JPRS L/9773

5 June 1981

USSR Report

CYBERNETICS, COMPUTERS AND AUTOMATION TECHNOLOGY

(FOUO 15/81)

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/9773

5 June 1981

USSR REPORT

CYBERNETICS, COMPUTERS AND AUTOMATION TECHNOLOGY

(FOUO 15/81)

CONTENTS

ERRATUM: In JPRS L/9766, 1 June 1981 (FOUO 14/81) of this series, please change "mobile robot" to "underwater robot" in the following places: page 77, next to last line; page 79, caption for fig. 1.7; page 80, eighth line from bottom; page 81, caption for fig. 2.4.

HARDWARE

HYBRID LSI MANUFACTURING TECHNOLOGY

Moscow PRIBORY I SISTEMY UPRAVLENIYA in Russian No 1, Jan 81 p 40

[Article by engineers A.I. Yakubinskaya, A.A. Kisilev and I.N. Pervunitskiy: "Technological Features of Manufacturing BGIS (Hybrid Large-Scale Integrated Circuits) Designed for Special Applications"]

[Text] The manufacture of hybrid special application LSI with high levels of integration is dictated by their use in digital-analog circuits. The design and technology of manufacturing LSI provides for the installation on plates of type K10-17 mounted capacitors, powerful transistors, and no-rack semiconductor circuits with flexible and ball-type leads. A complex intracircuit commutation is required here, which is executed in two layers, separated by an insulating layer of organic dielectric.

Existing manufacturing methods for multilayer separation of thin-film circuits involve positioning the first electrical conducting layer on the insulating base, then a second insulating layer and a second conducting layer. In order to achieve electrical contact between the layers, a selective etching of the dielectric is accomplished using a protective mask, a process which sometimes causes shorts to occur through the punctures in the protective film, resulting in a subsequent reduction in reliability.

A feature of the proposed technology is the absence of etching the dielectric film, a continuous layer 15-20 micrometers in thickness, a feature which insures reliability in the entire circuit.

Electrical contact between the metal plated layers is achieved by the use of interlayer contacts—volume copper specialized leads, obtained by galvanic buildup on first layer conductors through a positive photoresist mask. Copper with a sub-layer of vanadium is employed for the first layer conductors.

Current supply is provided through jumpers, which are then removed. A feature of the plating process is that to obtain equal rate of coating or precipitation upon the entire surface, the base is suspended in the galvanic bath at an angle of 80 to 85 degrees to the anode. This process is intensified through the use of variable polarity current. The approaches outlined provide for the formation of copper columns (leads) 15-20 micrometers in height, with a height variance on the entire surface of not more than 2 micrometers, with smooth surface, fine-crystalline structure, and strong adhesion to the sprayed copper. The material used for the interlayer insulation is PAK-1, a polyimide electrical insulating lacquer, which is advantageous for its high mechanical and electrical insulating properties over a broad temperature range (from minus 200 to plus 400 degrees C), properties lacking in other dielectrics. The insulating layer is a continuous film of lacquer 15 to 20 micormeters thick, consisting of four layers, providing high electrophysical parameters for crossovers, and minimal porosity. The lacquer is poured onto the base surface and is subsequently equally distributed over the entire surface of the base by centrifuging.

To achieve electrical contact of the conductors of the second layer with the interlayer junctions, the lacquer layer located at the tops of the interlayer junctions is removed until the tops of the junctions are uncovered. For this, burnishing or undercutting of the uppper insulating layer may be employed. The selection of one method or other is determined by the relief of the film surface obtained, which is dependent upon the initial viscosity of the lacquer. With high initial viscosity of the lacquer (approximately 1500 sec according to the VZ-4 viscosimeter), a layer of polyimide film of virtually the same thickness as the insulation layer forms on the interlayer junctions over the entire surface of the base. The film so formed is suitable for burnishing requirements, but is not suitable for undercutting. Use of a lacquer with initial viscosity of 300-400 sec produces a film with indistinctly expressed relief, which is more suitable for etching.

All operations of the technological cycle were performed on series produced equipment utilizing existing procedures and operational modes.

Thus the technological process for the manufacturing of BGIS for special applications includes sequentially executed operations of vacuum dusting metal plating of the first layer with further formation of the drawing using the method of photolithography, formation of the interlayer junctions on conductors of the first layer through a mask of positive photoresist using jumpers, removal of the jumpers, formation of the insulating layer, uncovering the interlayer junctions, and the formation of the second layer conductors.

Tests of 500 plates manufactured according to the technology outlined demonstrated excellent electrophysical parameters for interlayer insulation: shorts were lacking for pratical purposes, and specific capacity was rated at 2-3 pF/mm^2.

8851
CSO: 1863/142

UDC 621.38:776.002.5

CHARACTERISTICS OF MANUFACTURING PRINTED-CIRCUIT CARDS USING SPF-2 DRY FILM PHOTORESIST

Moscow PRIBORY I SISTEMY UPRAVLENIYA in Russian No 3, May 81 pp 41-42

[Article by engineers M. G. Miko and T. M. Marenkova and Candidate of Technical Sciences B. S. Irugov]

[Text] Intensive development of modern computer equipment and instrument building requires solution of a number of complex production problems and primarily development and serial assimilation of technological processes in producing precision and large printed-circuit cards. Specifically, cards with dimensions of more than 200 mm in which printed-circuit conductors and gaps 150-200 microns wide must be reliably formed are used in minicomputers of the SM EVM [International small computer system] and ASVT PS [expansion unknown].

Many years of experimental investigations conducted at the Institute of Control Problems (Moscow) and at a number of other organizations show that one of the most effective methods of solving this problem is to use SPF-2 dry film photoresist of type 2-40 or 2-60 and the corresponding highly productive production equipment.

Main Properties of SPF-2 Photoresist

Dry film photoresist (SPF) was especially developed to produce printed-circuit cards. It makes it possible to produce conductors with clear edges 150 microns wide and with the same gaps between them, whereas the minimum width of conductors and the gaps between them now produced comprises 250-380 microns for liquid photoresists. Investigations are now being carried out to improve the composition of the photopolymerized composition for SPF-2 photoresist, which will provide a further increase of its resolution.

Since SPF is much thicker than layers of liquid photoresist, electrochemical precipitation of the metal occurs inside the formed channel. For this reason there is no lateral fungus-like growth of the metal above the edges of the photoresist and consequently one can manufacture cards with more rigid tolerances on the width of conductors and gaps.

By using SPF-2 photoresist, one can produce high-quality two-sided printed-circuit cards by the positive image method with pre-drilling of holes with conductor width of 200 microns and gaps between them of 150 microns. The use of SPF also permits extensive introduction of the "screen" production process when the conductors and the copper in the holes are protected by photoresist during etching. However, this process is applicable only when the contact areas on the phototemplates are not cut.

The use of SPF-2 photoresist simplifies the production process (drying, tanning and retouching operations are eliminated), complete automation of the process becomes possible, it becomes more productive and permits a sharp increase of card quality and the percentage of yield of acceptable cards.

Due to the high chemical stability of SPF-2 photoresist in acid and alkaline solutions, there are no restrictions on the use of different galvanic baths and etching agents. Thus, the use of the same alkaline etching solution based on copper chlorate is possible for etching on SPF-2 photoresist (the "screen" method) and on metallic resist (tin-lead alloy).

It is important that the protective SPF now being widely developed to produce thermoresistant electric insulating coatings of printed-circuit cards are also developed in methylchloroform [12]. However, it should be noted that, as shown by operating experience, failure to observe the necessary requirements when using SPF significantly reduces its advantages over liquid photoresists.

Requirements on Production Buildings

Since SPF is protected by a lavsan film prior to developing, it is less sensitive to dust than liquid photoresists. However, to completely eliminate retouching, the entire process using SPF must be carried out under "clean room" conditions: the incoming air should be filtered and excess pressure should be maintained in the room.

The temperature should be in the range of 20-25°C and relative humidity should comprise 50 $\pm$ 10 percent in the section where operations of lamination and exposure of the photoresist are being carried out. Lamination, exposure and development should be carried out in yellow light to prevent preliminary polymerization.

Requirements on Phototemplates

The large thickness of the photopolymerized composite in the presence of a lavsan film during exposure place more rigid requirements on phototemplates. The optical density of the opaque sections of the image should not be below three units of density (otherwise partial polymerization on the protected sections occurs), while that on the transparent sections should not be more than 0.1 units of density (otherwise insufficient polymerization of the exposed sections is observed).

A clear, contrast image on the phototemplates is required and a corona and washout of lines are not permitted.

The slides should have a dark mask around the perimeter of the phototemplate (the minimum distance from the circuit pattern to the mask is not less than 10 mm), which creates the best conditions during development and contributes to more uniform galvanic buildup of the metal.

Centering points in the middle of the contact surfaces are not permitted in the phototemplates. The diameters of the contact surfaces should be 1.5-fold larger than the hole diameter to protect the metallized holes during the "screen" method.

Surface Preparation

Blanks with pre-drilled holes and which have undergone chemical metallization of the holes with galvanic prolongation by copper to a thickness of 5 microns prior to lamination should be dried in a drying cabinet with air recirculation for 20-30 minutes at t = 80-100°C to remove moisture from the holes. The blanks with the galvanic coating are subjected to additional mechanical cleaning.

It is necessary that the blanks have an even surface without deep scratches and pits, that the edges be even without outgrowths and that no drilling burrs be in the holes (drilling is accomplished without counterboring).

After cleaning, the copper surface should be slightly mat. The best adhesion of the SPF to it is provided by mechanical cleaning, but the prepared surface should not be polished.

It is required that a thoroughly cleaned blank hold a water film in the vertical position for approximately 15-20 seconds over the entire surface. Since the cleaned surface is easily oxidized, lamination is accomplished no later than 1 hour after surface preparation.

Lamination

The SPF-2 photoresist is laminated on an installation for application of dry film photoresist to both sides of a clean dry blank with the rollers being strictly parallel and at a pressure of 0.5 kgf/cm. The temperature of the heating shoes should be maintained in the range of 115 $\pm$ 5°C over the entire length of the shoes and should be monitored constantly to avoid polymerization of the SPF-2 in case of overheating. Poor adhesion of the SPF to the blank is observed if the temperature decreases to a value below that indicated. Nevertheless the negative effect of temperature deviation from the given range can sometimes be compensated for by changing the rate of lamination. The latter is selected experimentally for each lot of SPF and is in the range of 0.7-1 m/min.

