatova, Ye. N. and Andrianov, V. I. Moscow, 1981, 58 pages (Automation of Research on Atomic Structure of Crystals by Diffraction Methods/USSR Academy of Sciences. Far Eastern Scientific Center. Institute of Tectonics and Geophysics, Institute of Crystallography, No 7). Bibl.: 7 titles. STRUCTURE FORTRAN software system for determining and refining atomic structures of crystals by data of X-ray, electron-diffraction or neutron-diffraction experiments, designed for structures in which the independent atoms are less than or equal to 100, the number of kinds of atoms is less than or equal to 12, and the maximum of independent structural factors is 10,000.

3374. Organization of file of models of atmosphere. WRAPX program for writing to the file. REAPX program for reading from file: Instructions. Sushkevich, T. A. Moscow, 1980, 16 pages.)Preprint/USSR Academy of Sciences. IPM, No 7). Bibl.: 3 titles.

3377. Computation of basic paarmeters of resonant and guiding band structures at high types of waves. Fialkovskiy, A. T.; Mikheyev, A. G. and Tonkikh, N. B. ELEKTRON. TEKHNIKA. SER. 1. ELEKTRON. SVCH, No 12(324), 1980, pp 61-62. Bibl.: 4 titles.

3434. Automated data acquisition and processing system based on "Mikro-Nova" computer. Blokh, M. A.; Kamolova, T. I. and Nechayev, Yu. I. Moscow, 1981, 54 pages. (Preprint/USSR Academy of Sciences. Physics Institute, No 55).
Key words: ang. [English], BASIC, Nova, automated data acquisition and processing systems.

3435. Interactive information retrieval system for decision-making in imperative surgery on organs in abdominal cavity. Arsent'yeva, A. V.; Zimnev, M. M.; Ovsyannikov, A. M. and Khay, G. A. Leningrad, 1981, 47 pages. (Preprint/USSR Academy of Sciences. Leningrad NIVTs , No 9). Bibl.: 12 titles.
Key words: methodology, BASIC, CYBER-172, interactive information retrieval systems, decision-making, surgery, abdominal cavity.

3451. Software for system of automated input of tasks into analog processor of hybrid computer system. Leonenko, V. I. and Rogovtsev, A. A. In book: "Vychislitel'naya tekhnika v sistemakh upravleniya letatel'nymi apparatami: Temat. sb. nauch. tr." [Computer Engineering in Flying Vehicle Control Systems]/ MAI [Moscow Aviation Institute], Moscow, 1981, pp 31-34.

3473. Scientific Research Automation Based on Facilities of System of Small Computers, Moscow, 1980, 89 pages. (TR./IN-T ELEKTRON. UPR. MASHIN, No 83). Bibl. at end of articles. ISSN 0320-3948.
Problems of restoration of image of pictures of spatial sections of controlled object in X-ray tomography with a computer. Development of integrated information systems and priority service systems.
Keywords: methodology, M-4030, M-222, SM-4, IRIS DBMS, ASVT-2 DOS, X-ray tomography, multichannel systems, priority service, input of information, integrated information retrieval systems.

3490. Pumping of space H_2O maser in two-temperature gas. Bolgova, G. T. Scientific Information/USSR Academy of Sciences. Astronomical Council, Latvian SSR Academy of Sciences. Radioastrophysical Observatory, 1981, No 47.
Thematic collection of articles on the problem, "Evolution of Stars and Star Aggregations," pp 9-14. Bibl.: 4 titles.
Refinement of necessary conditions for colliding-colliding pumping of space H_2O maser by the method of digital simulation for refinement of necessary physical conditions.

3495. Package of programs for loading information into a file in an information retrieval system. Vishin, V. V.; Zaytsev, S. A.; Maslennikov, A. M. and Potapov, A. V. ELEKTRON. TEKHNIKA. SER. 1. ELEKTRON. SVCH, No 12(324), 1980, pp 62-64. Bibl.: 6 titles.

3498. Interactive software in DISFORP system. Agafonov, Yu. M. and Durmanenko, Yu. P. VOPR. RADIOELEKTRON. SER. ASU, No 3, 1980, pp 76-79.
Software structure: control program (interactive monitor), language processor, body of apckage, service programs, generation facilities. All package programs are written in Assembler, service programs are in PL/1.

3499. Basic operating system as basis of software for computer complexes of USSR Gosbank automated systems. VOPR. RADIOELEKTRON. SER. EVT, No 13, 1980, pp 14-19. Bibl.: 3 titles.
Described is a set of components of basic OS, developed on basis of YeS OS, and a method of generating OS complexes.

3547. Interactive plan generation system (DISFORP). Agafonov, Yu. M. VOPR. RADIOELEKTRON. SER. ASU, No 3, 1980, pp 71-75.
The DISFORP system has been implemented on a base of the YeS-1033 (YeS-1060) with at least 256K of main storage in the OS-4.0, 4.1 environment. Job run time in optimal mode is 3-5 minutes (for a 200 x 20 matrix). Time of system response to request in update mode is 1-2 minutes, and in query mode (with regard to solving a multicriterial problem) is 5-10 minutes.

3549. Structurization of interaction between decision-maker and computer in solving plan problems of optimization. Bordonosenko, V. A. and Grebel'skiy, S. Z. VOPR. RADIOELEKTRON. SER. ASU, No 3, 1980, pp 80-83. Bibl.: 3 titles.
Method of structurization in developing interaction for decision-maker (LPR-EVM), for solving optimal problems on generating annual nomenclature plans for product production in a sector and enterprises (using a package of programs for mathematical programming) and for making multialternative plan calculations on optimizing the five-year plan for allocation of capital investment in a sector in the DISFORP interactive system. System software includes control procedures, optimization procedures and user programs for the Unified System of Computers. Main storage of at least 256K, interactive program is 64K.

3556. Optimization of capital investment allocation in mid-term planning. Sheynkman, L. E. VOPR. RADIOELEKTRON. SER. ASU, No 3, 1980, pp 87-93. Bibl.: 3 titles. Described is an interactive system for generating an alternative for facilities of capital construction. System response time in generating it on the YeS-1030 in the data update mode is about 0.5-1 minute; computation of analytic tables takes 2-3 minutes; and in the mode of running an optimization job (dimension 250 x 100), 10-15 minutes.

3558. Simulation system for drafting production plans for sector enterprises. Yampol'tsev, G. M. VOPR. RADIOELEKTRON. SER. ASU, No 3, 1980, pp 107-110.

3565. Solving one problem of computer-aided design of printed circuit boards. Belenko, V. V.; Ioseliani, A. N. and Lordkipanidze, L. L. TRUDY/GSSR Academy of Sciences. Institute of Control Systems. No 20:1. Theory and Devices for Automatic Control Systems, pp 126-132. Bibl.: 3 titles.
Methods for layout of micromodules, selection of sequence of connections and their layout on a printed circuit board. For electric circuits containing up to 20 microcircuits, computation time on the M-222 computer is 4-5 minutes.

3585. Programs for processing measurements of linear-polarized galactic radio emission on the "Nairi-K" computer. Arkhangel'skiy, V. G. and Kuznetsova, I. P. Gor'kiy, 1981, 43 pages (Preprint/Gor'kiy Scientific Research Radiophysics Institute, No 145).

3614. Organization of specialized multiservicing on a large computer (DEC-10). Ivanov, Yu. N.; Ivanova, N. S.; Laskovoy, V. N. et al. Serpukhov, 1981, 17 pages. (Preprint/IFVE [institute of High Energy Physics], No 81-27). Bibl.: 13 titles. Principles of design of the COSDES (Comprehensive Operating System for DEvice Servicing) specialized time-sharing system, and of organizing multiservicing of scanning-measuring instruments within the bounds of one job under control of the DEC-10 computer OS. System is used for servicing basic PDP-11/40 computer for design of rpinted circuits boards.

COPYRIGHT: Gosudarstvennaya publichnaya nauchno-tekhnicheskaya biblioteka SSSR (GPNTB SSSR), 1981

8545
CSO: 1863/79

APPLICATIONS

UDC 681.3.014

BASIC PRINCIPLES FOR DEVELOPMENT OF DIGITAL COMPUTER COMPLEXES FOR MULTICHANNEL PROCESSING OF FULL-SCALE TEST DATA

Kiev KIBERNETIKA in Russian No 6, Nov-Dec 81 (manuscript received 8 Jan 81) pp 35-39

[Article by Grigoriy Ivanovich Korniyenko, candidate of engineering science, deputy director of the SKB MMS [Special Design Office for Mathematical Machines and Systems], Institute of Cybernetics, UkSSR Academy of Sciences, Kiev]

[Text] The main purpose in using on-board and field digital computer complexes (TsVK) for multichannel processing of experimental data (ED) in conducting full-scale complex tests of objects of new technology consists in raising the overall effectiveness of these tests based on data from analysis of intermediate stages and of the operational planning and management of their subsequent stages. Using digital computer complexes based on modern electronic digital computers ensures a high degree of informativeness and control of tests in all stages of their performance. Test efficiency is characterized by indicators of reduction in the time for performing them, by the capability of performing comprehensive tests and by obtaining more valid test results, which in the final analysis ensures their considerable national economic effect.

This purpose is achieved by solving a number of basic problems [1] that have a specific nature caused by the performance of full-scale tests of complex dynamic systems and objects. Let us list them.

Problem 1. Provide for efficient acquisition, representation, recording and transmission of experimental data to the digital computer complex input channels from the measuring devices and units connected to transducers or converters of nonelectrical quantities into electrical quantities.

Problem 2. Provide for continuous input, sorting and recording of multichannel experimental data in a broad spectrum of frequencies from fractions of a Hertz to tens of kilohertz.

Problem 3. Provide for the process of visualization of realizations of the input data on raster graphic information displays in real time as the data comes in.

Problem 4. Provide for input and processing of half-tone images coming from video recorders and television cameras.

Problem 5. Define and form basic files of realizations of the experimental data and perform preprocessing of them.

Problem 6. Perform proximate analysis of the data stream in real time.

Problem 7. Perform digital processing of multichannel experimental data by the methods of mathematical statistics and theory of probabilities, spectral-correlation and regression analysis.

Problem 8. Perform general mathematical calculations for evaluation of the basic parameters of the tested object, as well as calculations associated with the planning, optimization and control of full-scale tests.

Problem 9. Provide for the processes of display on system displays and panels and of documentation on alphanumeric and graphic peripherals of the results from processing and analysis of the realizations of the experimental data, object parameters and results of the general mathematical and special calculations.

Problem 10. Provide for storing on magnetic media the results of individual tests and series of tests for subsequent evaluation, comparison and analysis.

Problem 11. Provide for control of data output from digital computer complexes to telemetering channels if there is telemetering equipment for data transmission to earth in the system supporting the performance of full-scale tests.

Problem 12. Provide for efficient operator interaction with digital computer complex for on-line control of its operating modes as well as for development and debugging of new user programs.

Problem 13. Provide for communication and exchange of files of archival data between on-board (field) digital computer complexes and the large general-purpose electronic digital computers used in computing centers (VTs) and automated experimental data processing systems (ASOED) for general and shared use.

This list is not an exhaustive list of the problems that must be solved by the experimenter in performing full-scale tests. However, as experience shows, these problems are the most common, extremely capacious in their information content and complicated for solving by the means of on-board and field digital computer complexes. The solution to the listed problems is conditioned not only by their complexity, but also by the conditions for operating these devices which demand strict limitations on the dimensions, weight, reliability and power consumption for the hardware complex, on the bulk of all components in the methodological, mathematical, program and information provisioning, and on the cost of the digital computer complex facilities as a whole.

Let us discuss the question of developing a digital computer complex capable of solving the problems posed.

To date, there have been two basic approaches to the development of computer complexes based on minicomputers. The first approach: the developer creates for each specific user a unique, narrowly specialized system that solves only the problems of the given user. In developing subsequent systems, using a considerable portion of the preceding system usually does not turn out well. This approach is used, basically, when there is no need of establishing distribution of the complex version. The advantage of this approach is that the complex need not have an excess of functions, thereby keeping development costs to the minimum. The shortcoming in the development of narrowly specialized complexes is their complexity and often even the impossibility of making changes and additions to functioning complexes to extend their capabilities.

The second approach consists in using problem orientation of a computer complex to a class of problems to be solved. The idea of computer problem orientation, proposed by V. M. Glushkov [2, 3], was specifically incorporated at the UkSSR Academy of Sciences Institute of Cybernetics during the development of second-generation computers: the "Dnepr," the "Promin'" and the "MIR." The efficiency of problem orientation has now been recognized by developers in the leading foreign firms too. Problem orientation [4] permits providing the complex being designed with maximum throughput for a given class of problems with minimal equipment cost and creating the most favorable conditions for the user. Before starting development of the complex, the developer performs a comprehensive systems analysis of the sphere of application as a whole, where not only the specific complex, but also other complexes similar to it, will be used. As a result of this analysis (for which, incidentally, numerous problem users are enlisted), the developer outlines the architecture and structure of a problem-oriented complex, the facilities of which permit relatively easy construction of many computer systems for the given sphere of application. This is especially useful in those cases when the problem to be solved by the complex is relevant to dozens and hundreds of applications, though each application has a certain specific nature.