The heating shoes and the laminating rollers should be completely clean, without photoresist residues on them, which are removed by methylchloroform. Methylene chloride must not be used for this since not only the methylene chloride vapors affect the SPF-2 photoresist, but they also break down the roller material. The shoes and rollers may not be cleaned mechanically since low-quality lamination which leads to retouching or rejection during subsequent processing is observed if they are damaged.

Lamination is usually carried out on cold blanks. However, if the SPF-2 photoresist is used for a "screen" or if it becomes subject to the effect of high temperatures during subsequent production operations, lamination must be carried out on hot blanks (80-100°C). To increase the adhesion of the photoresist to the blank, the latter is aged for not less than 30 minutes after lamination prior to the next operation.

SPF-2 photoresist of type 2-60 or 2-40, laminated two times, is used to produce a photopolymer layer of great thickness while providing the "screen" process on blanks 1.5 mm thick with metallized hole diameter of 0.8-1.2 mm. The blank is aged for 30 minutes prior to the second lamination. The lavsan film which provides mechanical protection and serves as a barrier against the effect of atmospheric oxygen on the photopolymer layer, is removed immediately prior to developing or during the second lamination.

Exposure

SPF-2 dry film photoresist is a negative material: it is polymerized on the exposed section and it becomes insoluble in metal chloroform.

SPF-2 photoresist has a thick photopolymer layer and is exposed through the protective lavsan film. Exposure is carried out in an exposure installation with collimated movable light source--flash lamps. Air-evacuating gaskets are used inside the vacuum frame for rapid removal of air and to provide good contact of a phototemplate and the SPF. Heating in the vacuum frame is not permitted since it increases the rate of polymerization of the photoresist and leads to variation of the optimum exposure time, which is selected experimentally and is periodically checked: after replacement of the lamps, prior to the use of a new lot of SPF and after replacement of the film in the vacuum frame. Determination of the optimum exposure time is also important because the electrolytes of the galvanic bath are rapidly contaminated by organic impurities with short exposure and accordingly with insufficient polymerization of the exposed SPF.

A test-card phototemplate on which the width of the conductors is measured at various points on its entire surface is used to determine the optimum exposure time. The blanks with the applied SPF-2 photoresist are exposed thorugh a test-plate negative with different speed of the flash lamp carriage. A blank with brilliant hard layer of photoresist and with even unwashed edges of the conductors is selected after development. The measurements are made at the same points as on the template on these blanks by using a microscope. The exposure time at which conductors with clear edges and minimum deviation of their dimensions toward the positive side are produced is optimum.

The time during which the conductor cross-sections on polished sections have plumb edges after exposure, development and galvanic buildup of metal will be optimum for the positive image method of printed-circuit card manufacture.

Development

Development of SPF-2 photoresist is based on dissolution and washout of the photoresist layer not polymerized during exposure in methylchloroform. The developing installation provides rapid and complete removal of this layer under a stream of sprayed solvent. The developing time depends on the thickness of the light-sensitive layer, on the temperature of the developer which should be monitored and should be in the range of 15-20°C and on the pressure at which the solvent is fed to the blank.

The optimum developing time is selected experimentally and is taken as 1.5 times greater than the time required to remove the unexposed layer of photoresist. The optimum developing time comprises 40 seconds for SPF-2 photoresist of type 2-40 at pressure of approximately $2 \cdot 10^5$ Pa at the nozzles and at solvent temperature of 16-18°C. The photoresist is washed out during development of the positive image from deep channels; therefore, the developing time is increased by 5-15 seconds depending on the density of the circuit pattern and the width of the conductors.

The developing time, temperature of the developing solvent, the purity of the developer and final flushing with cold water affect the quality of the developed image. Complete washing of the unexposed photoresist is necessary since even traces of it during subsequent operations lead to rejection.

The contaminated solvent is partially broken down during purification (distillation) while the impurities and products of methylchloroform decomposition affect the exposed photoresist and cause a reduction of the range of developing time. Therefore, no more than two parts of the distilled solvent and one part of purified solvent must be used in the developing chamber, while only pure solvent must be used in the washing chamber.

The solvent is contaminated not only by the dissolved photoresist and impurities but also by water whose presence threatens low-quality developing.

Rapid careful washing of the pattern by a stream of cold water under pressure is required after developing to remove residues of solvent before the developer begins to evaporate, after which the blank with developed image is protected by fine polishing powder with subsequent flushing with water. Becoming greasy--the unwettability of the copper surface on blank sections--occurs with low-quality developing. A well-developed image should support a continuous film of water for 20-30 seconds. Introduction of regulators to the photopolymerizing composite which facilitate the developing process is possible with further improvement of SPF-2 photoresist.

Not only the chemical action of the solvent but also mechanical action, which is provided by spraying methylene chloride under pressure at no less than $3 \cdot 10^5$ Pa in installations to remove the SPF are required for complete removal of the photoresist from the exposed sections.

But since methylene chloride has a negative effect on the dielectric properties of the base after etching the copper from the blank spaces of the card, the removal time should be minimum. Thus, the removal time for SPF-2 photoresist of type 2-40 at temperature of the removing solution of 16°C and at pressure of $3 \cdot 10^5$ Pa comprises 25-30 seconds. The pattern is carefully flushed with water after removal of the photoresist with subsequent cleaning of the surface with polishing powder and flushing.

The use of dry photoresists as a base to produce saturated conducting patterns on large printed-circuit cards makes it possible to solve the most important production problems in manufacture of modern microelectronic apparatus and computer equipment. This conclusion has also been confirmed by the experience of leading foreign and domestic firms.

Extensive development of these materials is a necessary condition to convert to more promising positive-image and combination methods of manufacturing two-sided and specifically of multi-layer printed-circuit cards. Dry photoresists find wide application not only for subtractive but also for additive (semiadditive) techniques.

Preliminary investigations also show that the production processes using dry photoresists are more easily subject to automation and find broad application in modern ASU TP [Automated production process control system] of printed-circuit cards.

BIBLIOGRPAHY

1. Fedilova, A. A., Ye. P. Kotov and E. R. Yarvich, "Mnogosloynyye pechatnyye platy" [Multilayer printed-circuit cards], Moscow, Sovetskoye radio, 1977.

2. Kuznetsov, V. N., V. A. Pogost et al, "Some Characteristics of Using Dry Film Photoresists Developed by Organic Solvents and By Water-Alkaline Solutions," VOPROSY RADIOELEKTRONIKI. TEKHNOLOGIYA PROIZVODSTVA I OBORUDOVANIYE, No 3, 1978.

3. Information Manual. Riston Photopolymer Resist, DuPont, 1972.

6521
CSO: 1863/155

MULTIMACHINE AND MULTIPROCESSOR SYSTEMS

Moscow NOVOYE V ZHIZNI, NAUKE, TEKHNIKE: SERIYA "RADIOELEKTRONIKA I SVYAZ'": EVOLYUTSIYA VYCHISLITEL'NYKH SISTEM in Russian No 3, Mar 81 (signed to press 12 Feb 81) pp 44-45, 62

[Excerpts from chapter 6, "Multimachine and multiprocessor systems," and conclusion from the monthly "Radioelectronics and Communications" serial, "What's New in Life, Science and Technology": "Evolution of Computer Systems", by Yevgeniy Pavlovich Balashov, doctor of engineering science, professor, author of 12 books and 96 inventions, who specializes in computer technology; and Arkadiy Petrovich Chastikov, candidate of engineering science, docent, author of over 40 scientific articles and many inventions, Izdatel'stvo "Znaniye", 38,260 copies, 64 pages]

[Excerpts] Multimachine and Multiprocessor Systems

In 1976, associates at the Institute of Mathematics, Siberian Branch of the USSR Academy of Sciences together with the Severodonetsk Impul's Scientific Production Association engineered the MINIMAKS project, a minimachine program-switched system. MINIMAKS is a homogeneous system with program-switched channels for communication between the elementary machines (EM). Each elementary machine consists of a computer complex (M-6000 or M-7000) to process data and a systems device for communication between the elementary machines. It is interesting to note that the ratio of cost of the systems device and the cost of the MINIMAKS processor does not exceed 0.5, while this ratio for multiprocessor systems with a common bus with the RDR-11 computer processor is 1.2.

The MINIMAKS system may function autonomously, as part of high-capacity concentrated computer complexes or as part of distributed computer networks. In the process, it may be successfully used both to solve scientific and economic problems, and to control processes.

Among subsequent developments of homogeneous computer systems, we must mention the minimachine system (SUMMA) developed in 1977 and based on the Elektronika 100 and Elektronika 100I domestic minicomputers, a system oriented to operation as part of automated systems for control of industrial processes. The Institute of Mathematics, Siberian Branch of the USSR Academy of Sciences, and the NITsEVT [Scientific Research Center for Electronic Computer Equipment] have now engineered a project for a general-purpose homogeneous computer system based on the YeS-1060 computer. Its overall speed is determined by the number of YeS-2060 processors and by ratings may be from 2 to 20 million operations per second.

The development and production in our country of homogeneous computer systems is a significant achievement in the field of building multimachine computer facilities with a programmable structure.

Conclusion

In the development of domestic computer technology, a great contribution was made by academicians S. A. Lebedev, V. M. Glushkov, A. A. Dorodnitsyn, G. I. Marchuk, I. V. Prangishvili, V. S. Semenikhin, corresponding members of the USSR Academy of Sciences I. S. Bruk, V. S. Burtsev, Yu. A. Bazilevskiy, G. P. Lopato, A. A. Lyapunov, A. P. Yershov, N. Ya. Matyukhin, B. N. Naumov, doctors of engineering science E. V. Yevreinov, M. A. Kartsev, A. M. Larionov, V. K. Levin, B. N. Malinovskiy, V. V. Przhiyalkovskiy, B. I. Rameyev, M. R. Shura-Bura and other researchers, designers and developers of data processing facilities.

Developers are now faced with a number of problems, among which the following may be mentioned: formation of new computer series, rational combination of hardware and software in computer systems, formation of advanced structures based on large-scale integrated circuits, further improvement of the design and technology base, etc.

It should be noted that the socialist countries have found a successful path in the field of developing computer technology, development of a second series of YeS [unified system] computers with a throughput of millions of operations per second is nearing completion successfully, and the high-throughput computer systems El'brus-1 and El'brus-2, M-10, YeS-1035 and Yes-1045 with matrix processors have been developed.

COPYRIGHT: Izdatel'stvo "Znaniye", 1981

8545
CSO: 1863/162

BASIC CHARACTERISTICS OF SOME DOMESTIC LSI MICROPROCESSOR COMPLEXES

Moscow NOVOYE V ZHIZNI, NAUKE, TEKHNIKE: SERIYA "RADIOELEKTRONIKA I SVYAZ'": EVOLYUTSIYA VYCHISLITEL'NYKH SISTEM in Russian No 3, Mar 81 (signed to press 12 Feb 81) p 55

[Table from chapter 7, "Evolution of Microprocessor Technology," from the monthly "Radioelectronics and Communications" serial, "What's New in Life, Science and Technology": "Evolution of Computer Systems", by Yevgeniy Pavlovich Balashov, doctor of engineering science, professor, author of 12 books and 96 inventions, who specializes in computer technology; and Arkadiy Petrovich Chastikov, candidate of engineering science, docent, author of over 40 scientific articles and many inventions, Izdatel'stvo "Znaniye", 38,260 copies, 64 pages]

[Excerpt] Basic Characteristics of Some Domestic LSI Microprocessor Complexes

Circuit Type	Function	Technology	Word Size, Bits	Speed, μs	Number of Instructions (Micro-Instructions)	Supply Voltage, V
K-536 (14 LSI)	Various	PMOS	8	10.0	149	-24; +1.5
K580IK80	Microprocessor	NMOS	8	2.0	78	+5; +12; -5
K581IK1	Register ALU	NMOS	16	1.6	--	+5; +12; -5
K581RU1	Microprogram Storage	NMOS	12	--	512	+5; +12; -5
K582IK1	Processor Element	I^2L	4	1.5	--	1.2
K-583 (13 LSI)	Various	I^2L	8	1.0	--	--
K584IK1	Microprocessor	I^2L	4	1.5	512	1.2
K-586 (3 LSI)	Various	NMOS	16	0.5	--	--
K587IK2	ALU	CMOS	4	2.0	168	9
K587RP1	Control Memory	CMOS	--	1.5	--	9
K587IK1	Data Exchange	CMOS	8	1.5	60	9
K587IK3	Arithmetic Expander	CMOS	8	2.0	64	9
564RU2A	256-bit Main Memory	CMOS	1	2.5	--	3-15
564RU2B	256-bit Main Memory	CMOS	1	3.5	--	3-15
K530AP2	Bidirectional Amplifier-Driver	TTLDSh	4	0.03	--	--

COPYRIGHT: Izdatel'stvo "Znaniye", 1981

8545
CSO: 1863/162

P320 PROGRAMMABLE CALIBRATOR

Moscow PRIBORY I SISTEMY UPRAVLENIYA in Russian No 3, Mar 81 p 37

[Text] The P320 calibrator with manual and program control is designed for use in automated checking installations and also as an independent device for checking analog and digital devices operating on direct current. The device is a desk type, portable and made from the designs of the ASET [expansion unknown].

The calibrator provides for transmission of calibrated voltages and currents in the range from $1 \cdot 10^{-5}$ to $1.1 \cdot 10^{3}$ volts and from 10^{-9} to $1.1 \cdot 10^{-1}$ A, respectively, and remote (program) control, including setting the maximum and the level of the output parameter. The level of the output parameter is controlled in binary-digital codes 8-4-2-1.

Maximum Calibrated Voltages (Currents)		Maximum Errors of Relative Value of Calibrated Voltages (Currents)
0.1 V		$5 \cdot 10^{-5}$ U_k + 10 μkV
1 V		$2 \cdot 10^{-5}$ U_k + 10 μkV
10 V		$1 \cdot 10^{-5}$ U_k + 40 μkV
100 V		$3 \cdot 10^{-5}$ U_k + 500 μkV
1,000 V	Up to 600 V	$4 \cdot 10^{-5}$ U_k + 5 mV
	Above 600 V	$5 \cdot 10^{-5}$ U_k + 5 mV
1 mA		$2 \cdot 10^{-5}$ I_k + 0.01 μkA
10 mA		$5 \cdot 10^{-5}$ I_k + 0.1 μkA
100 mA		$5 \cdot 10^{-5}$ I_k + 1 μkA

Note. U_k and I_k are the established value of calibrated voltage and current.

The maximum permissible main error of the relative value of calibrated voltages and currents does not exceed the values indicated in the table.

The main error is numbered for the temperature range of $t_n \pm 2$°C, where t_n is the temperature at which calibration was made. The output impedance of the calibrator is in the range of 10 V $\leq$ 0.001 ohms. The level of the variable components in the frequency band up to 100 kHz (a mean square value) at the calibrator output $\leq$ 300 (for range of 0.1-100 V) and $\leq$ 1,000 μV (in the range of 1,000 V). The calibrator has an overload protection device and limiter of the input voltage (current) level. The power consumed by the device at nominal mains voltage $\leq$ 150 V·A. The mains voltage is 220 $\pm$ 22 V (at frequency of 50 $\pm$ 0.5 Hz). The overall dimensions and mass are 488 X 535 X 250 mm and 26 kg. The device is operated at 10-35°C and at relative humidity up to 80 percent.

Calibrators are delivered by order to Soyuzglavpribor (117219, Moscow, V-219, ul. Krzhizhanovskiy, 16).

The incorrectness of the data presented in the article of V. V. Korchin and V. T. Kuz'min NTTM-80, published in issue No 10 of our journal for 1980, is the operating reason for publication of the given remarks.

6521
CSO: 1863/155

NEW COMPUTER HARDWARE

Moscow PRIBORY I SISTEMY UPRAVLENIYA in Russian No 1, Jan 81 pp 44-45

[Article by S.B. Abramov, candidate of economic sciences: "New Hardware"]

[Text] The USSR VDNKh (USSR Economic Achievements Exposition) Pavilion, "Computer Equipment", featuring theme expositions "Advanced Knowhow of the Kazan Computer Plant in Perfecting Series Production of Computer Hardware" and "Mechanization and Automation of Computer Operations" demonstrated new computer hardware manufactured by enterprises of the Ministry of Instrument Building, Automation Equipment, and Control Systems.

The article provides a brief description of the more interesting exhibits, noted for the novel design-engineering solutions incorporated, and the high level of technico-operational parameters displayed.

A desk-top terminal featuring simplest information processing, the "Iskra-900" is designed for the arithmetic processing of data, timekeeping, automatic telephone switching-connecting, simultaneous conversation among three members-subscribers of the telephone system, and exchange of digital information with the computer. Including the indicated functions in the terminal precludes the need for a group of single-function instruments, (keyboard computer, clock-calendar, auto-dial telephone, teletype), and reduces the number of operations involved in data preparation for input into the computer.

The terminal can be employed by upper and middle-management links in institutions and enterprises from various areas of the national economy, and by operators involved in data preparation at remote peripheral points of the ASU and directly by production personnel. It is used independently and in conjunction with the M-6000 and SM computers.

From a design standpoint, the "Iskra-900" consists of a control console and interface unit. The control console provides for executing the functions of the EKVM (keyboard computer), the clock-calendar, auto-dial telephone, and also, in conjunction with the interface unit, four terminal operational modes (EKVM, auto-dial telephone, clock-calendar, terminal).

Specifically, in the "EKVM" mode, the terminal performs arithmetic operations, chain-catenary operations, and operations with constants with allowance for the result character of the executed operation and overflow of the machine binary word format.

In the "Terminal" mode, the machine provides for the transmission to the computer of data output from the keyboard, selected from the operational memory, EKVM computation results, current time, and also the receipt of data from the computer.

Transmitted and received data are indicated on the terminal's indicator display.

The electronic ticket counter outlet (EBKM), "Iskra-307", is designed for mechanization and accounting of ticket-outlet operations for passenger traffic on railroad transportation facilities.

Incorporating integrated circuits (IS) with low level integration of the K155 series, the machine consists of a calculator, tape perforation unit, and a printer. Input of data is accomplished via an alphanumeric keyboard on the calculator and minicassettes, with output to the ELT screen (during the process of ticket registration), printer (travel document, control tape), and tape perforator.

The "Iskra-307" performs the following operations: accounting of monetary receipts in a overall summation register, and cashier total receipts per shift; an accounting of the number of travel documents issued by category; an accounting of the cost and number of voided tickets; and the automatic price determination for various types of travel tickets.

The machine "Iskra-307", in comparison with earlier used machines of the same class of electromechanical machines has expanded functional capabilities: recording a large volume of data on a machine medium, with subsequent computer data processing, lesser dimensional size, and lower weight.

Technical Characteristics of the "Iskra-307" Computerized Ticket Outlet (EBKM)

Number of counters:	
summary (monetary)	6
operational	1
Volume of counters in decimal places	
monetary	6
operational	4
Volumes (capacities):	
OZU, bits	4K X 8
MOZU, bits	1024
PZU, bits	8K X 16
ELT, symbols	256
VOLTAGE, volts	220 (50 Hz)

The electronic control-registration machine (EKRM) "Iskra-300" performs cash accounting operations in trade and public catering enterprises.

The machine incorporates large scale IS and a microprocessor. Data input is from an alphanumeric keyboard, output to a seven-digit built-in indicator and to a printing device (control check, tape, detachable check, accounting register). Two documents are simultaneously printed out—the check or statement, and the control tape.

The "Iskra-300" performs the following operations: accounting, individual total, change, multiplication, addition, subtraction, repeat, and void.

Use of the machine increases the productivity of accounting purchases with consumers, as the cost of purchases is calculated by price, weight, or number.