The advantage of the second approach compared to the first is the capability of rapid and efficient adaptation of the developed complex to the problems of a specific application, which permits solving the problem of establishing distribution. Let us note that in practice, a complex designed to solve certain problems usually cannot be used for a new application. Establishing distribution is possible only under the condition of modification of the existing complex.

The shortcoming of the second approach is the high labor input both for performing systems analysis of the applications and for implementing the problem-oriented complex. Such complexes have some excess of functions in individual applications which can be justified only in the case of extensive incorporation of the complexes by various users.

Thus, the second approach permits adapting the problem-oriented complex to a specific application and developing user complexes based on it. Development of a user complex based on the development of problem-oriented complexes seems to be the most promising method of making use of computer hardware for processing of full-scale test data.

Development of problem-oriented digital computer complexes methodologically combines two groups of aspects (fig. 1): of the user and the developer.

In the operation of a digiatl computer complex, a technique is used that is based on the use of certain procedures and rules for interaction between the experimenter and the complex hardware. The main user facilities for controlling the process of data processing are problem-oriented high-level interactive languages and various functional keyboards. A feature for digital computer complexes is that the experimenter operates on the complex. This is necessary because the experimenter, based on his own experience and intuition, must analyze the results of the proximate processing of data, make conclusions on the experiment performed and plan the next one. Moreover, a digital computer complex is usually operated under field conditions in immediate proximity to the object of the tests or on board the object, where access to attendants is sometimes restricted.

In accordance with what has been presented, let us formulate the first principle for development of a digital computer complex: A digital computer complex must be oriented to the problem user, which is the experimenter or tester.

The second group of aspects is associated with the developer who is faced with the problem of developing a problem-oriented digital computer complex that meets the requirements of all potential users with the minimum possible cost for the facilities and time for development.

Key:
1. Level 1
2. Level 2
3. Level 3
4. Problem user
5. Developer
6. Main programs for complex
7. Application program packages
8. Programming system
9. Operating systems
10. Hardware
11. Structural execution

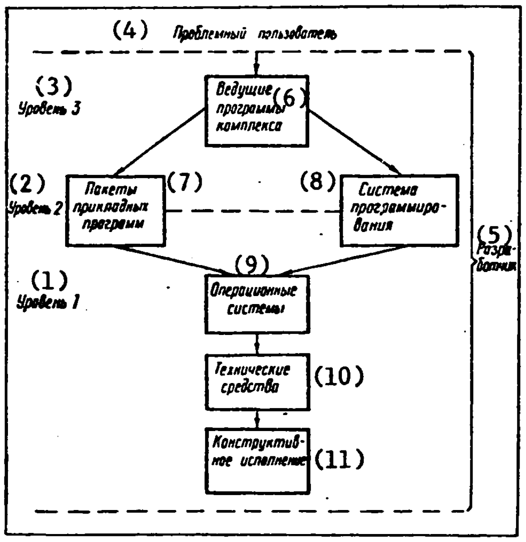

(4) Проблемный пользователь
(3) Уровень 3
Ведущие программы комплекса (6)
(2) Уровень 2
Пакеты прикладных программ (7)
(8) Система программирования
(5) Разработчик
(9) Операционные системы
(1) Уровень 1
Технические средства (10)
Конструктивное исполнение (11)

Fig. 1.

Let us consider the components of a digital computer complex (fig. 1).

1. Hardware: processors, main and passive storage units, input/output channels, peripheral controllers, nonstandard peripherals, power supplies, and extensively applied peripherals: magnetic disks, magnetic tapes, video terminals, teletypes, perforators, etc., that provide the necessary speed and high throughput.

2. Operating systems, which are the interface between application software and hardware. They must make full use of the capabilities of both the individual hardware units and their joint operations in the mode of overlapping in time (task management, job control, storage control, peripheral control, file management).

3. Programming systems: user language translators (Assembler, FORTRAN IV), text editors, linkage editors and debuggers. They are the facilities for automating the development of large complexes of programs.

4. Packages of application programs, oriented to the application problems that perform various computations (linear algebra, mathematical statistics and spectral analysis).

5. The main programs which implement all the functions specified by users, based on the preceding software levels (problem-oriented language translator, modes number 1 and 2).

6. Structural execution is the element base, printed circuit boards and connectors, cable connectors, and other units.

Selecting the hardware is the most fundamental aspect in the process of developing digital computer complexes. For digital computer complexes for various applications, the most widespread approach is to select the SM3, SM4 or SM1, SM2 series general-purpose control computer compelxes (UVK). In this case, not only the hardware is determined, but also the operating systems, programming systems and in part, the application program packages. The developer's task is to connect to the computer complex the peripherals not included in the equipment set supplied by the manufacturing plant and to write the packages of application and main programs for the problem-oriented digital computer complex.

However, attempts to use series control computer complexes for processing the data of full-scale tests have not been crowned with success. This is due primarily to the special requirements imposed on the digital computer complex hardware structural execution under field and on-board conditions of operation: resistance to vibration both during operation and while being transported, resistance to the effect of the environment, higher than usual humidity and dust content, the capability of operating from unstable voltage sources, the necessity of staying within the dimensions of individual units, software functioning reliability and others.

All this allows formulating the second principle that should be followed in developing a digital computer complex: newly developed hardware units and their structural execution must meet the severe conditions of full-scale tests and experiments.

In manufacturing the hardware, the developer is faced with the alternative of following the architecture of some series control computer complex or developing a new one.

Following known architecture, at first glance, has advantages over developing a new one. Thus, since the computer complex is software-compatible with the prototype, there is no need of developing an operating system, a programming system and application program packages. Moreover, there is the capability of using the prototype peripheral controllers since the input/output channel interface is identical. However, as the experience of domestic industry indicates, achieving full software compatibility with a prototype is a very labor-intensive task that requires large time inputs. The highest compatibility can be achieved with the operating system programs since here an effect is produced by the time relations of the control signals at the level of the circuits of the individual units and assemblies.

Let us assume that we have succeeded in building hardware software-compatible with series control computer complexes and meeting the requirements of structural execution. Then inevitably the question arises of how much of the architecture of the hardware and the operating system for the series control computer complex is adaptable to the specific nature of the problems of full-scale test data processing. Analysis data show that in solving the problems, devices should be developed for communication between the numerous peripherals nonstandard to series control computer complexes (on-board magnetic tape and other recorders, telemetric magnetic tape recorders, multichannel high-frequency ATsP [analog-to-digital converters] and others) and the input/output channels in the complex. To preclude data loss in switching input buffers, the operating logic of the channel for direct access to storage must differ from that of the channel for the series control computer complexes [5].

Here is what is required to solve problems 3 and 4: First, develop raster-type half-tone displays and graphic plotters, controllers for them, and also controllers for the video magnetic tape recorder and television camera. Second, special methods are required to process images by using the mode of dual-processor operation and to output moving images to display screens [6]. The series control computer complexes do not have these facilities.

To solve problem 6, a high processor speed has to be provided (on the order of 1.5 to 2 million operations per second) and this cannot be achieved with the series control computer complexes [7, 8].

To solve problems 7 and 8, we need a high-speed floating-point processor [9] and we have to develop special hardware methods for controlling a main storage of limited size [10-12].

The most significant limitation in building digital computer complexes is that under field and on-board conditions, the complex often must operate effectively even with minimal configurations of hardware, for example without a magnetic disk. With that, it must provide for the multiprogramming mode, the capability of the experimenter to actively intervene in the computing process by using job control language, and the operation of a complicated programming system in developing program complexes. Consequently, the conclusion can be drawn that the question of the efficient use of the main storage in the digital computer complex is of paramount importance [13].

Operating series control computer complexes in the multiprogramming mode with real-time problems is usually possible only when the disk operating systems are used. Operating systems for minimal configurations have insufficient capabilities for implementation of digital computer complexes.

Summing up what has been presented, let us formulate the third principle for development of problem-oriented digital computer complexes for multichannel processing of full-scale test data: The architecture and structure of the hardware and the operating system in the digital computer complex must be adaptable to the full-scale test data processing problems. According to this principle, development of an efficient digital computer complex is possible only when new architectural and structural solutions are used in developing the hardware and all software components. This approach is extremely labor-intensive since in addition to the performance of functions specific to the applications, the digital computer complex must have the characteristics and basic capabilities of general-purpose control computer complexes. Development of digital computer complex software that is compact and sufficiently powerful in its capabilities presents the greatest difficulty.

The relation between functions performed by hardware and those by software is of great importance in achieving high efficiency in the digital computer complex. The most important fragments of components should be identified in the systems analysis for the design of the digital computer complex and the hardware should be oriented to implementing them. Thus, with the input of multichannel information, the problem of unpacking the frames from external media into linear files of values for each of the specified channels arises. Using special computer instructions permits solving this problem successfully when the data comes in at a high rate [14]. Frame structure regularity disturbances occurring during input due to a malfunction are eliminated by devices for communication with a multichannel recorder.

To increase speed, a digital computer complex must have the capability of writing and executing microprograms that differ from the basic set of microprograms and processor instructions. To this end, there have to be facilities for expanding microprogram control storage both through adding new permanent storage units and through using main storage for microinstructions. Using microprogramming in problem programs and operating system modules permits raising the program execution rate three- to sixfold compared to the write rate by using the basic processor instructions.

Thus, the fourth principle for developing problem-oriented digital computer complexes can be formulated: The relationship between hardware and software components in digital computer complexes under development has to be selected so that the maximum speed and throughput during processing of input data streams can be provided at minimum total cost.

The principles and considerations presented above have served as the basis in developing a number of digital computer complexes for multichannel processing of data from full-scale tests of intricate specimens of new technology, among which special mention should be made of the "Ekspress" systems [15], the "Etalon" problem-oriented computer [16], the "Pirs" system [17] and some others. The experience of the development and industrial operation of such problem-oriented digital computer complexes has shown the high effectiveness and viability of the principles and propositions used as the basis for their design which lent the designed complexes a number of distinctive features ensuring their high technical and economic parameters.

BIBLIOGRAPHY

1. Korniyenko, G. I., "Problems Solved in Automated Experimental Data Processing Systems in Performing Experiments and Tests on Complex Objects under Full-Scale Conditions," in "Sistemy ekspress-obrabotki dannykh v real'nom masshtabe vremeni" [Systems for Proximate Processing of Data in Real Time], Kiev, 1979, pp 3-12.

2. Glushkov, V. M., "Vvedeniye v ASU" [Introduction to Automated Control Systems], Kiev, 1972, 310 pages.

3. GLushkov, V. M. et al., "Some Trends in Development of Digital Computer Software Structures," UPRAVLYAYUSHCHIYE SISTEMY I MASHINY, No 1, 1972, pp 79-85.

4. Barsuk, Ya. I.; Korniyenko, G. I.; Sergiyenko, I. V. and Tesler, G. S., "Problem Orientation of Minicomputers," UPRAVLYAYUSHCHIYE SISTEMY I MASHINY, No 6, 1973, pp 45-49.

5. Korniyenko, G. I.; Dianov, M. I. and Dianov, V. I., "Method of Organizing Process of Continuous Input in Real-Time Systems Based on Minicomputers," KIBERNETIKA, No 4, 1980, pp 122-125.

6. Korniyenko, G. I.: Dianov, M. I. and Dianov, V. I., "Organizing the Process of Visualization of Results in Systems for Proximate Processing of Full-Scale Test Data," UPRAVLYAYUSHCHIYE SISTEMY I MASHINY, No 1, 1981, pp 108-112.

7. Korniyenko, G. I., "Organizing Information Processing in the Proximate Analysis Mode in Tests of Complex Technical Systems," in "Proyektirovaniye i vnedreniye novykh sredstv vychislitel'noy tekhniki" [Design and Introduction of New Computer Hardware], Kiev, 1978, pp 11-17.

8. Korniyenko, G. I.; Diadov, M. I. and Diadov, V. I., "Principles of Design of Multichannel Digital Proximate Analyzers," KIBERNETIKA, No 6, 1980, pp 63-67.

9. Korniyenko, G. I.; Diadov, V. I. and Diadov, M. I., "Digital Analysis of Signals in Systems for Processing Experimental Data," UPRAVLYAYUSHCHIYE SISTEMY I MASHINY, No 6, 1980, pp 100-104.