Technical Characteristics of the "Iskra-300" EKRM

Information volume, OZU in bits	256 X 4
Printing speed, lines per second	2.5
Control registers	4-digit
Number of character positions per line	16
Power required, in Watts, not more than	200
Dimensions, in mm	486X405X400
Weight, in kilograms, not more than	40

Centralized control machine (MTsK), Series M-40-43, is designed to solve control and management problems for technological processes in various industrial sectors and constitutes the latest development of Series M-40 machines, which are based upon microprogrammed control.

The Series M-40-43 machine is employed in enterprises with continuous and discrete type production (enterprises with ASU controlled lifting-transporting machinery, control of metal quality for hot and cold rolling, control and management of individual functional groups of power units, small and medium substations, and others).

The machine is produced in four versions (see table). The first two versions are used by enterprises with continuous and discrete production, the latter two are used only by enterprises with distinct type production.

The basic set of machines includes a central control unit (UUTs), also an input-output unit, technological data print unit, input-output of discrete data unit, telemechanical communications (UTS), external memory (UVP-43), alphanumerical terminal, and an indication console.

Modification	Number of high level signals	Maximum number of module points of discrete		Capacity of external ZU, Kbits
		input	output	
M-40-43/2	128	352/22	160/16	16
M-40-43/21				--
M-40-43/3	—	1024/64	320/32	16
M-40-43/31				--

By design version, the MTsK of the M-40-43 series belongs to the ASVT-M system. Communication with external devices is accomplished by a 2K communication rank.

Technical Characteristics of the MTsK M-40-43

Maximum number of external units (peripherals) connected to:	
UUTs	up to 16
UUTs via UTS	up to 28
Maximum number:	
High level analog input signals	up to 512
Discrete input data points	up to 1400
Maximum speed for execution of microcommands in thousands of operations per second	500
Capacity in bites	
Random access file (UVP-43)	16K
Memory unit:	
Current	2K
Permanent	8K
Voltage, volts	320/220 (50 Hz)
Required power, in V.A.(volt-amperes)	3500
Full operating time in hours, not less than	1000
Dimensions in mm	650X678X1600
Weight, kilograms	2000

The SPK-80 aperture punchcard search selector is designed for the rapid search of data input in the form of perforations on standard 80-column punchcards, and is employed in searches for patents and inventors' certificates, information in technical information offices, and for design analogs for the use of ready solutions or the study of existing technical levels of tasks solved, in personnel departments and other institutions.

The card feed and photoelectric method of reading of data provides for a significant reduction in the wear of punchcards, an extension of their service life, and the single-phase feeding net, low weight, and small dimensions and low noise level permit the use of the selector in any organic equipment complex, with ASU or autonomously in any sub-element where it is necessary to select from a deck of cards with the required data.

The selector selects punchcards with total coincidence of information on the selected punchcard with information input to memory, the coincidence of information of one of the zones of the designated punchcard with information of the zone input to memory, and the coincidence of information from a given column of the selected punchcard even if only one of the characters input to memory for that same column.

Technical Characteristics of the SPK-80

Technical speed in cards/min	400
Capacity of magazines in cards:	
receiving	600
feeding	400
Number of simultaneously read positions	12
Voltage, in volts	220 (50 Hz)
Required power, in V.A	500
Dimensions in mm	900X560X460
Weight in kg	100

COPYRIGHT: Izdatel'stvo "Mashinostroyeniye". "Pribory i sistemy upravleniya", 1981

8851
CSO: 1863/142

SOFTWARE

UDC 681.328

SYSTEM FOR COMPLEX DEBUGGING OF ELEKTRONIKA S5-12 MICROCOMPUTER BASED ON FUNCTIONAL MICROPROCESSOR MODULES

Moscow PRIBORY I SISTEMY UPRAVLENIYA in Russian No 3, Mar 81 pp 36-37

[Article by Candidate of Technical Sciences V. V. Sumin and engineers V. N. Syrtsev, A. B. Vasil'yev and N. M. Kalinin]

[Text] Single-board microcomputers of the Elektronika S5 family are now a promising technical base for construction of various automated systems. It is very important in this case for the user to have the capability of complex debugging of the equipment and which is more timely the program parts of the system under development.

One of the variants of constructing a debugging system for the Elektronika S5-12 microcomputer on the basis of functional microprocessor modules serially produced by our industry is described in the article. The complex debugging system is designed for preparation, editing and debugging of programs for systems constructed on the basis of the Elektronika S5-12 microcomputer. The system permits entry of information from a type FS 1501 photoreading device, program editing by means of keyboard and console machine of the Consul-260 type, derivation of information to the PL-20 printer and light diode displays and listing of the programs being debugged can be achieved if necessary.

The debugging system is capable of successfully replacing the industrial prototype system constructed, for example, on the basis of the multiboard Elektronika S5-02 microcomputer (for a specific class of problems).

A block diagram of the debugging system OS is presented in Figure 1. The system includes a microcomputer of the Elektronika S5-12 type, functional microprocessor modules OZU [Internal storage] Elektronika S5-125 with capacity of 2,048 X 16 bits and TsVV [Digital input-output] Elektronika S5-122 (it has four each digital inputs and outputs), read-only memory PZU with capacity of 256 X 16 bits, matching devices SU, light-emitting diode displays SID, peripheral devices (electronic console machine EPM of the Consul-260 type, perforator P of the PL-20 type and photoreading device FU of the FS 1501 type) and a keyboard Kl.

Without describing in detail the organization and information exchange in the system, let us note only some of its features. Thus, the use of the Elektronika S5-122 TsVV for exchange of the debugging system with the periphery permits

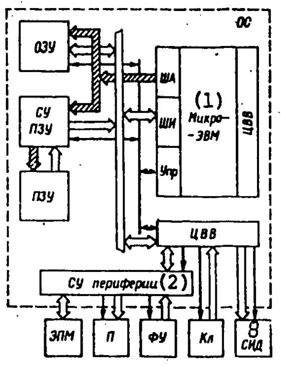


Figure 1. Block Diagram of Debugging System: ShA, ShI and Upr--address, information and control buses

Key:
1. Microcomputer
2. Peripheral switching devices

preservation of the digital inputs-outputs by the microcomputer itself for working with the automation facility. The use of the Elektronika S5-125 OZU module with capacity of 2K (or 4K) permits storage of user working programs and the program for exchange with the console machine. The read-only memory PZU with capacity of 256 words is realized on a micro-OZU of type 561RU2 with buffer power supply. It consumes less than 20 μA in the storage mode; the information is retained for 5,000 hours after cutoff of the power supply. The booster storage battery is precharged automatically when the system power supply is switched on. A dispatcher program which services the work of the system with keyboard Kl, SID display block, perforator P and FS 1501 transmitter is written into the PZU device. The keyboard of the debugging system consists of keys of the operating code ("Writing of initial address," "Writing by address," "Reading by address" and "Function") and of a keyboard of a 16-digit symbol field. The "Function" operation is introduced to expand the capabilities of the system without additional apparatus expenditures. Check addition, file shift, initial start of program from given address, entry from the photoreading device, printout of information on punch tape and conversion to program of working with Consul-260 machine can be accomplished by using this operation. Specific functions are introduced by means of the keys of the 16-digit keyboard field.

When realizing the complex debugging system, the greatest difficulties may be encountered in organizing the integration of the Elektronika S5-122 TsVV module with the peripheral devices. One of the versions of this organization is presented in Figures 2.

The Elektronika S5-12 microcomputer continuously interrogates the keyboard of the complex (two keyboard racks are related to a 16-digit code and one is related to

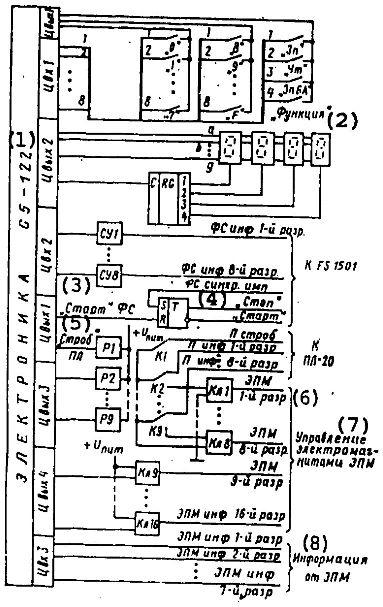


Figure 2. Diagram of Integration of TsVV Module and Periphery: Tsvykh1-Tsvykh4--digital outputs 1-4 of TsVV module; Tsvkh1-Tsvkh3--digital inputs 1-3 of TsVV module; ZP, Cht, ZpBA and Function--operating codes "Writing by address," "Reading by address," "Writing of initial address" and "Function": a, b, ..., g--light-emitting diode display buses; SU1-SU8--matching devices in photoreader circuit; R1-R9--encapsulated solid-state switches of control of perforator and selection of decoder column of EPM machine; K1-K9--encapsulated solid-state switch contacts; K11-K116--case for selecting the decoder column of EPM machine; RG--recorder-distributor; T--flip-flop for control of FS 1501 transmitter operation; EPM inf 1r-EPM inf 7r--information from EPM encoder; FS--photoreading device; P--perforator

Key:

1. Elektronika S5-122	5. Strobe
2. Function	6. Digit
3. Start	7. Control of EPM electromagnets
4. Stop	8. Information from EPM

the operating code) through the digital output Tsvykh1 of the Elektronika S5-122 module. If not a single key is pressed at a given moment, the microcomputer accomplishes dynamic display of the contents of the display buffer in 16-digit code through digital output Tsvykh2.

Digital output Tsvykh3 of the TsVV module is used to connect the perforator to the debugging system. The form of retrieving the information is determined by the subprogram for working with the perforator.

Information from the FS1501 transmitter through the matching devices is entered through digital input Tsvkh2 of the module in parallel code; the form of the entered information is determined by the subprogram.

The status of the console machine is interrogated through digital input Tsvkh3. The code for control of the corresponding electromagnets of the printing mechanism is transmitted through digital outputs Tsvykh3 and Tsvykh4 of the module as a function of the type of control code. If a code comes in that corresponds to a 16-digit symbol, the microcomputer prints this symbol and enters it in the information buffer after conversion.