10. Korniyenko, G. I.; Diadov, V. I. and Diadov, M. I., "Data Storage Control Method for Computers with Limited Size of Main Storage," UPRAVLYAYUSHCHIYE SISTEMY I MASHINY, No 4, pp 29-31. [no year given]

11. Dianov, V. I., "Problem of Efficient Use of Minicomputer Memory," UPRAVLYAYUSHCHIYE SISTEMY I MASHINY, No 5, 1978, pp 23-25.

12. Korniyenko, G. I. and Sheverda, O. N., "Features of Dynamic Program Maintenance in Data Proximate Processing Systems," KIBERNETIKA, No 5, 1980, pp 86-88.

13. Korniyenko, G. I. and Sheverda, O. N., "Resource Control in Full-Scale Experiment Data Processing Systems," KIBERNETIKA, No 5, 1978, pp 41-45.

14. Dianov, M. I. and Timchenko, G. I., "Organizing Real-Time Experimental Data Processing Systems," in "Sistemy ekspress-obrabotki dannykh v real'nom masshtabe vremeni," Kiev, 1979, pp 19-23.

15. Korniyenko, G. I., "System for Proximate Analysis of Experimental Data for Full-Scale Tests of Complex Objects (Ekspress-1), UPRAVLYAYUSHCHIYE SISTEMY I MASHINY, No 6, 1978, pp 125-128.

16. Korniyenko, G. I., "'Etalon' Problem-Oriented Digital Computer for Real-Time Systems," UPRAVLYAYUSHCHIYE SISTEMY I MASHINY, No 1, 1979, pp 104-106.

17. Korniyenko, G. I., "Digital Computer Complex for Multichannel Processing of Experimental Data in Real Time ("Pirs" TsVK)," UPRAVLYAYUSHCHIYE SISTEMY I MASHINY, No 6, 1979, pp 130-136.

8545
CSO: 1863/91

UDC 62-52:681.3.06.44

PROBLEM-ORIENTED COMPLEX FOR PROCESSING GENERAL SHIP INFORMATION BASED ON MINICOMPUTER (TASKS AND ALGORITHMS)

Kiev PROBLEMNO-ORIYENTIROVANNYY KOMPLEKS OBRABOTKI OBSHCHESUDOVOY INFORMATSII NA BAZE MINI-EVM (ZADACHI I ALGORITMY) in Russian 1980 (signed to press 31 Dec 80) pp 2-8, 40-42

[Annotation, table of contents, excerpts, conclusion and bibliography from book "Problem-Oriented Complex for Processing General Ship Information Based on a Minicomputer (Tasks and Algorithms)" by Aleksandr Aleksandrovich Bakayev, Vladislav Sergeyevich Petukhov, Valeriy Luk'yanovich Revenko, Al'bert Mikhaylovich Stafeyev and Nikolay Nikolayevich Tsymbal, Institute of Cybernetics, UkSSR Academy of Sciences, 300 copies, 44 pages (Preprint 80-70)]

[Excerpts] The tasks and algorithms for solving the problems of increasing the automation of the basic services of modern automobile and passenger ships are described. These tasks and suggested solutions are novel and superior to similar foreign systems in their characteristics. The tasks of accounting and registration of passengers, compilation of lists and muster rolls, and payroll computations are described comprehensively.

All tasks are combined into three independent problem-oriented complexes based on a common problem task and uniform technology for processing operational, normative and reference information.

Contents

"Belorussia" type passenger and car ferries are now equipped with the modern highly efficient PDP-8E minicomputers.

The main goals in automating the management of the vessels and the individual services of a vessel are to reduce costs for routine manual labor, improve navigation conditions and increase the economic effectiveness of vessel operation. To reach these goals, a high degree of automation is required to process operational and reporting general vessel information.

In this connection, studies were made to develop algorithms and programs to implement them on the minicomputers.

From here on, what we mean by the problem-oriented complex for processing general vessel information is the set of hardware and software modules oriented to the common problem task and unified technology of data processing.

The common problem task is determined on the basis of the goals and tasks of each of the vessel services, which are characterized by uniformity of tasks and data processing technology. The main services of passenger and car ferries include passenger service, navigation service, chief mechanic service, hull and mechanisms maintenance service and the restaurant and accounting service.

Navigation service has at its disposal facilities for determining vessel location, dead reckoning and anticollision measures. The hull and mechanisms maintenance service provides for the performance of all freight operations and determination of the parameters and characteristics dependent on vessel load (calculation of displacement, longitudinal trimming, etc.)

The restaurant and accounting service computes wages for crews on domestic and foreign trips and performs accounting and monitoring of the utilization of material resources, including food stores.

The vessel passenger service is the key on "Belorussia" type passenger and car ferries; it is expected to provide a high level of service to passengers and tourists directly on board the vessel. The passenger service standard is shaped largely by the interrelations and harmony of the operation of all vessel services, and by the stability, reliability and harmony of the functioning of the large and small systems and mechanisms on a vessel.

Development problems were studied within the framework of the "Morpasflot" ASU and a problem-oriented complex for processing general vessel information was implemented on the base of the minicomputer on board. This complex functions on the "Belorussia" type vessels.

In future, this complex will become the lower level of a "Morpasflot" ASU subsystem will will allow solving the entire aggregate of tasks on passenger service from ticket sales to debarkation.

The first, or top, level of the "Morpasflot" ASU is located on shore at the computer center for the vessel line.

The problem-oriented complex for processing of general vessel information includes: the PDP-8E minicomputer, the ASR-33 operator consoles, detached terminals and the DRI firm's magentic tape storage unit.

1. The DISP1/OTKHOD Problem-Oriented Complex

The problem task for processing general vessel information and automating the basic operative functions of the passenger service on the vessel is the DISP1/OTKHOD complex.

Development of the DISP1/OTKHOD tasks is aimed at producing indicators for operational reporting that describe the composition and number of passengers, the status of the vessel on a trip and at increasing the economic effectiveness of operation of the vessels through fuller utilization of vessel passenger capacity.

The composition of indicators included in the DISP1/OTKHOD complex is prescribed by the USSR Ministry of the Maritime Fleet.

As a result of performing calculations, data sets are generated to expeditiously inform the interested services on the availability of open and occupied spaces by cabin categories and on the movement of passengers to vessel ports of call, and a file of immediate information on vessel operation is compiled in DISP1/OTKHOD form.

The principle of the systems approach is used in solving the problems of the DISP1/OTKHOD complex. The tasks making up the complex are solved on the basis of unified entry, immediate and normative-reference information.

The DISP1/OTKHOD complex includes the following tasks: registration of passengers, keeping track of movement of passengers to vessel ports of call and compilation of the DISP1/OTKHOD [dispatcher 1/departure].

Passengers are registered when they board the vessel upon presentation of a boarding ticket. The following data is used as entry information: passenger type, deck number, cabin number, berth.

The information obtained from passenger registration is used to solve the problem of raising the vessel passenger capacity utilization factor, to improve passenger service on the vessel and in solving subsequent complex problems.

Passenger traffic accounting is performed for each vessel port of call by these indicators: how many passengers got off, how many got on and total number of passengers on board the vessel.

In addition to passenger data, the task includes recording data on the traffic and movement of cars and freight transported by the vessel by ports of call.

Used as entry information is data on the number of passengers and cars and amount of freight on board the vessel.

Based on the information on passengers and freight and deck log data, an operational summary report is generated that describes the status of the vessel by its utilization and financial status.

The documents and information obtained as a result of solving the problems in the DISP1/OTKHOD complex are used for passenger cruise vessels on foreign and coastal trips.

The tasks in the DISP1/OTKHOD complex are run at each port of call upon completion of a trip and upon request.

Reports in DISP1/OTKHOD form are made to the line office and cover the past 24 hours (from 1800 to 1800); on the first day of the month, vessels report information on operations from 0000 to 1800 on the current day, and on the last day of the month, the reports cover the period from 1800 on the past day to 2400 on the current.

Information on the operations for the 24-hour period are recorded and sent to the line office as of 1800 Moscow time.

When a vessel calls at several ports in the reporting period, a DISP1/OTKHOD report is sent for each port separately.

The information recorded as a result of performing calculations is used to solve the complex of tasks, "Information on Operation of Vessels," based on which the "Immediate Accounting" subsystem of the "Maritime Fleet" Sector Management Information System functions.

All files used in solving this complex of tasks have fixed-length records and contain both digital and alphanumeric information.

The following documents are used to solve this complex of tasks: boarding ticket, freight documents, information on revenues, deck log and trip ticket.

The following indicators are recorded: port code, port name, arrival date, arrival time, departure date, departure time, passengers boarded by ports of call, cars boarded by ports of call, ordinal number of port of call (call for taking on freight), ordinal number of port where unloaded, freight code, revenues from transporting freight in Soviet rubles, revenues from hauling freight in foreign currency, type of passenger, deck number, cabin number, berth number in cabin.

From this data, the following files are generated: trip start file (MOR), passenger file (MP), car file (MA), freight file (MG), revenue file (MD), plan-chart file (MPK).

Normative-reference information from the M2 file, "Classifier of Sea Ports," is used.

The composition and characteristics of the requisites for the M2 file are given in table 1.1.

Table 1.1. Composition and Characteristics of M2 File Requisites

File name: Classifier of Sea Ports File code: M2

Requisite codes by which this file is formed:

		Field value	
Field name	Field symbol	Length	Variation range
Port code	P	9(6)	up to 30 ports per trip
Port name	IP	A(12)	

The MOR trip start file is filled in at the start of each trip and contains information in the form of statements

$$P/IP/date_{arr}/time_{arr}/date_{dep}/time_{dep}/.$$

The file may contain up to 30 statements. The statements are arranged in vessel port of call order, beginning with the first port of call and ending with the last port of arrival.

The composition and characteristics of the MOR file requisites are given in table 1.2.

Table 1.2. Composition and Characteristics of MOR File Requisites

File name: Trip Start File File Code: MOR

Requisite codes by which this file is formed:

		Field Value	
Field name	Field symbol	Length	Variation range
Port code	P	9(6)	up to 30 ports per trip
Port name	IP	A(12)	
Arrival date	$date_{arr}$	9(4)	
Arrival time	$time_{arr}$	9(4)	
Departure date	$date_{dep}$	9(4)	
Departure time	$time_{dep}$	9(4)	

Conclusion

1. Passenger sea transportation is a complex technical and economic system distinguished by its integrity, complexity of roganization, continuous evolution and the capability of mechanization and automation of the processes occurring in it.

There is a need for developing flexible and dynamic structures capable of supporting the rapid making of resource-balance decisions that provide for its operation.

2. The "Morpasflot" ASU subsystem is a two-level complex designed for automated preparation for, making and implementation of organizational and technical decisions that provide for optimal operation of the fleet and performance of the operational processes for serving passengers.

3. The problem-oriented complexes for processing general vessel information, developed on the base fo the PDP-8E minicomputer installed on "Belorussia" type passenger and car vessels, are substantially reducing costs of routine maunal labor and promoting qualitative improvement of passenger service.

4. The task complexes that have been developed can be operated on the base of the domestically produced "Saratov-2" minicomputer at negligible cost.

BIBLIOGRAPHY

1. Petukhov, V. S.; Bakayev, A. A.; Khayrnasov, M. Kh. and Sklyarov, A. V., "Modeli ASU morskogo transporta" [Models of Sea Transportation Management Information Systems], Moscow, Transport, 1976, 223 pages.

2. Brusentsov, N. P., "Minikomp'yutery" [Minicomputers], Moscow, Nauka, 1979, 269 pages.

3. Kutsenko, A. V.; Polos'yants, B. A. and Stupin, Yu. V., "Mini-EVM v eksperimental'noy fizike" [Minicomputers in Experimental Physics], Moscow, Atomizdate, 1975, 283 pages.

4. Bakayev, L. A.; Petukhov, V. S.; Revenko, V. L. and Revin, V. A., "Zadachi i algoritmy avtomatizirovannoy podsistemy registratsii i ucheta passazhirov na baze bortovoy mini-EVM" [Tasks and Algorithms for an Automated Passenger Registration and Accounting Subsystem Based on an On-Board Minicomputer], Kiev, Institute of Cybernetics, UkSSR Academy of Sciences, 1979, pp 23-40 (Preprint 79-32).

8545
CSO: 1863/92

UDC 681.32:551.46:53.08

INTERACTION OF COMPUTING PROCESSES IN AUTOMATED RESEARCH VESSELS

Riga AVTOMATIKA I VYCHISLITEL'NAYA TEKHNIKA in Russian No 1, Jan-Feb 82 (manuscript received 30 Mar 81) pp 12-21

[Article by O. S. Zudin, S. N. Domaratskiy and L. Lindfors: "Organizing the Interaction of Computing Processes in Variable-Structure Systems for Automating Scientific Research"]

[Text] The most important indicators of the efficiency of multipurpose scientific research vessels for studying the world ocean are the volume and quantity of data collected and processed during a cruise and expenditures for reorganizing the program of the cruise and essential downtime between cruises. Solving the problems of data collection and processing is inconceivable today without automation of research, without installing measurement and computer hardware combined into a unified system for automation of scientific research (SANI) on the ship. Automation of research on multipurpose scientific research vessels is made more complex by the unique character of particular experiments, the necessity of collecting data from large areas over long periods of time and in a broad range of studies, the existence of a large number of measured parameters and measurement techniques, the diversity of algorithms for recording and processing data using deck, towed, and sounding equipment, the necessity of rapidly restructuring the automation system to conform to the requirements of new experiments, heightened requirements for system reliability, and the like. This task is made even more complex by the fact that the research contingent is not constant from one cruise to another, while the data recorded both during the cruise and after its completion must be accessible and understandable to a broad range of specialists in different fields of science who did not participate in preparation for and actual conduct of the particular experiment.

Meeting these sometimes conflicting requirements makes it necessary to design a SANI with flexible and quickly reorganizable structure, easy to master, simple to use, with graphic representation of intermediate and final results.

Work [1] reviewed the basic principles of constructing SANI's for multipurpose research ships using the example of the integrated system for automation of scientific research of the research vessel Akademik Mstislav Keldysh, which was built for the USSR Academy of Sciences at the Hollming AO shipyard in Finland.

This system consolidates measurement and computer instruments and hardware, devices for sounding the body of water, and devices to measure different parameters of the environment, with subsystems for data recording and processing, into a single integrated complex. The final product of recording is files in standard format with data in the form of values of the measured and computed parameters. The information contained in these files may be subjected to further directed processing in the ship's processing subsystem and at on-shore computer centers. The values of the recorded parameters are fed to the system both directly from various sensors, measuring instruments, and subject subsystems and manually, using a keyboard, by operators of laboratory computers based on the results of visual observations of the state of the environment or laboratory analyses of samples taken.

Let us dwell in greater detail on the methods and procedures which are the basis for formulating the software of the SANI and make it possible to restructure the system for new experiments in a fairly flexible way with minimum expenditures. The basis for this is including the appropriate resources in the recording subsystem: system tables, means of generating, maintaining, and updating them, and certain procedures for organizing interaction of computing processes taking place at different levels of the system hierarchy. In this case, according to [1], we will bear in mind that the state of the computing processes in the system at any moment in time is determined by the program, the current steps of this program, and the state of program variables and input-output units.

One of the most important resources of the system is the Table of Parameter Descriptions (TOP), which makes it possible to break the full set of parameters recorded in the SANI down into subsets assigned to definite laboratories or subject subsystems and to describe each element of them. The TOP is generated or updated during adjustment of the system for a certain group of experiments. The recording subsystem insures recording of the values of those, and only those, parameters whose descriptions are contained in the TOP at the current moment. The elementary TOP entry, describing one parameter, contains mandatory and optional fields. The mandatory field includes, as a minimum, the name of the parameter and units of measure of its values, the type of sensor or measuring instrument, the format and type of values obtained, and a description of its subset affiliation and reference to use in other resources. The subset affiliation is described by a composite code whose first two characters are a mnemonic abbreviation of the subset name; the next four digits indicate the ordinal number of the parameters in the subset, if other methods of ordering elements are not used. The optional fields appear in TOP's when they are oriented to standard formats for records of finite files with data. Table 1 below shows the format of the elementary TOP entry and an example of recording one of the parameters recorded by the automatic weather station of a synoptic weather laboratory compiling a finite file in a format close to that of work [2].

As a rule the SANI is restructured for new experiments by connecting equipment to it to measure new parameters, which also requires a certain restructuring of programs at all levels of the hierarchy. To minimize expenditures for reconfiguration the software of the Integrated System of the research ship Akademik Mstislav Keldysh is built on the modular principle. The functions of primary processing, transmission, sorting, and storage of data are distributed among the computing processes, which are defined by particular program modules so that the required changes touch the minimum number of these modules.

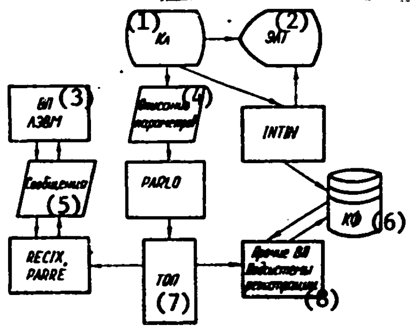


Figure 1. Structure of Interaction of Computing Processes During Generation and Updating of TOP and Formation of Files.

Key: (1)	Keyboard:	(5)	Messages;
(2)	Cathode Ray Tube (Display);	(6)	Finite File;
(3)	Computing Process of Laboratory Computer;	(7)	TOP;
(4)	Parameter Description;	(8)	Other Computing Processes of Recording Subsystems.

Figure 1 shows the structure of interaction of computing processes during generation (updating) of the TOP, transmitting information on a parameter from the recording subsystem to the laboratory computer at the request of an operator, and formulating new files. The computing process defined by the PARLO program receives entries on new or changeable parameters from the terminal in the format shown in Table 1 and structures the TOP in the computer memory of the recording suosystem. The volume of the TOP is defined by the program of the cruise and does not exceed 300-500 entries in the system under description. The computing process defined by the PARRE program, when requested by the laboratory computer, finds the necessary description by its composite code and sends the laboratory computer a message which contains information on the queried parameter.

Table 1.

No.	Content of Field	Primary Field	Length, bytes	Displacement from Start, bytes
1	Code of parameter by classifier	No	4	0
2	Name of parameter	Yes	15	4
3	Code of unit of measure	No	3	19
4	Name of unit of measure	Yes	11	22
5	Scale 1	No	8	35
6	Scale 2	No	8	41
7	Attribute 1	Yes	7	49
8	Definition of attribute 1	Yes	11	56
9	Attribute 2	No	7	67

[Table continued, next page]

[Table 1 continued]

No.	Continent of Field	Primary Field	Length, bytes	Displacement from Start, bytes
10	Definition of attribute 2	No	11	74
11	Type of sensor	Yes	10	85
12	Code of measurement technique by classifier	No	2	95
			1	97
13	Number of characters after comma	No	1	97
14	Total length of value in characters	Yes	2	98
15	Format identifier	Yes	1	100
16	TSS index	Yes	4	101
17	Composite code of parameter in SANI	Yes	6	105
18	Reserve	No	3	111
19	Word description of measurement technique	No	29	114

Example of recording the parameter of air temperature measured with a precision down to tenths of a degree using a sensor of an automatic weather station installed at a height of 20.5 meters from the deck on the starboard side:

000000	GATE CODE:	5000
000001	PARAMETER NAME:	AIR TEMP SB
000002	UNIT CODE:	500
000003	UNIT NAME:	DEG C
000004	SCALE 1:	1.000000
000005	SCALE 2:	0.000000
000006	ATTRIBUTE 1:	20.5
000007	DEFINITION 1:	HEIGHT, M
000008	ATTRIBUTE 2:	-
000009	DEFINITION 2:	-
000010	SENSOR:	DTS-11
000011	METHOD CODE:	80
000012	LENGTH OF DEC.:	1
000013	TOTAL LENGTH:	06
000014	VALUE TYPE:	F
000015	STATE TABLE IND.:	0059
000016	COMPOSITE CODE:	MS0021
000017	RESERVED:	
000018	METHOD DESCR.:	PLATINUM RESIST. OF WEATHER ST

The programs which provide for transmission of data among subsystems of different levels are singled out in a separate group. The formats of the fields of data of transmitted messages have been standardized so that the modules of the communications programs can remain constant when the subsystem is adjusted for different experiments. The protocols selected for the communications lines between laboratory computers and the recording subsystems are subsets of well-known standards ISO-1745 and X.25, while the communications lines for subject subsystems of the automatic weather station type use simplified procedures that resemble the BSC [3].

Because the protocols do not affect the essential features of the messages themselves, we will describe here some principles of formulating them that make it much simpler both as a problem of reconfiguration of software when the system is adjusted for new experiments and a problem of retrieving recorded data. Whereas the water sounding experiment cited in [1] changed the set of sensors and adjusted the work of the submarine and deck units, with traditional procedures that depersonalize data and give the right to identify it to the computing process defined by the recording program it is necessary to make changes in practically all programs related to the experiment. To avoid this, the system being described realizes the principle of data self-identification [4], where all data being received are given tags that indicate their type, properties, time and place of recording, and so on.

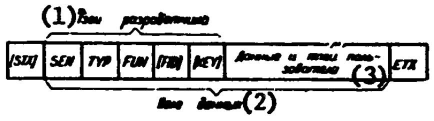
Tagging data in the SANI's of multipurpose scientific research vessels should be done beginning from the lowest levels of the hierarchy. At each level, as the result of the operation of the corresponding computing process, the data that are transmitted to the higher level are formulated. Tags are added to these data. Their principal purpose is to organize the interaction of computing processes taking place at the next hierarchical level of the system. We will identify two levels of tags conforming to their purposes. First-level tags (developer tags) are designated for controlling interaction of computing processes defined by known program modules in the process of recording data. Second-level tags (user tags) are designated to give greater informational value to data during subsequent processing and for control of the processes of sorting and retrieving needed elements of a set of recorded data. The programs of these processes are determined by the user and may be unknown in the stage of system development.

The number and composition of developer tags transmitted with the data from certain computing processes to others at different levels of the hierarchy depends on system structure. For instruments and sensors only the simplest developer tags are needed, such as the tags for type and quality of data, for example the code of the RR regime and the overflow sign P in the message PRRZDDDDDK for laboratory pH-millivolt meters [5]. Other tags of this type include the code of the quantity being measured or the number of the sensor in a set where there is a large number of similar sensors. We consider information on physical overloading of an instrument and on a quantity going outside the boundaries of the measurement range to be quality tags that describe the degree of suitability of the data for processing; in the case of more complex microprocessing instruments such tags may be data on errors that are recognized. When a new instrument is connected to the laboratory level in a laboratory computer the computing process that receives incoming data must be determined. This computing process may be expanded into a series of simpler processes, each of which corresponds to one program module and performs one particular operation. For example, for the laboratory pH-millivolt meter data may be dumped in a dump file based on the value of quality tag S by computing process No 1 that makes a quality check, while on the basis of the value of an MV type tag the data are transferred to computing process No 2, which receives the data measured in volts (or millivolts). When the processes are expanded by the values of user tags the new computing process may be defined so that it includes primarily program modules that already exist, with a small number of new (or modified) modules. This greatly reduces the time required for reconfiguration of software at the level of the laboratory computer. We should note that at this level in the stage of preliminary data processing user tags with values of recording time are added to them. In our opinion, this is essential for the SANI's of multipurpose scientific research vessels.

In order to speed up the reconfiguration of software in the messages being transmitted to the recording subsystem, there must be the following developer tags at a minimum: the sender tag, which indicates precisely which one of the similar computing processes originated the message that has arrived; the type tag, which describes the affiliation of the process to a certain group of computing processes of the recording subsystems; and, the tag for the message function in the subsystem. In addition, for data from prolonged measurements there must be a tag to indicate the name of the file for recording, while for data from one-time measurements a key parameter tag is needed to show the subset of recorded parameters with which the particular message should be associated. Figure 2 shows the structure of a field of data messages transmitted to the recording subsystem of the Integrated System of the scientific research ship Akademik Mstislav Keldysh, while Table 2 indicates the values of the developer tags for all types of system messages. The data in the messages are encoded with characters of the standard seven-bit code ISO-646 (Soviet analog KOI-7 GOST 13502-74). When the composition of the available subsystems is changed this set of developer tags makes it possible to limit changes in the recording subsystem to simply editing the TOP, and when a new subsystem is connected to the SANI the only thing necessary is to develop one more program for receiving messages, without significant changes in existing software.

The most frequently used user tags are parameters such as the date, time, and place (latitude and longitude) of the experiment, the identifier of the researcher, and parameters that describe the environment: direction and velocity of wind, air temperature and humidity, sea state, direct and reflected solar radiation, and the like. Second-level tagging requires inclusion of another resource in the structure of the recording subsystem: internal memory, a supervisor Table of the State of the Ship, system, and environment (TSS). This resource is structured and modified concurrently with the TOP and contains the current values of parameters that can potentially be used as user tags. Figure 3 below shows the structure of the TSS. The names of the parameters whose values form the TSS are defined in the TOP by non-empty values of the TSS index (see Table 1 above). All elements of the table are put in order according to the value of the index. Because the format of parameter values is not known in advance and may change many times in the TOP during the life of the system, a two-level structure is used for the TSS that makes it possible to modify its format and dimensions on an operational basis.

Figure 2. Format of the Field of Data Messages Transmitted to the Recording Subsystem.