The described debugging system permits one to realize two important debugging functions in development of automated systems based on single-board microcomputers of the Elektronika S5 family which are determined by the location of the microcomputer. If the latter is located in the system to be debugged (a microcomputer is then unnecessary in the debugging complex), the periphery of the debugging complex works with the user program and permits complete debugging. If the microcomputer is contained in the debugging complex (the system to be debugged may then have no microcomputer), the periphery of the complex is connected by the microcomputer to the facility equipment and permits debugging of the relationship of the facility and microcomputer.

Finally, let us note two additional circumstances related to the described system. First, the use of serially produced Elektronika S5-122 and Elektronika S5-125 functional microprocessor modules, hardware- and software-compatible with the Elektronika S5-12 microcomputer, permits significant facilitation in realization of the debugging system and completion of it without some significant circuit developments. Second, although the described system assumes the presence of a rather developed periphery, debugging can also be carried out in minimum peripheral configuration, for example, by using only a keyboard and light-emitting diode displays.

The described system replaces the scarce prototype systems based on multiboard models of Elektronika S5-02 type for most practical purposes.

6521
CSO: 1863/155

UDC 658.012.011.56:002.53.001.12

MAIN TRENDS IN FORMATION OF AN ALGORITHM AND PROGRAM FUND FOR OASU

Moscow PRIBORY I SISTEMY UPRAVLENIYA in Russian No 3, Mar 81 p 5

[Article by engineer S. M. Khovanskaya]

[Text] An increase of the efficiency of introduced OASU [Automated sector control system] and their effect on the improvement of the technical and economic operating indicators of ministry and agency subdivisions can be achieved by standardization of OASU components and automation of its design. The problem of standardization includes formation of a system which satisfies the specifics of a given facility from standard components. The use of methods of standardization in design of OASU permits a reduction in the cost of developments, a reduction of the periods of systems development and a reduction in the number of OASU developers.

Standardization of the control system components in our country is being developed in three directions: development of standard ASU [Automated control system], unification and standardization of individual parts of functional and support subsystems and construction of ASU by use of applied program packs (PPP).

Practice shows that development of standard ASU creates large complexities and is economically unfeasible since there are no completely identical ministries in nature. Consequently, two methods remain: unification and standardization of the individual parts of an OASU and design of OASU using PPP. This problem can be solved through extensive use of a centralized algorithm and program fund (TsFAP).

The PPP now available in TsFAP and oriented toward use in OASU can be divided into four groups by designation.

1. Means of organization and management of the information base of the OASU.

2. Functional subsystems.

3. Means of programming automation.

4. Means of organizing the calculating process.

The "Glossary organization and managment" PPP, the "Driada" data bank, the "Accounting" PPP and the "Data" PPP belong to the first group.

OASU subsystems developed within the ASU-pribor system are related to functional subsystems. These subsystems can be utilized to the maximum in design of OASU of machine building. Analysis of the possibility of using subsystems in other sector groups (the light, local and mining industries) showed that it is impossible to introduce programs available in the fund without modifications. The circumstance that conversion to the two- and three-section structure of management has still not been completed in the ministries until now also creates great difficulty in use of the programs.

The "Compiler" PPP and the "Monitor" PPP are included in the group of PPP that automate the process of program development and debugging.

An important problem is development of an effective technology of design with extensive use of the algorithm and program fund. One of the forms of automation of OASU design may be development of a so-called systems fund by groups of sectors which, unlike existing funds, should be a unified organizational system.

The following facilities may be included in the systems fund: general-purpose PPP to which PPP that realize development and management of glossaries and files of normative and reference information are related, development and management of data bases of current and archival information and printout of output forms and display of information; functional PPP that include those which enter and monitor accounting information as it comes in with regard to functional features and calculation of the output indicators of PPP for organization and management of the calculating process; and a system for teaching how to work with PPP.

The developers of OASU using the systems fund will be able to realize specific management functions. In this case the degree of encompassing OASU problems with facilities of the systems fund may be close to 100 percent.

Therefore, formation of an algorithm and program fund for OASU should be planned and purposeful. There should be no PPP in the fund that do not have a high degree of universality with respect to specific functional solutions and that do not have means of adaptation to a specific medium of use by adjustment for given parameters.

The need to meet the named requirements generates yet another problem: development of specific tests that check not only the functional but also the operational capabilities of PPP.

Investigations are now being conducted by a number of scientific research institutes (VNIPI [All-Union Scientific Research and Planning Institute] of OASU, Moscow, VNIIugol' [All-Union Scientific Research Institute of Coal], Moscow, NPO [Scientific Production Association] Pishchepromavtomatika, Odessa, and so on) to develop standard software for OASU for its own sector groups, which will undoubtedly serve as the basis for organization of systems funds of OASU.

6521
CSO: 1863/155

UDC 519.6

COMPARATIVE COURSE OF THE PL/1 LANGUAGE (ON THE BASIS OF ALGOL-60)

Moscow SRAVNITEL'NYY KURS YAZYKA PL/1 (NA OSNOVE ALGOLA-60) in Russian 1980 (signed to press 17 Apr 80) pp 6-8

[Foreword from book "Comparative Course of the PL/1 Language (On the Basis of ALGOL-60)" by Yuriy Mikhaylovich Bezborodov, in the series "Bibliotechka Programmista", Main Editorial Board of Physical and Mathematical Literature, Izdatel'stvo "Nauka", 65,000 copies, 192 pages]

[Text] In the last 15 years the main, and at the beginning even the only algorithmic language in the Soviet Union was ALGOL-60, used both for teaching the principles of programming and for solving computational problems on computers. In recent years, however, after the appearance of YeS computers, PL/1 (Programming Language), which also is an algorithmic language, has become more and more widely distributed. In contrast with ALGOL-60, which is intended for the solution of problems of numerical analysis, PL/1 is a universal language which permits, in addition, formulating solutions of commercial problems, problems in processing symbolic and binary information and ASU problems. It can be considered, in particular, that PL/1 combines the possibilities of such widespread languages as ALGOL, FORTRAN and COBOL. On YeS machines, PL/1 replaces ALGOL in practical work, displacing it, which can be explained not only by the universality of PL/1 and its other merits, but also by major inadequacies of the translator from ALGOL-60 in the YeS computer (ineffectiveness of the translator, inconveniences of the input language and especially of the input/output operators, the impossibility of coupling with other algorithmic languages and the absence of packages of applied programs).

The present course of the PL/1 language is intended to ease for programmers the transition in their work from other computers to YeS computers and is primarily designed for programmers who have experience in programming in ALGOL-60 and want to become acquainted in a short time with the main possibilities of PL/1 and to learn to program in that language for YeS computer operating systems. In the course an attempt has been made in teaching programming in PL/1 to use in a very essential manner the skill and experience in programming in ALGOL which the readers already have. True, besides practical experience, acquaintance with the terminology and syntax of ALGOL-60 also are required of the reader in that case [1,2].

In Chapter 1 of the course the concepts of ALGOL-60 and PL/1 are compared in such a way as to give the reader a conception about the creation and differences in the designs of ALGOL-60 (in the standard version) and PL/1 (for YeS computer operating systems) which have the same functional possibilities. The used method of comparison will enable the reader to rapidly start compiling programs in PL/1 within the range of its linguistic possibilities, which intersects the possibilities of ALGOL. At the end of Chapter 1 information is presented which will help the reader prepare his own program for translation and readoff on the machine.

In Chapter 2 the PL/1 concepts introduced in Chapter 1 are refined and, in addition, new concepts are introduced which are as it were a logical development of the concepts of ALGOL. Acquaintance with this chapter will enable the reader to use PL/1 means similar to ALGOL means in programming and will also help him to apply in complex cases concepts of PL/1 introduced in Chapter 1.

In Chapter 3 PL/1 concepts are stated which have no direct analogues with ALGOL concepts. Without them the picture of the main PL/1 resources would be incomplete and it would not be possible to compile effective programs in a broad enough area of application. At the end of the chapter those PL/1 possibilities are listed which have not been dealt with in this course and with which the reader can become acquainted from the available literature [3,4,5,10] or, for very complex concepts, from documentation for YeS computer operating systems [6,7,8,9].

In Chapter 4, a reference chapter, a summary is given of built-in PL/1 functions and conversions, tables of PL/1 syntactic forms (one of them in comparison with ALGOL), a list of symbols used and also other reference materials.

All the chapters (except the last) are accompanied by exercises, the answers to which are presented at the end of the book (sometimes one of the possible answers is given). The solutions of exercises and analysis of the examples given in the presentation of the main materials play a very important role in the study of the resources of the language and the acquisition of skills in compiling programs in PL/1.

Along with examples and exercises, another means of checking the correctness and completeness of understanding of the studied concepts of the language can be reducible syntactic forms, which express some properties of those concepts more clearly and graphically than when verbal descriptions are used. Analysis of and, later, reference to those forms ought to prevent the appearance of syntactic errors in programs.

Let us note once more that the course is oriented mainly toward practical programmers who need not so much to become acquainted with peculiarities of the language or to study all its concepts, as to acquire fairly rapidly the ability to compile real programs in it. Taking into consideration also that it is assumed that the reader has some experience in programming in ALGOL, a language in a certain respect very similar to PL/1, the author thinks that a certain brevity in the account (compensated by examples, forms and exercises) will be desirable for such readers.

In the present course the PL/1 language is described on the level of a specific representation as the input language of a version F translator for YeS computer operating systems.

The author is obliged to all who contributed to some degree to the writing and publication of this book. The author especially thanks N. P. Trifonov for detailed and constructive criticism of the manuscript, and also T. N. Moshonkiva and A. Kh.-M. Aldakov for numerous useful discussions and constant help in the work.

2174
CSO: 1863/156

UDC 681.3.06:678

USING DATA BASES IN THE ASU TP OF LARGE-CAPACITY UNITS

Moscow PRIBORY I SISTEMY UPRAVLENIYA in Russian No 3, Mar 81 pp 1-3

[Article by engineers V. K. Sheremet'yev and A. I. Vikhter, USSR, and H. Berger, GDR]

[Text] Separation of Programs and Data as a Basis for Standardization

The applied program packs of ASU TP [Automated Production Process Control System] are rather stable in functional composition. Thus, for example, the interrogation, processing and information display modules are a realization of the simplest information processing formulas and are practically identical by content in many systems. Nevertheless a large number of diverse systems are being developed that perform similar functions determined to a significant degree by the different structure of the information files. In most cases data organization is natural and is the result of programmers' use of individual program-writing procedures.