Key: (1) Developer Tag;
(2) Data Field;
(3) Data and User Tags;
(STX) and (ETX) Control Characters of ISO646 Code;
(SEN) Tag of Message Sender;
(TYP) Tag of Message Type;
(FUN) Tag of Message Function in Recording Subsystem;
(FID) Tag of Name of Data Recording File;
(KEY) Tags of Key Parameters.

Table 2.

No.	Character Before Data Field	Value of TYP Tag	Value of FUN Tag	Designation
1	STX	0	-	Control message
2	STX	1	1	Sequential reading of values of TSS parameters
3	STX		2	Reading value of TSS parameter
4			3	Entering the value of TSS parameter
5	STX	2	-	Setting up data files
				— File set-up messages —
6	STX	3	0	Position during experiment
7			1	Position at start of experiment
8			2	Position at end of experiment
9			3	Environment at start of experiment
10			4	Environment during experiment
11			5	Environment at end of experiment
12			6	Request for description of parameters
13			7	Final request for description of parameters
14	STX	4	1	Data from prolonged measurements — parameters of a cycle
15			2	Data from prolonged measurements — parameters of entry
16			3	End of recording data from prolonged measurements
17	STX	5	1	Data from one-time measurements
18			2	End of recording of data from one-time measurements

[Table 2 continued, next page]

[Table 2 continued]

No.	Character Before Data Field	Value of TYP Tag	Value of FUN Tag	Designation
19	STX	6	1	Starting process in recording subsystem
			2	Stopping process in recording subsystem
20	STX	7	–	Transition of laboratory computer to regime of recording subsystem terminal
21	STX	8	0	Formation of finite files without entry in SZF [possibly locking filter bay]
22			1	Formation of finite files with entry in SGF [expansion unknown]
23	STX	B	1	SZF entry for trans
24			2	SZF entry for DBACK
25		G	0	Starting DBACK process
26			1	Stopping DBACK process
27		TYP	FUN	Response message on receipt
28	S	TYP	FUN	Response message on error

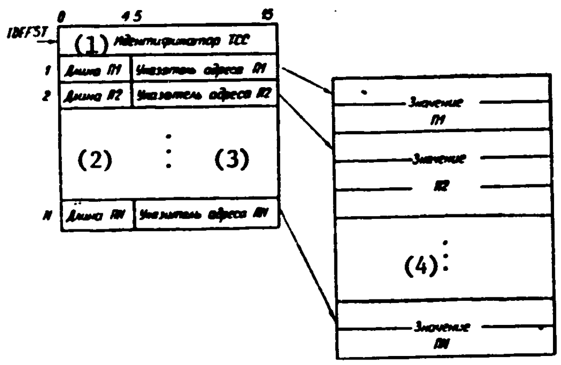


Figure 3. Structure of the TSS.

П are parameters included in the TSS. $1 \leqslant i \leqslant N < k$ (k is the number of parameters in the TOP)

Key:
(1) Identifier of TSS;
(2) Length [of П1, П2...];
(3) Indicator of [address of П1...];
(4) Value [of П1...].

The first part of the TSS contains N+1 words where N is the number of parameters of the Table (for the system being described N = 200) and begins at the heading IDEFST, which occupies one word and makes it possible to localize its position in memory. The N words that follow contain fields with an indication of length in machine words and the addresses of the first word for the value of each TSS parameter. The second part has variable length and contains as many words as are occupied by all the values of the parameters included in the Table at the particular moment of time taken together. The values of the parameters in the second part of the TSS are periodically updated. Most of them are measured by subject complexes during the entire cruise independently of other experiments, while the rest are obtained as the result o. laboratory experiments and observations. Updating times are assigned when the corresponding computing processes are initialized and may be chosen from the set of permissible values for the purpose of increasing system flexibility.

Figure 4 below shows the structure of interaction of computing processes during generation and updating of the TSS, and during recording of data from prolonged measurements of parameters of the water using the hydrologic laboratory equipment in the experiment given in work [1]. Any change in the set of sensors in the underwater unit necessitates an updating of the TOP. For the sake of brevity in describing the interaction of computing processes we will designate each process with the name of its defining program. After each updating the TOB must be analyzed by the computing process defined by the STBIN program, which constructs the internal memory, the supervisory TSS in conformity with the structure shown in Figure 3 above. Parameters are taken from the TOP according to the value of the TSS index. The values of the TSS parameters are updated using the NAVIG, WESTA, ECHOS processes which receive messages from the navigating system (updated at periods of 10 seconds), the automatic weather station (periodicity of 1-10 minutes), the echosounder subsystem of the geophysical laboratory (periodicity of less than one second), and also from the computing processes activated by the programs of the latoratory computers and REC11-REC17 and STBWR programs in the recording subsystem, for example in the case of feeding visual observation data from the keyboard of the synoptic meteorology laboratory computer. The computing processes of the laboratory computers should prepare messages that contain the composite codes of parameters and their values accompanied by the values of the developer tags TYP=1 and FUN=3. When it is necessary to obtain the values of user tags from the TSS by the processes of the laboratory computers, the latter are sent to the message recording subsystem with developer tag values of TYP=1 and FUN=1, 2 (see Table 2 above). Program semaphore engineering is used to protect the TSS against simultaneous access by different computing processes; in this case the next computing process receives access to the resource only after the preceding one has completed its use and thrown the semiphore [6].

When recording data from prolonged measurements a file must first be set up for these data in the recording subsystem. This is done by a statement from the recording subsystem terminal with the help of the INTIN process (see Figure 1 above). Then the computing process of the laboratory computer is activated and, together with the REMIN process, shapes the structure of the file and enters its titles, sending a message with the values of the tags TYP = 2, 3. The necessary user tags may also be put in the file as parameters of the cycle and as recording parameters. This is determined by the content of the message on setting up the file (TYP = 2) which the REMIN process uses to structure the file. The title includes information on the location of the ship and state of the environment. Data for a heading are taken from the TSS by the REMIN process. The computing processes defined by the ASYNKLINK programs transmit messages on communications lines following protocol ISO-1745. After the file is structured the computing programs of the laboratory computers go into operation. In the case of sounding the water with hydrologic laboratory equipment (HLLAB) they request values of the measured parameters from the deck sounding unit, perform preliminary processing and computation of calculated parameters, form messages with data and the values of the developer tags TYP = 4 and FUN = 1, 2, and copy them using the processes shown in Figure 4 below, the HANDY recording subsystem. The HANDY and REMIN processes first store the values of the parameters in the buffer file, then shape the finite files in the format chosen. Data in the buffer files is stored in a form close to that in which it is recorded in the finite file. In this case the necessary values of user tags are added from the TSS. They are requested by the REC11 and STBRE programs accordng to the tag values TYP = 1 and FUN = 1, 2.

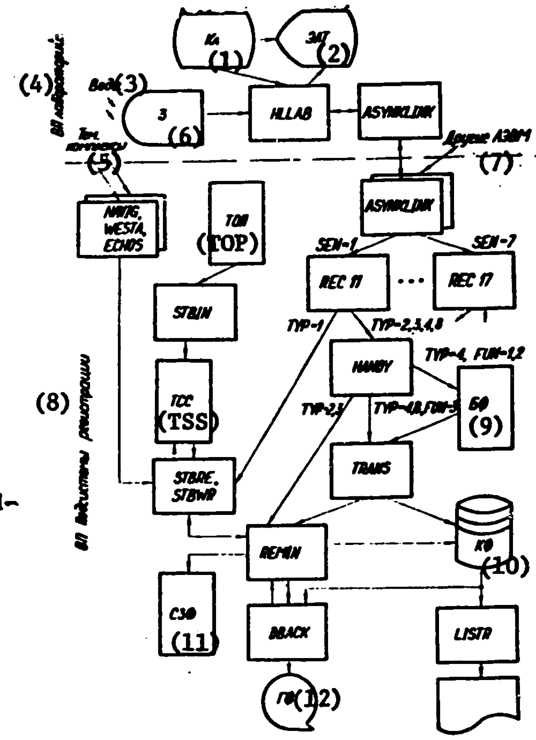


Figure 4. Structure of Interaction of Computing Processes During Generation of TSS and Recording the Data of Prolonged Measurements of Environmental Characteristics.

Key:
(1) Keyboard;
(2) Cathode Ray Tube;
(3) Water;
(4) Laboratory Computing Processes;
(5) Subject Complexes;
(6) Probe Submerged in Water (Submarine and Deck Units);
(7) Other Laboratory Computers;
(8) Computing Processes of Recording Subsystem;
(9) Buffer File;
(10) Finite File;
(11) List of Completed Files;
(12) Ready File.

Data in the form of messages may be received by the recording subsystem simultaneously from several computing processes of different laboratories. Accordingly, several finite files can be constructed and opened for service by the TRANS process simultaneously. Data from all sources are recorded first in the proper file. As necessary the laboratory computing processes output messages according to which data on the location of the ship and state of the environment are recorded in the middle and at the end of the experiment. When recording of data in a file (temporary or final) stops, a message comes from the library with the tag values TYP = 4, FUN = 3, and when it is necessary to complete recording and set up a finite file, the tag values are TYP = 8, FUN = 1. These messages are transmitted to the TRANS process, which shapes the finite file in the format selected, putting in it entries from the buffer according to the tag value FID, which indicates belonging to a certain file. When a small number of sources are working at one time and the density of data flows is not great, the software can be generated in such a way that messages from the HANDY process can go to the TRANS process, bypassing the stage of storage in the buffer file. In this case, the values of the FID tags are used to record data in the required open finite file.

When receiving a message with the tags TYP = 8, the TRANS closes the file identifiable by an accepted value of the tag FID and immediately sets up and structures a new file, free and ready to receive data, under the same identifier. This permits minimizing the fairly protracted and labor-intensive operations of setting up and structuring files. The user performs these operations just once for each type of file structure.

The interaction of computing processes on different levels of the hierarchy is carried on according to this description, as is interaction when recording data from one-time measurements. But in this case user tags are used as key parameters and play a more important role in shaping the finite file. Requests in the form of composite codes of parameters to add to the values of the user tags selected from the TSS are, by the choice of the laboratory operator (user) included in messages received by the processes RECI1-7. The data together with the values of these tags are first stored in a structure close to the data base [7]. Then, based on requests structured according to different user criteria, they may be selected by values of key parameters and grouped in finite files similar to files with data from prolonged measurements.

After completion of a finite file it may be stored on a magnetic disk for some time so that the computing processes of the processing subsystem can request the data stored in it. When the need for this is over, the DBACK process is activated and copies the finite file onto magnetic tape, adding essential information on the cruise, organization, leader of the expedition and experiment, and so on to the title. The ready file is stored on magnetic tape and may be used in the processing system at a later time or transmitted to shore-based computer centers for processing. The LISTR process makes it possible to output a finite file to a display screen or print it for visual monitoring and maintaining the archives.

When forming the finite file the REMIN process makes an entry about it in the third resource of the system, the list of completed files in the format shown in Table 3 below. When it is necessary to get access to data recorded in different finite or ready files, the user scans the contents of the entries in the list of completed files on a screen or initiates a computing process that selects the necessary files according to the contents of the entries. After selecting the necessary files their contents can be manipulated both in the processing subsystem and at shore-based computer centers with the help of computing processes defined by applied programs, making broad use of the values of user tags recorded in the files together with the data.

Table 3.

No.	Content of Field	Length, bytes	Displacement from Start, bytes
1	Name of Station	6	0
2	Name of Experiment	6	6
3	Name of Medium	12	12
4	Name and Version of File	12	24
5	Time of Creation	13	36
6	Time of Completion	12	48
7	Latitude	7	60
8	Longitude	8	67
9	Number of Parameters	2	75
10	Composite Code $\Pi 1$	6	77
11	Composite Code $\Pi 2$	6	83
12	...		
k+9	Composite Code Π_k	6	77+6(k-1)

Π_k is parameter k recorded in the file.

Conclusions

1. The structure of the SANI software of multipurpose scientific research ships must be flexible, easily restructured, accessible for quick assimilation, simple to use, and be able to represent results in graphic form.

2. The flexibility of software structure and possibility of quickly adjusting the SANI to a new group of experiments can be accomplished by introducing three system tables in the recording subsystem (the table of parameter descriptions, the table of the state of the system, ship, and environment, and the list of completed files), as well as means to generate, update, and maintain them.

3. It is advisable to tag data in the SANI of multipurpose research ships beginning from the lowest levels of the hierarchy. Two levels of tags can be identified in such systems. First-level tags are developer tags, designed to control the interaction of data recording processes; second-level tags are user tags, designed to give the data greater informational value in subsequent processing and to control the processes of sorting and retrieving elements of a set of recorded data.

4. The number of developer tags transmitted with the data from certain processes to others at different levels of the hierarchy depends on the structure of the system. With an eye to stepping up the reconfiguration of software it is essential to standardize the formats of the data fields of all messages. Among the developer tags transmitted to the recording subsystem there should be tags for the sender, type and function of the message, recording file, and key parameters. The current values of user tags should be kept in the state table.

FOOTNOTES

1. O. S. Zudin, S. N. Domaratskiy, I. P. Kotik, G. N. Kuklin, A. A. Novikov, L. S. Sitnikov, O. Laaksonen, and R. Aarinen, "Integrated Systems of Automatic Scientific Research on Multipurpose Research Vessels," AVT. 1981, No. 6, pp 72-80.

2. "The International Data Management Plan for the GARP Atlantic Tropical Experiment. Part 1. General Description of the GATE Data Management Scheme and Its Specification," Geneva, 1974, April, (GATE Report No 13."

3. A. Butrimenko, and Dzh. Sekston, "Protocols in Communications Networks for Transmitting Digital Information," AVT, 1978, No 6, pp 56-65.

4. G. I. Marchuk, and V. Ye. Kotov, "Problems of Computer Technology and Pure Research," AVT, 1979, No 2, pp 3-14.

5. S. N. Domaratskiy, O. S. Zudin, and O. Vaynio, "The Organization of Interaction of Instruments and Computers in Variable-Structure Systems To Automate Scientific Research," AVT, 1981, No 3, pp 81-91.

6. E. W. Dijkstra, "Cooperating Sequential Processes in Programming Languages," "Academic Press", 1968 pp 43-112.

7. J. Martin, "Computer Data-Based Organization," 2nd Ed, "Prentiss-Hall Inc.", Englewood Press, N. J., 1977, pp 451-487.

11,176
CSO: 1863/120

NETWORKS

UDC 681.324

FORMS OF COMPUTER NETWORK ORGANIZATION

Riga AVTOMATIKA I VYCHISLITEL'NAYA TEKHNIKA in Russian No 1, Jan-Feb 82 (manuscript received 8 Jul 81) pp 3-11

[Article by E. A. Yakubaytis: "The Architecture of Regional and Local Computer Networks"]

[Text] Computer networks are a highly efficient base for the contemporary data processing industry. A single network consisting of large, medium, and small computers can provide [1] access to the most diverse information and computing resources; it can process work data, retrieve necessary data and documents, and control scientific research, design development, and industrial equipment. Associating (combining) computer networks makes it possible to transfer and process information related to any object of the national economy.

Computer networks differ by size, productivity, methods of data processing, and types of equipment in use. At the same time they all have characteristics that define them as the key contemporary means for dynamic distributed processing of large data arrays. Therefore, the present article considers various classes of computer networks constructed on the same architectural principles.

The fundamental element of the model of a computer network is the logical system, a group of functions realized in the computer complex consisting of one or several computers (see Figure 1 below). The systems together with their lines of interaction, which are called physical connections, form the logical structure of the computer network. Each of the systems is open if it meets the general requirements [2] for the architecture of interaction of open systems established by the International Standards Organization (ISO).

All systems are divided into seven levels. Each level performs a definite task in the computer network and provides service to the level located above it. The collection of rules for interaction among objects of the same level of different systems is called the protocol.

We will divide all systems into two groups. The systems that route and transmit information we will call communications systems. The systems that either provide or use the resources of the computer network will be subscriber systems. The structures of a subscriber system and communications systems are shown in Figure 2 below. Levels 4-7 of communications systems do not participate in

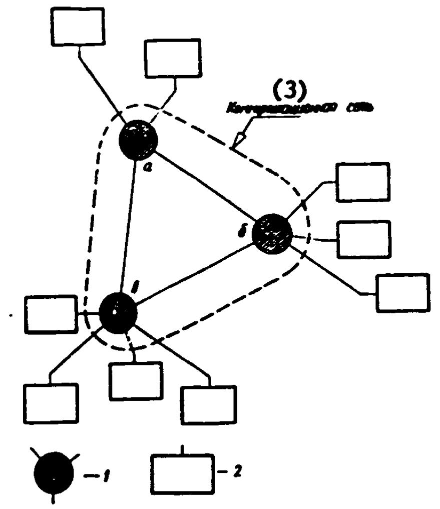


Figure 1. Logical Structure of a Computer Network.

Key: (1) Communications Systems;
(2) Subscriber System;
(3) Communications Network.

primary control related to routing data arrays. Therefore they are omitted in Figure 2.

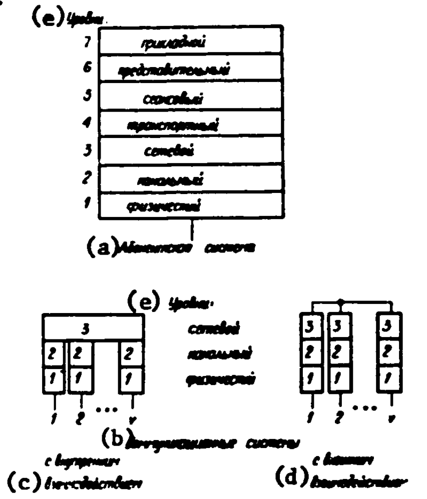


Figure 2. Structures of Systems.

Key: (a) Subscriber System;
(b) Communications Systems;
(c) With Internal Interaction;
(d) With External Interaction;
(e) Levels:
(1) Physical;
(2) Channel;
(3) Network;
(4) Transport;
(5) Session;
(6) Representative;
(7) Applied.

Two types of structures are used for communications systems depending on the method of switching. In the first type there is a common third level that insures execution of all network functions for the physical connections. In the second structure v identical sets of third-level functions are formed and connected with one another by means of external network links. The first structure is simpler, but as will be shown below, the second allows greater reliability of functioning in the communications system.

As stated above, each of the systems is realized with one or several computers. Moreover, several systems can be realized in one computer. Each connection that links two systems is realized by a data transmission channel. As a result, the logical structure of the computer network is converted into the physical structure of the network.

Subscriber complexes perform the functions of subscriber systems and therefore offer the most diverse information and computer resources to a broad range of users. Each complex consists of ψ, where $\psi > 1$, computers and the network adapter. There are three possible variations for distribution of the functions of the subscriber system. They are shown in the table below.

Table. Distribution of Functions Between Computers and Network Adapter

Variation	Functions Realized by Adapter	Functions Performed by Computer
I	Physical, channel, network, and transport (levels 1-4)	Session, representative, and applied (levels 5-7)
II	Physical and channel (levels 1-2)	Network, transport, section, representative, and applied (levels 3-7)
III	Physical and part of channel (levels 1, 2a)	Part of channel, network, transport, session, representative, and applied (levels 2b, 3-7)

The first variation is preferable because the network adapter in it realizes all functions that depend on the standards of the interface with the communications network. In this case the computer is spared all jobs related to transmitting data arrays and only performs its primary functions, defined by levels 5-7 of the protocols. The network adapter here is built on the basis of a small minimachine or a group of microprocessors.

Because most of the computers that have been produced still do not have network adapters, in practice variations II and III are most widely used for the physical structures of subscriber complexes. Variation II is preferable between them. Variation III is used in those cases where the computer in its assortment of external units does not have the adapters necessary for the first two variations. In this case, the adapter performs only those functions which it is very difficult or impossible to realize by program means in the computer (insuring physical connections, performance of bit-staffing operations, assigning flags that mark the start and end of a frame, and so on).

The realization of a subscriber system in two units (computer and adapter) requires introduction of a channel (line) (abbreviated "ksh") designed to link them. For this reason, in addition to the protocols which define (see Figure 2 above) the logical structure of the subscriber system, there appear additional logical levels (1 ksh and 2 ksh) which are necessary to interlink and control the channel (line) that links the computer and the adapter.

Figure 3 below shows an example of this kind of structure of a subscriber complex (variation II from the table). Here the upper levels of the logical

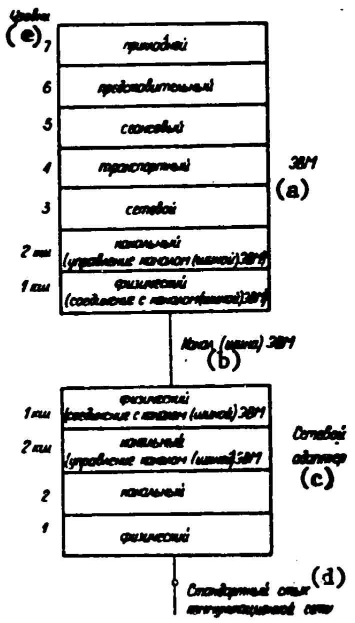


Figure 3. Structure of Subscriber Complex.

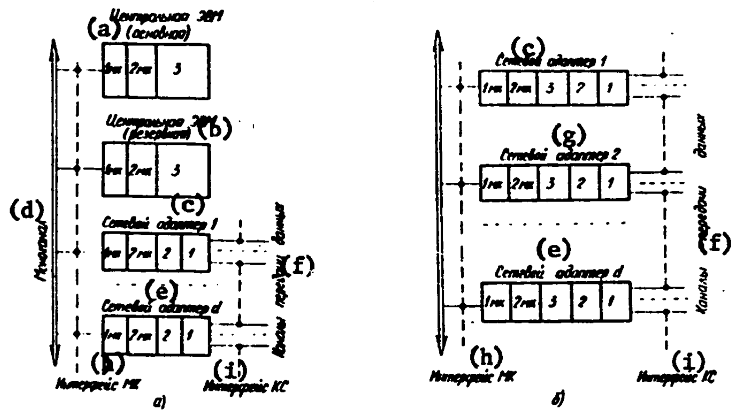
Key: (a) Computer;
(b) Channel (Line) of Computer;
(c) Network Adapter;
(4) Standard Interface of Communications Networks;
(5) Levels:
(1ksh) Physical (Connection with Computer Channel/Line);
(2ksh) Channel (Control of Computer Channel/Line;
(3) Network;
(4) Transport;
(5) Session;
(6) Representative;
(7) Applied.

structure of the network (3-7) are realized in the computer, while the lower ones (1-2) are realized in the network adapter. Two additional logical levels are introduced in the computer and adapter for linking the adapter to the computer through a channel or line. These levels (1ksh and 2ksh) provide interlinkage with the computer channel (line) and control of this channel.

The two types of communications system (see Figure 2 above) are realized, respectively, in two types of communications complexes (see Figure 4 below), which are often called communications centers. As Figure 4 shows, each center consists of a monochannel and the central computers and network adapters connected to it.

We will use the term monochannel for the physical medium and the hardware and, possibly, software used collectively by a significant number of subscriber-adapters. One monochannel (see Figure 4) can interconnect hundreds of adapters in central computers if the latter are used in the structure of the communications complex. The physical medium of the monochannel may be radio, a light guide, coaxial cable, group of parallel wires, or a cable with intertwined pairs of strands. The size of the monochannel ranges from dozens and thousands of meters, and sometimes even thousands of kilometers. The monochannel is used collectively by dozens and hundreds of subscriber complexes. It is often also called the line.

Figure 4. Types of Structures of the Communications Complex.



Key:	(a) Central (Primary) Computer;	(f) Data Transmission Channels;
	(b) Central (Standby) Computer;	(g) Network Adapter 2;
	(c) Network Adapter 1;	(h) Interface MK;
	(d) Monochannel;	(i) Interface KS.
	(e) Network Adapter d;	

When transferring from the communications system to the communications complex it should be kept in mind that the communications complex is built on a microprocessor basis to increase the speed of data transmission. Therefore, to link the different processes it is necessary to introduce additional logical levels (1nk and 2nk) which insure interaction with the monochannel.

The physical structure of the communications complex which realizes a communications system with internal interaction (see Figure 2 above) is shown in Figure 4a. Here the network (third) level is realized by a central minicomputer or microcomputer. The lower levels (first and second) are done by individual adapters. Multiplexers are added to the adapters for channels with low speeds. Then one adapter can control 4-8 channels at one time.

The communications system with external interaction is realized by the physical structure shown in Figure 4b. In this diagram each adapter performs the functions defined by three levels (1-3) of the protocols of the computer network. Multiplexers are added to the adapters, as in Figure 4a, for low speeds.

Depending on the type of physical medium used and the size of the monochannel, we receive two types of communications complexes: concentrated and distributed. In the concentrated complex all adapters and central computers, if there are any, are arranged in several rows of stands (cabinets). In this case the size of the monochannel is only dozens of meters.

As for the distributed complex, its elements (adapters or central computers) can be located at considerable distances from one another. The length of a coaxial monochannel connecting them (without repeaters) can be up to one kilometer. When a radio channel is used the elements of the communications complex may be thousands of kilometers apart from one another.