Only the data to be processed change when a specific class of ASU TP is introduced at domestic facilities. Therefore, allocation of the variable part to a special domain, standardization of the data structure and development of facilities for formation of the data base (BD) for each specific facility should be the tasks of the systems programmer responsible for the overall ideology of software (PO) construction.

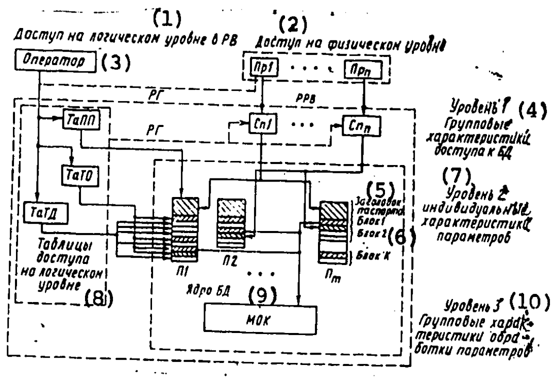
Conversion of a system to a domestic facility should not require significant changes of software. Determination of the processes for development of programs from formation of the data base results in development of the mechanism for combining the programs and data in real time (RMV). To do this, methods of constructing the data structures, organization of access to data at logic and physical levels used in general-purpose information systems, the basis of which is the concept of the data bank [1, 2], can be used in development of ASU TP software. Specifically, the language of processing control to be considered is close to the languages of data description, while the language of operator-data base communication is similar to the languages of data manipulation utilized in data banks.

Specialists of the USSR and the GDR developed a system using a data base on the basis of the M-6000 computer complex within the framework of the intergovernmental agreement between these countries to develop installations of the Polimir type [3, 4].

The characteristics of using a data base on the example of the ASU TP of the Polimir installation for polyethylene production are considered in the article. Special attention is devoted to design of the data base structure, organization of access to it at logic and physical levels and formation of the data base using special language facilities.

Structure of the Data Base

The structure of the data base and organization of access to it are shown in the figure. The data base consists of a core, tables for describing its structure that provide access to the core at the logic level and access lists SP1-SP$_n$ to the core at the physical level.


(1) Доступ на логическом уровне в РВ
(2) Доступ на физическом уровне
Оператор (3)
Пр1
Прn
РГ
РРВ
ТаПП
Сп1
Спn
ТаГО
ТаГД
Уровень 1 (4) Групповые характеристики доступа к БД
(7) Уровень 2 индивидуальные характеристики параметров
(5) Заголовок паспорта
Блок 1 (6)
Блок К
Таблицы доступа на логическом уровне (8)
П1
П2
Пm
Ядро БД (9)
МОК
Уровень 3 (10) Групповые характеристики обработки параметров

Structure of Data Base and Organization of Access to It: RG--generation mode; RRV--real-time mode

Key:

1. Real-time access at logic level
2. Access at physical level
3. Operator
4. Level 1: group characteristics of access to data base
5. Certificate heading
6. Block
7. Level 2: individual characteristics of parameters
8. Tables of access at logic level
9. Data base core
10. Level 3: group characteristics of parameter processing

The data base core is a combination of the certificates of the technological parameters and the common constant files. The certificate of the technological parameter contains all data or references to the data which are needed to process the parameter. The common constant file (MOK) combines the constants repeated for individual certificates. The tables for describing the core structure identify the individual structural components of the data base.

The information in the data base is distributed in three levels (see the figure). The table of the parameter lists (TaPP) and lists SP1-SP_n form the lists of identifiers and the relative addresses of the certificates of the technological parameters and are the group characteristics of user access to the data base. The set of certificates P1-P_m of the technological parameters in the data base core and the tables of access to individual elements of the certificate characterize the individual features of processing indivudal parameters. All the repeated characteristics for groups of parameters are allocated to the lower level in the MOK. Organization of the data base permits collective access of user programs PR1-PR_n and of the operator to the information concentrated in the data base core.

Structure of the Certificate

Each certificate consists of a title and several units, the number of which is equal to the number of types of processing required for a given parameter. The title of the certificate serves as the external interface required to organize access of information users to the data base. It also contains information on the state of the measuring circuits of sensors of the corresponding parameters. Each block contained in the corresponding certificate is the internal interface with respect to the standard primary or secondary processing operator. The certificate block actually comprises part of the standard operator and contains coefficients and placings and also the results of the work of a standard operator. These data also provide external access for making changes and to print out the results. Moreover, the block title contains features for control of the standard operator work. On the whole, the certificate is a set of external and internal program interfaces which link the operation of the software system.

The modular principle of constructing the certificate provides independence of separate standard processing operators from the overall data structure and the specific use of the system.

The structure of the data base core guarantees simple access to the information and correction of it. The information available in the certificate can arbitrarily be separated into control and that to be processed. Control information is used to organize preliminary processing of the information, while information to be processed contains the input data and the results of processing. Moreover, one can distinguish permanent, conditional-permanent and variable information components. Permanent and conditional-permanent parts are formed during initial generation of the data base. Conditional-permanent information can be changed in the RMV by using the access equipment. Variable information reflects the status of processing the information coming from the sensors.

Access to the Data Base

A description of the core structure is stored in the data base to provide independence of user programs from its structure and for convenience of access to the data base. The data in the base core have structures of several types: information component, certificate block consisting of the components and the certificate which includes the title and a number of blocks.

Description of the core structure (tables of access at the logic level) is a complex of retrieval tables (TaPP, TaTO and TaTD) to which the data required to retrieve information in the data base core are reduced.

The table of the lists of parameters contains identifiers of the technological parameters and data for retrieval of the certificates corresponding to them, the table of the types of operators (TaTO) contains identifiers of the types of standard processing operators and the corresponding certificate blocks and the table of types of data (TaTD) contains identifiers of the types of data that are the elementary units of information from which the sub-blocks and titles of the certificates are formed. Each of the tables describes the types of structures of a specific level and consists of blocks of identical structure fixed in value and which form in combination the table characterizing the entire set of types.

Access to the data base core is organized in a different manner as a function of the information user. To increase speed, the access of user programs to the data base is gained at the physical level using lists $SP1$-SP_n of the relative addresses of the required cells. Thus, retrieval of information in real time is essentially excluded. Lists of relative addresses are formed by the generation system using the tables which describe the structure of the data base core.

When service personnel responsible for modification of the data processing system with regard to changing a production situation gain access to the data base, data retrieval occurs in real time according to the tables for description of structure. This permits one to use parameter identifiers. The rate of access should have no special significance since this access does not have high priority.

Thus, the rate of access to the data is provided in the first case and convenience of access is provided in the second case, although the process of linking programs to data itself is identical. The difference is only in the fact that information retrieval is related to the generation process in the first case.

The distinguishing feature of the data base for an ASU TP is joining of the reference function and the information restoration function. To economize on machine resources, the information coming from the sensors is restored directly prior to being given to the user. Thus, interrogation and preliminary processing of information are accomplished only if there is a corresponding request.

Processing Assignment Language

A special language similar in form to forms of the "fill in the blank" type, is used to enter input data and to assign processing of it. But unlike forms with fixed positions and rigid format of macroinstructions, it is similar to free format languages. The immediate significance of the data follows immediately after the identifier which determines the type of input data. A restriction is observance of the required sequence of identifiers. Precise observance of the format and the number of spaces between positions is not obligatory.

The processing assignment for each production parameter is described alternately in this language. The data required to generate the tables of access to the data base and the titles of the certificates are described in the first part of the assignment and then the data required to realize individual types of processing (correction of flow rates, scaling and so on) are described in specific sequence. Up to 25 types of processing of parameters is provided. Each parameter utilizes its own set of types of processing. Thus, the sequence of the types of processing determines the assignment for processing each parameter.

Data Base Generation

Information entered by means of the processing assignment language is converted during generation and the data base core and some tables of access to it are formed. Thus static adjustment of the system for specific parameters of the facility is accomplished. The generation system has a modular structure and consists of a core and set of generators, each of which generates blocks of specific types. The set of generated blocks can be changed by replacing the generators.

Generation of an entire system is accomplished in five steps. The first two steps are used to convert the expressions of the input language to some intermediate language in which the specific structure of the data base is not yet reflected. Conversion from the intermediate language to the internal structure of the data base is completed only in the fourth step of generation. Therefore, utilizing different modifications of fourth-step programs, one can create different versions of data base structure.

The software system is generated by using both the pack and dialogue modes. The latter is used to control the generation modes and operational corrections upon detection of errors.

The system can be generated directly at the computer complex installed at a facility to permit one to make changes directly during introduction of the system at the production facility.

Language of Maintenance Personnel Requests

A special language is used to correct the data base and to organize access of maintenance personnel to it. The language facilities permit callup onto the display screen and to print information from the data base that characterizes the status of individual units and of the installation as a whole in digital and graphical form, to call up complete information from the data base about a single production parameter, including the preliminary processing constants, results of processing and so on, check and modification of individual components of the data base, inclusion and exclusion of any production parameter to and from processing, for example, with regard to repair of a sensor, receipt of a list of the measurement points related to a single cycle and transfer of a measuring point from one interrogation cycle to another, elimination or inclusion of individual functions of the system, programs for display output or for printing individual forms of information display, printout of information about malfunctioning sensors upon callup and entry and changing of the time parameters of the system (entry of systems time, changing the time of information restoration on the display and so on).

The language is open for increasing the number of requests. The main components of the language are the type of request, the identifiers of the certificates of production parameters, the blocks of the certificate and the components of the blocks. When individual components of the data base are summoned and corrected, tables of access at the logic level are used (see figure). Thus, for example, the identifier T 6036 GA GI is used to retrieve the upper temperature setting with number 6036. The certificate is retrieved by using the TaPP according to the designation of production parameter T 6036, the required block is retrieved using the TaTO and the name of the GA block (the block of exceeding emergency boundaries) and the component inside the block is retrieved by using the TaTD and the name of component GI (the upper emergency boundary). In those cases when high speed is required, for example, when restoring information on the screen, access is gained at the physical level.