The structure of the communications complex has a high level of reliability, but it does not have one central point that defines the reliability of transmission. And the reliability of the complex with external interaction is particularly high. In this case (see Figure 4b) there are no central computers at all, and a failure by an adapter can (if the system does not have the necessary switches) lead to the failure of one or several channels.

Communications complexes include two types of interfaces. The MK interface (see Figure 4) determines the standards for interlinking central computers and adapters with the monochannel. Communications complex that form a single communications network can use all different kinds of MK interfaces. The KS interface characterizes the standards of interaction between adapters and communications channels and, through them, with subscriber complexes. The KS interfaces must meet the standards adopted in the communications network.

Computer networks can be divided into two classes: local and regional. We will use the term local network for a computer network whose subscriber complexes are located close to one another (usually in one building or a few adjacent buildings).

We will call a computer network whose subscriber complexes interact with one another through a distributed data transmission network a regional network. Each regional network can cover a large city, an oblast, a republic, or the territory of the entire country.

Consolidating regional and local computer networks makes it possible to set up associations of networks that afford an economically sound base for processing enormous arrays of information. The local networks are the key elements of these associations.

Three types of associations of computer networks should be identified. The first comprises networks in which different sets of protocols are used. These networks are interconnected by internetwork interface convertors (see Figure 5 below) which provide logical data conversions necessary when data is transmitted from one network to another. The second type is computer networks in which the same protocols of levels 1-3 are used, but the protocols of the higher levels (4-7) differ. In this case the computer network uses a common communications network, but only those subscriber complexes which belong to the same network can interact directly with one another. Additional interface conversions are necessary for interactions among subscriber complexes located in different networks. Naturally, their structure in associations of the second type is simpler than in associations of the first type.

The third type of association includes computer networks which have uniform protocols for all seven levels. These associations provide interaction for any set of subscriber complexes regardless of the computer networks to which they belong. All that is required is to indicate the universal address of the

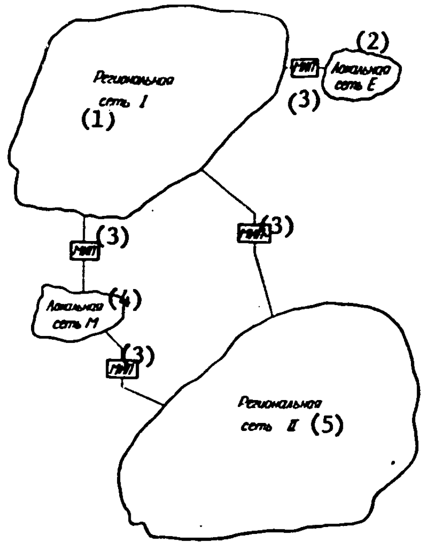


Figure 5. Association of Different Types of Computer Networks.

Key: (1) Regional Network I;
(2) Local Network E;
(3) Internetwork Interface Convertor;
(4) Local Network M;
(5) Regional Network II.

subscriber, which includes three subaddresses: network, subscriber complex, and applied process within this complex. Figure 6 below shows an example of such an association. It includes several local and regional computer networks. The regional networks in the association may also overlap, that is, one or several subscriber complexes of one network may also belong to another regional network (for example, complexes a and b in Figure 6). When this is necessary, any local network can be considered a large distributed subscriber complex of the regional network.

The widespread development of large-scale integrated circuits (LSIC's) greatly reduced the cost of data processing hardware. At the same time, processing costs are declining much more rapidly than the cost of data transmission channels. The importance of interactive methods of solving many problems related to collecting, storing, retrieving, and processing various documents is growing every year. The requirements for flexibility of computer networks are also growing. All this supports rapid development of the processes of decentralization of data processing. And if we consider that the primary data flows are contained within the enterprise or organizations, the reason for the appearance and rapid development of local computer networks becomes clear. The local networks insure greater data processing reliability than large regional networks; optimization of the processes of data processing are much easier; software is simpler; and, better conditions are created for integrating the processing of different types of data (management control, control of industrial processes, processing work documents, and the like).

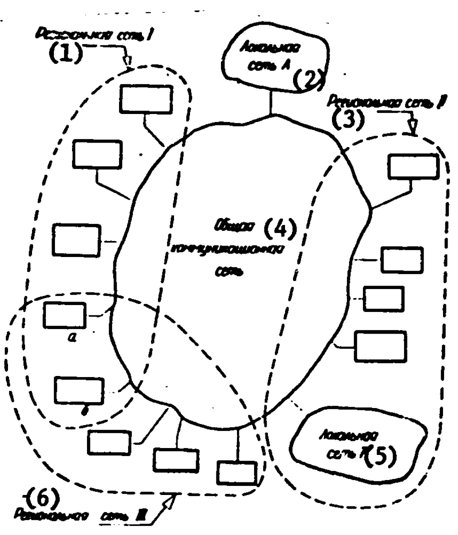


Figure 6. Association of Computer Centers of One Type.

Key:

(1)	Regional Network I;
(2)	Subscriber Network A;
(3)	Regional Network II;
(4)	Shared Communications Network;
(5)	Local Network T;
(6)	Regional Network III.

In the local computer network, owing to the short distances between subscriber complexes, it is not necessary to use telephone channels and the speed of transmission of data arrays can be increased quite simply. Because errors occurring during transmission are reduced in this manner, the algorithms for finding and eliminating these errors are simplified.

Local networks are divided into two groups according to functional designation. The first group is general-purpose computer networks designed for all possible types of data processing at large institutions, associations, or science centers. The second group is made up of specialized local computer networks. Among them, for example, are networks for performance of planning work at design bureaus, networks for financial transactions at state banks, and the like.

Networks are divided into single-center and multi-center networks depending on the nature of switching of data arrays. The establishment of a large number of communications complexes in a local network is an unacceptable luxury. Therefore, as a rule, in multi-center networks communications systems are realized in the same computers where the subscriber systems function, and the result is the formation of subscriber-communications complexes. In addition, the local networks establish the required number of subscriber complexes.

Multi-center local networks can be broken into three groups (see Figure 7 below). The branching network is used most frequently in those cases where a base (located at the base of the tree) complex must be connected with a series of other complexes. For example, the local networks shown in Figure 7a include three subscriber-communications (1-3) and six subscriber (4-9) complexes. The communications system (logical switching centers) in the complexes are arbitrarily identified by a dotted line.

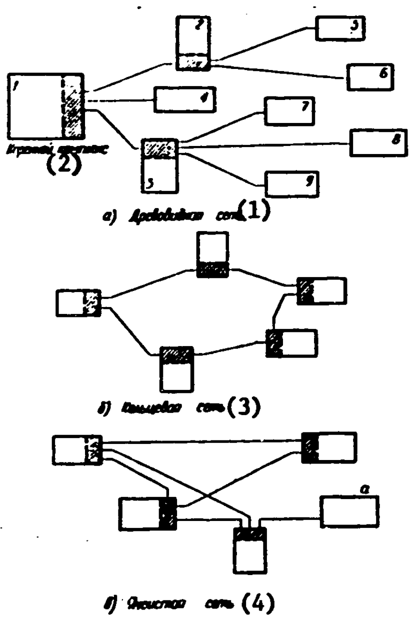


Figure 7. Structures of Multi-Center Local Networks.

Key: (1) Branching Network;
(2) Base Complex;
(3) Circular Network;
(4) Cell Network.

The branching local networks are fairly simple, but they have low reliability because at the outlet from the base complex system the network breaks into parts or completely stops work (if the base complex is a central one). Moreover, the branching network differs from the others because it costs more to set up long data transmission channels.

The local circular networks consist of complexes of the same type: subscriber-communications complexes. In other words, each complex contains a logical switching center. Thus, Figure 7 shows a five-center circular network. In principle information can be transmitted in both directions around the circle, but in practice this is very difficult, so information is ordinarily transmitted in one direction only. When comparing the circular network with the branching network, it should be observed that the former affords more economical methods of connecting complexes and does not have a base complex. But its reliability is also low. A break in the circle often means that the whole network fails [4]. In the cell-type local network (see Figure 7) the large majority of complexes also have logical switching centers. Only the deadend complexes (for example complex a), each of which is connected with the network by just one data transmission channel, do not have such a center. In those cases where each complex of the cell network is connected with all other complexes by channels, the network is called a fully linked network.

The fully linked network has high reliability because in a large majority of cases a failure of one of the complexes does not lead to a shutdown of the rest of the network. But a fully linked cell network has poor modular capability because adding one new complex (complex i) to it requires the gradiation of i-1 new data transmission channels. Expenditures for a large number of logical switching centers in data transmission channels are also significant.

In recent years single-center local networks have become increasingly widespread. (Figure 8 gives an example of one.) The efficiency of this network results from the fact that one of the complexes (the hachured one in Figure 8) specializes in the functions of a communications system. As for the other complexes, they are subscriber complexes and use all their resources to perform their primary task: offering or using the resources of the computer network.

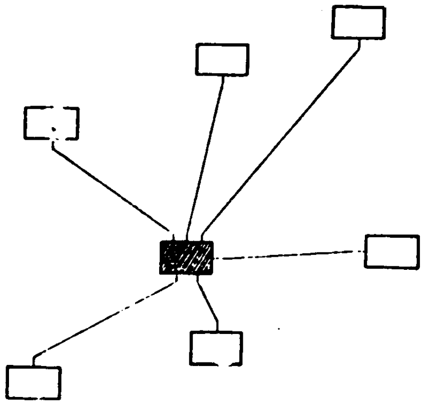


Figure 8. Structure of a Single-Center Local Computer Network.

The single-center network is outstanding for simplicity of realization, diagnosis, and administrative control. It also has good modular capability. Adding a subscriber complex or removing one from the network does not change the nature of work of the entire network.

Some people think that the single-center network has relatively low reliability because it has just one communications center, and when it fails the entire network goes down. This was indeed true in earlier local networks where one computer served as the communications complex. In recent years, however, development of the architecture of communications complexes and broad use of microprocessors in it has enabled the single-center network to become the most reliable type of local computer network.

In fact, a review of the structures of contemporary communications complexes (see Figure 4) shows that the failure of one active element (central computer or adapter) hardly affects the work capability of the complex at all. The monochannel is passive and therefore has high reliability. Furthermore, if necessary all elements in the center may have backups.

As pointed out above, communications centers are grouped as concentrated or distributed depending on the type of monochannel (see Figure 4 above). In conformity with this the local network may have a communications center installed at the approximate geometric center of the network. The center of this network can be distributed and have a monochannel in the form of a light guide, coaxial cable, or twisted pair of wires. Figure 9 shows an example of such a network.

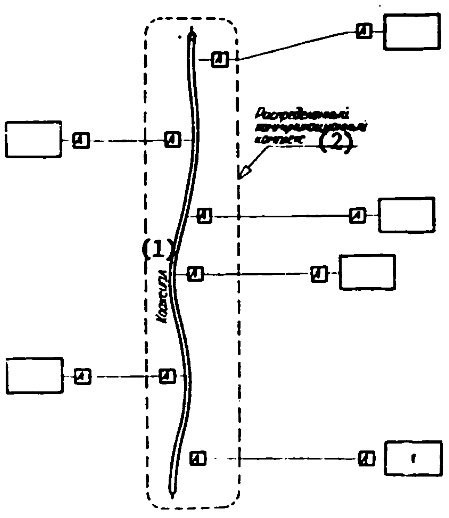


Figure 9. Local Network with Distributed Center.

Key: (1) Coaxial Cable;
(2) Distributed Communications Complex.

In this case the subscriber complexes are connected into the local computer network through adapters (a) and coaxial cable.

Local networks where each has one distributed switching center that performs the functions of a communications system with external interaction (see Figures 4-6) are becoming increasingly popular. This popularity results from the fact that these networks have a number of important advantages, chief of which are the following: high reliability; simplicity of set-up and reconfiguration; and, the possibility of connecting not only a computer but also various peripheral units into the network without complications. In addition it is important that such a network is a fully linked one, that is, that every subscriber complex in it interacts directly with all other subscriber complexes.

BIBLIOGRAPHY

1. Yakubaytis, E. A., "Arkitektura Vychisletel'nykh Setey" [Architecture of Computer Networks], Moscow, "Statistika", 1980, 278 pages.

2. "Reference Model of Open Systems Interconnection," ISO-TC 97/SC, 1979, Vol 16, No 227, pp 1-181.

3. "CCITT Sixth Plenary Assembly." Geneva, 27 September-8 October1976. Orange Book. 2, Public Data Networks, Geneva, 1977, Vol 8, 217 pages.

4. Bass, C., Kennedy, J., and Davidson, J., "Local Network Gives New Flexibility to Distributed Processing," ELECTRONICS, 1980, No 21, pp 114-122.

11,176
CSO: 1863/120

PUBLICATIONS

TABLE OF CONTENTS FROM JOURNAL 'AUTOMATION AND COMPUTER TECHNOLOGY', JANUARY-FEBRUARY 1982

Riga AVTOMATIKA I VYCHISLITEL'NAYA TEKHNIKA in Russian No 1, Jan-Feb 82 pp 93-94

[Text] Table of Contents

11,176
CSO: 1863/120

UDC 62-50

CYBERNETICS AND COMPUTER ENGINEERING: DISCRETE CONTROL SYSTEMS

Kiev KIBERNETIKA I VYCHISLITEL'NAYA TEKHNIKA: DISKRETNYYE SISTEMY UPRAVLENIYA in Russian No 53, 1981 (signed to press 16 Jun 81) pp 2, 118-120

[Annotation and abstracts of articles from collection "Cybernetics and Computer Engineering", No 53, "Discrete Control Systems"; editorial board: V. M. Glushkov (editor-in-chief), V. L. Volkovich, G. F. Zaytsev, V. M. Kuntsevich (assistant editor-in-chief), A. I. Kukhtenko (assistant editor-in-chief), V. B. Larin, B. Yu. Mandrovskiy-Sokolov (issue secretary-in-chief), V. S. Mikhalevich, Yu. I. Samoylenko, K. G. Samofalov, A. A. Stogniy, B. B. Timofeyev, A. A. Tunik, N. S. Furs (series secretary-in-chief) and A. G. Shevelev; Institute of Cybernetics, UkSSR Academy of Sciences, Izdatel'stvo "Naukova dumka", 1000 copies, 120 pages]

[Text] This collection is devoted to the solution to current problems in discrete control systems that make use of specialized computers and devices. Considered are problems of analysis and synthesis of optimal control systems with regard to determinate and random external disturbances, as well as of the practical use of discrete systems to control physical experiments and technological processes.

For sceintific workers, post-graduates and students in senior courses in VUZ's engaged in problems of discrete control systems.

UDC 681.516.75

OPTIMIZATION OF PARAMETERS OF LINEAR PERIODICALLY NONSTATIONARY AUTOMATIC CONTROL SYSTEMS

[Abstract of article by V. I. Gostev and A. I. Il'nitskiy]

[Text] A graphic-analytical method is suggested for optimizing parameters of linear periodically nonstationary automatic control systems by integrated criterion. Figs. 5, Bibl.: 4 titles.

UDC 681.511.22

DISCRETE MULTICHANNEL AUTOMATIC SYSTEM WITH CHANNEL SERVICING BY WEIGHT

[Abstract of article by A. V. Danil'chenko, G. F. Zaytsev and I. A. Izotov]

[Text] A multichannel automatic system is discussed for correlation processing of random signals with one correlation feedback connectable to compensating channels by weight criterion. Figs. 2, Bibl.: 1 title.

UDC 681.3.01:621.372.5

DESIGN TECHNIQUE FOR DIGITAL SHAPING FILTER WITH FIXED POINT AND MAXIMAL DYNAMIC RANGE

[Abstract of article by A. A. Petrovskiy and A. Ye. Leusenko]

[Text] A technique has been developed for designing digital shaping filters with fixed point that allows preventing overflow of summing units when processing information with selection of maximal absolute value of input signal equal to upper bound of filter dynamic range. Figs. 4, tables 2, bibl.: 10 titles.

UDC 681.5

APPLICATION OF ERGODIC THEOREM TO NONSTATIONARY OBJECTS

[Abstract of article by M. M. Makarchuk]

[Text] For nonstationary nonlinear objects of the class of objects with limited second moment, that have finite storage, and the variable coefficients of models of which meet Dirichlet's condition, ergodic relations have been derived to determine the constant models of these objects. Bibl.: 6 titles.

UDC 62-50

METHOD OF ITERATION REFINEMENT OF LINEAR OBJECT CONTROL TRAJECTORY WITH LIMITED CONTROL

[Abstract of article by Shcherbashin]

[Text] An economical algorithm is considered for step-by-step improvement of control trajectory without violating conditions of two-point boundary problem, based on methods of linear programming. Boundary problem is solved by using inverse matrix computed in advance. Bibl.: 7 titles.

UDC 62-50

ADAPTIVE CONTROL OF STATIC OBJECTS WITH UNKNOWN PARAMETERS VARYING IN TIME

[Abstract of article by V. M. Kuntsevich and M. M. Lychak]

[Text] Discussed is the problem of control of static objects (objects without storage), the true parameter vector value of which is unknown before starting control. The game approach is used to solve it. In the process, on the one hand, control is selected in such a way that enables achieving the basic (initial) aim of control, and on the other, that enables solving the problem of identification of object parameters to adapt the control system to specific characteristics of the controlled object. Bibl.: 3 titles.

UDC 621.319.8519 :

CONVERGENCE OF METHODS OF MAXIMUM OPTIMIZATION OF FIRST AND SECOND ORDERS

[Abstract of article by S. L. Ivanov and V. Ya. Katkovnik]

[Text] Discussed are conditions of convergence of iteration algorithms of the Newton and gradient projection type for solving maximum extremum problems. The given conditions are investigated in detail in the case of solving problems for the conditional extremum, minimax and regulation. Figs. 2, bibl.: 11 titles.

UDC 681.3.62.52

ANALOG-DIGITAL AUTOMATIC SIGNAL SPATIAL-TIME PROCESSING SYSTEM

[Abstract of article by A. V. Danil'chenko, G. F. Zaytsev and I. A. Izotov]

[Text] Discussed is an automatic system for spatial-time processing of signals with high spatial resolution of jamming signal sources with digital optimization of sampling of signals for correlation-time processing. A mathematical model of an optimizer that enables priority sampling by intensity of a specified number of jamming signals has been constructed. Figs. 2, bibl.: 4 titles.

UDC 62.50

ON ONE PROBLEM OF ADAPTIVE CONTROL OF A NONLINEAR STATIC OBJECT

[Abstract of article by L. S. Zhitetskiy]

[Text] Discussed is the problem of adaptive control of a nonlinear static object with a fixed set of control actions when interference is present. It is shown that under certain assumptions relative to properties of object and external actions, the solution to this problem can be reduced to solving a nonfinite system of inequalities. A recurrent algorithm is constructed for adaptive control and sufficient conditions are formulated for convergence of this algorithm within a finite number of steps. Bibl. 6 titles.

UDC 62.50

ADAPTIVE ASSESSMENT OF PARAMETERS AND STATUS OF STATIC OBJECT

[Abstract of article by O. M. Oleksenko]

[Text] The problem of one-time assessment of object status vector and identification of its unknown parameters is considered for a static object. Sufficient conditions of optimality are applied. Closed expressions of adaptive assessment are derived in the case of a scalar unknown parameter. Bibl. 3 titles.

UDC 62.50

SOLVING PROBLEM OF SYNTHESIS OF OPTIMAL CONTROL OF DISCRETE LINEAR STATIONARY DYNAMIC OBJECT IN FINITE TIME INTERVAL

[Abstract of article by V. V. Volosov]

[Text] Problem of synthesis of optimal control of linear discrete dynamic object is solved. Convex quadratic functional is used as optimality criterion. New solution is derived for problem with free right end, based on using mathematical programming methods. Problem of synthesis of optimal control for case of moving right end, belonging to specified spectrum of phase space, is solved. Bibl. 8 titl.

UDC 62.50

SYNTHESIS OF MULTIPOINT DISCRETE MODELS OF CONTINUOUS PROCESSES AND THEIR APPLICATION FOR DESIGN OF OPTIMAL CONTROL

[Abstract of article by L. M. Boychuk]

[Text] New approach is considered for construction of multipoint discrete models of continuous controlled processes. Models derived are non-local type, but also have some properties of local spline-type models. Their application allows simplifying computational procedure for design of optimal control of dynamic objects, since it reduces the number of limitations in the form of equalities in the corresponding mathematical programming problem. Bibl. 11 titles.

UDC 519.8

PARAMETRIC METHOD OF LINEARIZATION FOR UNCONDITIONAL PROBLEM OF DISCRETE MINIMAX

[Abstract of article by V. M. Panin]

[Text] It is shown that with the introduction of a scalar parameter, one can not only derive the linear rate of convergence of the linearization method, but also avoid additional computations when determining the step multiplier. The suggested method for speeding up convergence is simpler than other known methods. Bibl. 6 ti.

UDC 519.688

FINITE-CONVERGENT ALGORITHM FOR SOLVING DENUMERABLE SYSTEM OF INEQUALITIES

[Abstract of article by G. M. Bakan and Ye. A. Nizhnichenko]

[Text] Algorithms are suggested that allow obtaining an estimate of the set of solutions as a whole while finding the solution to a system of inequalities. Numeric experiments with these algorithms have shown satisfactory results. Bibl. 5.

UDC 62-50

VECTOR UNDEFINED QUANTITIES AND STATIC OBJECT CONTROL WITH PARTIAL INFORMATION

[Abstract of article by A. P. Nesenyuk]

[Text] Introduced are concepts of vector undefined quantities, general characteristics of undefined quantities and operations on general characteristics. The concept of undefined quantities is applied for solving the problem of static object control with partial information. In the process, a priori general characteristics of unknown parameters of the object are refined in the control process by using a modified method of least squares, and optimal controls are sought in closed mode. Bibliography: 3 titles.

UDC 62-50

ANALYSIS OF STATIONARY MODE IN FREQUENCY-DURATION PULSE SYSTEM FOR CONTROL OF AUTONOMOUS ELECTRIC DIRECT-CURRENT DRIVE WITH INDEPENDENTLY EXCITED MOTOR

[Abstract of article by I. F. Radchenko]

[Text] Discussed is stationary mode of autonomous electric drive, consisting of storage battery, force pulse converter and direct-current motor with separate excitation with armature current control. Analysis is made of efficiency function, sampled in form of power of switching loss in converter and at active resistances of equivalent circuit. The optimal relationship of frequency, duty factor and parameters of drive is derived, which ensures minimum loss in drive. Recommendations are made for selecting type of modulation in converter. Results obtained are used for analysis of concrete drive equivalent circuit. Figs. 3, bibl.: 9 titles.

UDC 681.3:62-52

INVARIANCE IN DIGITAL-ANALOG SYSTEMS FOR CONTROL OF VECTOR RANDOM PROCESSES

[Abstract of article by A. A. Tunik and M. I. Ryzhkov]

[Text] Authors discuss effect of precision of operation of analog subsystems on convergence and precision of functioning of algorithms for control of a digital subsystem in a hybrid (digital-analog) system for control of vector random process. Invariance of digital subsystem to interference in analog subsystems is shown. Bibliography of 5 titles.

UDC 681.332.5

SOME DESIGN FEATURES OF DIGITAK SYSTEMS FOR AUTOMATIC CONTROL OF SPECTRAL CHARACTERISTICS OF RANDOM TIME SEQUENCES

[Abstract of article by M. A. Gnatyuk, B. Yu. Mandrovskiy-Sokolov, A. G. Nechayev and A. A. Tunik]

[Text] Authors discuss implementation of digital systems for control of spectral characteristics by using digital computers and specialized processors for various functions. Features of their operation and some technical characteristics are analyzed. Figs. 4, bibl.: 12 titles.

UDC 62.50

CORRECT CONTROL OF SPECTRUM OF RANDOM PROCESSES AT OUTPUT OF OBJECT WITH POORLY DEFINED MATRIX OF FREQUENCY RESPONSES

[Abstract of article by V. P. Yakovlev and A. I. Savenkov]

[Text] Problem of correct finding of control matrix of spectral densities is considered. The studied method of solution is based on determining the stable projection of the unstable solution with a poorly defined matrix of object frequency responses by using methods of reducing it to triangular shape with subsequent diagonalization. Bibliography of 14 titles.

UDC 62-50

DEFINING FEEDBACK MATRICES IN ALGORITHMS FOR CONTROL OF SPECTRAL CHARACTERISTICS OF VECTOR RANDOM PROCESSES

[Abstract of article by M. M. Lychak and N. K. Brovdiy]

[Text] Problem of control of spectral matrix of vector random process at output of multidimensional dynamic object is discussed. Technique is given for defining feedback matrices in matrix algorithm of control for matrices of frequency responses of object of any type (well shaped, poorly shaped, degenerated). Bibliography of 8 titles.

8545
CSO: 1863/73

UDC 62-50

PROBLEMS OF CONTROL IN ENGINEERING, ECONOMICS AND BIOLOGY

Moscow PROBLEMY UPRAVLENIYA V TEKHNIKE, EKONOMIKE, BIOLOGII in Russian 1981 (signed to press 13 Aug 81) pp 222-223

[Table of contents from book "Problems of Control in Engineering, Economics and Biology", editor-in-chief Ya. Z. Tsypkin, corresponding member of the USSR Academy of Sciences, Izdatel'stvo "Nauka", 1900 copies, 230 pages]

[Text] Contents Page

8545
CSO: 1863/77 END