Conclusions

A system in which a data base was used as the basis was checked in the confectionery shop of the Production Association Polimir imeni 50-letiya Belorusskoy SSR (Novopolotsk). The following functions were realized in the system: interrogation of the low- and medium-level sensors, preliminary processing of analog information, diagnosis of the operation of the interrogation and preliminary processing subsystem, recording of malfunctions in the circuit of the monitoring and measuring devices, signalling of deviations from the norm, writing a report of the course of the production process, presenting information to the operator on displays and calculation of secondary indicators.

The use of a data base provided convenience of static and dynamic adjustment of the system, operational rearrangement of the latter as changes were made in the instrument and production equipment and a considerable saving of memory. Static adjustment of the data base for processing specific production parameters and for realization of a given set of functions for each parameter are accomplished by using a generation system. Dynamic rearrangement of the operation of the system in real time is accomplished by entry of tasks through the display.

Tests confirmed the correctness of the adopted engineering solutions. The system as a whole showed good adaptability to the conditions of the specific object due to the use of a centralized data base and is now being developed further on the basis of using a unit system of software and hardware of the SM EVM [International Small Computer] series.

Bibliography BIBLIOGRAPHY

1. "Informatsionnyye sistemy obshchego naznacheniya" [General Purpose Information Systems], Moscow, Statistika, 1975.

2. Martin, J., "Organizatsiya baz dannykh v vychislitel'nykh sistemakh" [Organization of Data Bases in Computer Systems], Moscow, Mir, 1978.

3. Vol'ter, B. V., V. K. Sheremet'yev, and A. D. Lesovik, "Standardization of Communications in the Software Structure of the ASU TP Monitoring System," PRIBORY I SISTEMY UPRAVLENIYA, No 1, 1975.

4. Sheremet'yev, V. K. and H. Berger, "The Structural Approach to Construction of Software of Information Functions of the ASU of a Large-Capacity Installation," VOPROSY PROMYSHLENNOY KIBERNETIKI, No 48, 1976.

6521
CSO: 1863/155

MULTIPROCESSOR SYSTEMS

STATE COMMISSION ACCEPTS INSTITUTE OF CYBERNETICS DESIGN FOR MULTIPROCESSOR SUPERCOMPUTER

Moscow SOTSIALISTICHESKAYA INDUSTRIYA in Russian 19 Feb 81 p 2

[Article by correspondent Zh. Tkachenko: "Computers and Systems"]

[Text] A State Commission has accepted the design of a new, high-performance, multiprocessor computer. The computer is unlike any other in the world.

T. Mar'yanovich, Party committee secretary at the Institute of Cybernetics of the Ukrainian Academy of Sciences, relates: "At the institute we have also created the latest automated systems for various levels." This concerns systems to control separate technological processes, enterprises, large territorial associations and sector complexes.

During the past five years the introduction of developments made by this advanced group of workers has produced an economic effect of 195 million rubles.

CSO: 1863/23-P

UDC 518.74:007:57

CONSTRUCTION OF ALGORITHMIC LANGUAGE FAMILIES FOR PROGRAMMING AND DESIGNING MULTIPROCESSOR COMPUTER SYSTEMS

Kiev KIBERNETIKA in Russian No 1, Jan-Feb 81 (signed to press 13 Feb 81) pp 1-7

[Article by Viktor Mikhaylovich Glushkov, academician and director of the UkSSR Academy of Sciences' Institute of Cybernetics, Kiev, Yuliya Vladimirovna Kapitonova, doctor of physical and mathematical science, laboratory director, Institute of Cybernetics of the UkSSR Academy of Sciences, Kiev, and Aleksandr Adol'fovich Letichevskiy, doctor of physical and mathematical science, section chief, Institute of Cybernetics of the UkSSR Academy of Sciences, Kiev. Received by the editors on 16 Nov 1980]

[Text] In articles [1] and [2], a method of formalized technical tasks for design of discrete systems was proposed. Use of this method for design of this particular class of systems requires development of mathematical models of this class of systems, methods and facilities for solving the basic problems of design of analysis, synthesis and optimization, as well as language facilities for representation of the systems being designed at the various stages.

A review of mathematical models and some methods for solving the basic problems of design of multiprocessor systems was made in article [3]. Here we shall consider the language aspects of design of multiprocessor system circuitry and software by the method of formalized technical tasks. The diversity of methods and means used in the various stages of design leads to the need of organizing language facilities into a family of coordinated languages that have a varied functional purpose. Independent development of various types of language facilities (problem orientation, functional and procedural languages, machine-independent and machine-oriented languages, etc.) prevails to a great extent in today's practice of programming. Attempts to unify the diverse facilities and construct "large" languages such as PL/1, ALGOL-68, ADA and others, even though they achieve conveniences in some cases, have at the same time considerable shortcomings, not the least among which are the difficulties of efficient realization.

Use of different algorithmic languages requires a coordination of them that would provide the capability of transfer from one language to another at different levels of realization, as well as the capability of using the programs written in the different languages within one system. Coordination of languages of a family is provided by the common character of the mathematical models behind their semantics, and the common requirements that express the relationship between the user, the language and the system in which the language is realized and used.

This article is a generalization of the experience of the development and use of language facilities in the activities on computer software by the UkSSR Academy of Sciences' Institute of Cybernetics. It also contains the methodological principles for developing algorithmic languages for new computer systems under development with the participation of the Institute of Cybernetics of the UkSSR Academy of Sciences. Formation of the point of view presented in this article was influenced by the ideas of today's technology of programming clearly expressed in the "Works of the Working Conference of the IFIP [International Federation of Information Processing]" [4]. The PASCAL language was selected as the basis for traditional language constructions [5].

Let us note that the idea of creating a family of coordinated languages for parallel programming is also discussed in article [13], although from somewhat different starting positions. This work may also be labeled a sufficiently complete review of today's facilities for parallel programming.

Classification of Languages of a Family

Languages of a family are classified by several features.

The first feature corresponds to program concept levels. The following four basic concept levels are distinguished:
--problem;
--functional;
--sequential-algorithmic; and
--parallel-algorithmic.

The problem level is used to define the classes of objects with which the program operates and their properties and the formulation of the problem that the program solves. A problem program expresses a requirement of constructing objects meeting particular properties and in particular relations with source data. The functional level is used to define the functions computed by a program. At this level, recursive functional definitions are widely used. A functional program requires computation of values of particular functions for particular values of arguments.

The sequential-algorithmic concept level defines a program as a means of generating sequential processes of computations with the information medium in which the program operates.

The parallel-algorithmic level defines the aggregate of sequential programs or devices (modules) functioning synchronously or asynchronously, interacting among themselves by exchange of data and control information (messages).

Facilities of different levels may be used in one and the same program. Therefore, the referral of a language to a particular level is not stringent and means that the facilities of a given level are most developed in the language in question and comprise its methodological basis. From the point of view of compatibility, it is convenient to consider any program as a program of the parallel-algorithmic level, isolating the other levels as special cases. A sequential program may be considered as a parallel program consisting of one module, and a functional or problem--as a sequential program defining the single step process of computations (nondetermined, generally speaking, in the problem case).

The second feature for classification of language families involves the orientation to methods of realization. Let us distinguish the following basic methods of realization:
-- interpretation;
-- translation;
-- complexing of off-the-shelf modules; and
-- hardware circuit implementation.

Languages may be classified by orientation to problem classes. This classification may be rough (numerical programs, string processing, data base management, etc.) or more refined (problems of linear algebra, optimal control, translation, designing of specific classes of objects, etc.).

Finally, an important feature of classification is the orientation to the user, for example:
-- the applications programmer;
-- the systems programmer; and
-- the hardware developer.

Data Types

Scalar types are defined just as in the PASCAL language. They include logic (bit) values, number and symbol sets and finite sets of objects that are specified by a listing of the designations of the elements of these sets. Additional scalar types may be included in system programming languages to designate parts of machine words (byte, halfword, etc.), addresses, names, etc.

Structural types, in the final analysis, are made up of scalar and are divided into types of strings and files and the language and theoretical-set types of the component objects. Structural types are also divided into static and dynamic. All static type objects have identical dimensions and the value of each variable of the static type may be stored in some previously assigned, connected region of storage. The dimensions of this region are defined at the time of the program call and may depend on the values of the actual parameters of this program. On the other hand, dynamic type objects have no fixed dimensions, and the memory for storage of values of a dynamic variable must be assigned in the process of program operation. In addition to these types, there are special structural types in a language for input-output operations: input and output files.

Strings are sequences of symbols or bit strings with a fixed dimension. Static type strings have a fixed length. The length of dynamic type strings may vary. Basic operations on static type strings allow only symbol-by-symbol processing, while complicated operations such as pattern recognition and substitution may be made on dynamic strings.

Arrays. The concept of an array as used here generalizes the traditional concept of a rectangular array with integral subscripts. An array is a mapping of a set C, called the domain of arrangment, into the domain of data D of a particular type. The range of definition of an array $A \subseteq C \rightarrow D$ (the sign of inclusion indicates that A is a partial mapping from C into D) is always assumed as finite. The array type is defined by the D type of the values of its elements, the domain of arrangement of C and the set $C_0 \subset C$, which limits the range of definition of this array.

The domain of arrangement is usually a Cartesian product $C = C_1 X...X C_m$ scalar types or types of strings, and $C_0 = C_1^{(0)} X...X C_m^{(0)}$ is the Cartesian product of types contained in $C_1, ..., C_m$. The number m is the dimension of the array. A conventional rectangular array is obtained if $C_1 = ... = C_m = Z$ is a set of integers, and $C_1^{(0)}, ... C_m^{(0)}$ are finite intervals. The finite set C_0 is necessarily indicated for static arrays. For dynamic arrays, $C_0 = C$. The dynamic array concept is usually applied for representation of large volumes of data that may be arranged in external storage (on disks).

Access to array elements is determined, as usual, using variables with subscripts. However, in addition to operations on elements of arrays, operations on arrays may be performed. Taken as the basis is the set of operations defined in [6] and which includes elminations, overlay, shifts and element-by-element operations.

Component objects are used to represent heterogeneous data. The static component corresponds to the concept of notation in the PASCAL language. Dynamic components are defined in articles [7] and [8]. The concept of a dynamic component has a number of advantages over the corresponding facilities in the ALGOL-68, PASCAL and PL/1 languages. First, it makes it possible to operate with components considered as a single whole, and not only with elements of the components. Second, the concept of a component is independent of representation and may be considered an abstract type.

Let us consider the definition of a component object. Let Ω be some set of elements called marks. (Arnost')--a nonnegative integer--is specified for some marks. Marks of (arnost') 0 are used to designate primary components and may, generally speaking, include any scalar types of data. A component object is an oriented multigraph with an isolated initial vertex, in the set of vertices of which Q has been assigned the function of marks α, which maps the set Q into Ω. If the mark α (q) of the vertex q has a positive (arnost') m, then m arrows come out from this vertex and they are all numbered with different natural numbers from 1 to m. If the (arnost') of the mark is not specified, then the number of arrows may be arbitrary. All vertices of the component must be accessible from the initial. Also discussed in [8] are (klubki) of component objects. A set of components is called a (klubok) if together with each object it contains all its subobjects, and if the initial vertices of two components coincide, then these components also coincide.

Marks of components are considered operations for forming complex objects. The algebra of components, which is obtained with a consideration of these operations, forms the basis for defining computations with components. It is also used to define the external language representation of component objects.

Language Objects. For external representation of data, first of all component objects and strings, data language is used. It is not fixed and various programs may be adjusted to various data languages through the use of special facilities for describing the syntax. Objects that allow description in data language are called language objects. Strings and component objects are used for their internal representation. Elements of algorithmic languages of a family may be imbedded in the data language. This makes it possible to use languages of a family as languages for designing algorithms.

Theoretical set structures are used, first of all, in the initial stages of development of algorithms, in particular, at the problem and functional levels. They include sets, relations and functions constructed with the operations permitted in the language. Set elements may be any data structures already defined in the language. Theoretical set types of data are considered abstract types of a special form. In the final analysis, theoretical set objects are realized that can be used in algorithms may be indicated at the intermediate level of development. The general methodology for introduction and use of theoretical set data structures is described in [9].

Information Medium and Structure of the Program

It is assumed that languages of a family are realized in a system that uses a specific organization of programs and data. A prototype of this organization is the organization of the PROYeKT system [10] and its realization the PROYeKT-YeS which was implemented in recent years on machines in the YeS [unified system] series. Each object that is input into the system, be it a program or data structure, is input into the information medium of the system and changes its status, which is defined by the aggregate of the objects put into the system earlier and the links between these objects and the adjustable mechanisms for processing and using them. Man and system interact with high-level languages which include:
-- directive languages;
-- data languages; and
-- algorithmic languages.

All objects of the information medium are divided into information libraries. Each information library contains the programs and data used in solving some problem or class of problems. In particular, an information library may correspond to a package of application programs that use a certain form of organization or a system of programs for some applied field. Each information library may be adjusted to its own languages (directive, data and program). In addition, an information library is adjusted to the particular types of objects that may be in it.

An information library is divided into sections, each of which uses its own system for naming objects and the link between them. A section of an information library consists of modules which are divided into basic and subordinate. Basic module names are called global. These names may be used directly by all modules of the section. Each basic section may have modules subordinate to it. These modules are given local names which are collected into the local nomenclatures of the basic module. Two basic modules may use common local nomenclatures if this is indicated when they are put into the system. Local nomenclatures may be divided by types of named objects. External representations of names in these nomenclatures may coincide. They are distinguished in texts by the use of names of types. Values of local names are directly accessible from a basic module and modules subordinate to it.

Linkage between the various information libraries and sections can be organized by the use of complex names (name of the information library, name of the section, name of the basic module, type and local name). In the process, it must be kept in mind that calls to other basic modules, sections or information libraries are more costly than calls within one basic module. In addition, such calls must meet certain limitations imposed by the diversity of languages and nomenclatures used.

Modules are divided into program and data modules. The external representation of a program module consists of a title, descriptions and a statement. Indicated in the title are the formal parameters used in calling a given module from others, and information on subordination. Some descriptions of a basic module are common to all modules subordinate to it (exactly what kind is determined by the programming languages used).

Problem Level

Descriptions at the problem level have the nature of mathematical definitions. Defined are the necessary types of data, among which the basic role is played by abstract and theoretical set types, objects, variables, functions and subroutines. Descriptions at this level may be incomplete; for a function, for example, perhaps only its designation, type of arguments and type of value are indicated. To construct objects and formulate conditions, theoretical set constructions, quantifiers, recursive definitions and other facilities of the language of mathematics are used extensively. As the basis for formalization of these facilities, the facilities of the language of the theory of a system for processing mathematical texts may be used [11] and [12]. A reasonably restricted language of the problem level may be realized directly through translation, interpretation or complexing of programs from off-the-shelf modules. The capabilities of parallel data processing in a multiprocessor system are derived in the process directly from the definition of the conditions which the objects being designed must meet. In particular, parallel computations permit theoretical set operations, checking of conditions that contain quantifiers, etc.

Richer languages are used to formulate the initial technical tasks which in turn are used as data for design programs. These programs make it possible to effect successive refinement of the algorithm being designed through construction of intermediate realizations of types of data, statements, functions and subroutines, to perform optimizing transformations, and check the dynamic and static properties of the program being designed.

Functional Level

The basic facilities of this level are recursive functional definitions which may be represented by systems of equations of the form:

$$f_i(x_1, \ldots, x_n) = F_i(f_1, \ldots, f_m, x_1, \ldots, x_n),\ i = 1, \ldots, m,$$

where $f_1, \ldots, f_m$ are unknown (determinable) functions, and F_i are expressions constructed from symbols of the unknown functions, arguments $x_1, \ldots, x_n$ and constants. The smallest fixed point is considered the solution. To calculate the value of $f_i(a_1, \ldots, a_n)$, each equation is developed into an infinite tree and analysis of computations occurs simultaneously along all branches with the specified set of values of the arguments. If the width of the tree is sufficiently large, then parallel computations may be organized. Especially important is the case when the unknown functions $f_1, \ldots, f_m$ are arrays, and the sets of values of the arguments $(x_1, \ldots, x_n)$ traverse some subset $E \subset C$ of the domain of arrangement of C. For example, let there be a need to compute the value of the arrays $f_1, \ldots, f_m$ at all points of the domain E. The possibility of parallel computations is now determined by the width of the front of the wave $E_k \subset E$, which consists of those sets $(x_1, \ldots, x_n) \in E$, for which the value of all functions $f_1, \ldots, f_m$ in the k step can be simultaneously computed. If the front of the wave is sufficiently large, then parallel computations can again be organized. As applied to the arrays, the system of functional equations may be brought to the structural form:

$$f_i = F_i\ (f_1, \ldots, f_m),\ i = 1, \ldots, m,$$

where F_i are expressions in the algebra of the arrays [6].

The functional level also includes facilities for transformation of component objects and strings using systems of relations. Called a relation is the equality of the form $F(x_1, \ldots, x_n) = G(x_1, \ldots, x_n)$, where F and G are expressions in the algebra of the component objects, and $x_1, \ldots, x_n$ are parameters. This relation is said to apply to the component object S, if there exist objects $a_1, \ldots, a_n$ such that $S = F(a_1, \ldots, a_n)$, and the object $G(a_1, \ldots, a_n)$ is called in the process the result of the application of this relation to S. The equality of the components in the process may indicate either their isomorphism or equivalence. In the general case, the relation may be supplied with the condition $P(x_1, \ldots, x_n)$, which the sought objects $a_1, \ldots, a_n$ must meet. If it is necessary to apply a system of relations to all subobjects of some component, then parallelism may be used both to check applicability and to apply the relations simulataneously to several subobjects. The technique of applying relations was developed in the ANALITIK language [14] and the programming languages of the PROYeKT system [10]. Problems of parallel realization of relations were studied in article [15].

The facilities of the functional level may be used in combination with the facilities of both the problem and the sequential-algorithmic levels. In the latter case, additional possibilities of parallel processing emerge through use of the overlap technique [3] for the sequentially executed parallel statements.

Sequential- and Parallel-Algorithmic Level

The basic facilities of the sequential-algorithmic level do not differ fundamentally from the standard facilities of traditional programming languages, except that use of certain types of parallel statements (parallel or group assignment, for example) is permitted.

Let us examine the parallel-algorithmic facilities. A mathematical model of a parallel program is a multilevel network of algorithmic modules with a variable structure [3]. The concept of an algorithmic module is derived through combination of two concepts: of a program that describes the functioning of a module, and of a component [komponenta]--of a location where the given program is executed. In the same component, different programs may be executed at different times. Descriptions of the components and the programs executed in them are placed among the descriptions of the parallel program which itself represents control of this network and is called a control program. Programs situated within the control are called subordinate with respect to their own control program. Since multilevel control is permitted, subordinate programs may be controlling with respect to programs at a lower level.

A program is combined with a component by a control program by using the initialization statement VYPOLNIT' PROGRAMMU $A(x_1, \ldots, x_m)$ V KOMPONENTE K [execute program $A(x_1, \ldots, x_m)$ in component K]. If component K is free at the given time, then the program A is put into this component (or initialized in it) with the actual parameters $x_1, \ldots, x_m$. If another program is being executed in K, then the statement A $(x_1, \ldots, x_m)$ goes to the queue and will be executed in K after completion of the program already in it and all the programs that are already in the queue. If only one program is always executed in some component, then the description of this component and the program may be combined in one description. Such a component begins operating automatically simultaneously with the call of the control program.

Interaction of Components. To describe interaction of components in the process of their parallel operation, two basic mechanisms are used: direct exchange of information and use of common memory.

Direct exchange of information occurs with the use of static and dynamic links between the components. Static links are described just as in the ALGORITM language [10]--through indication of identification between input and output variables (channels of the components). Descriptions of these variables are included in the descriptions of the components. An input (output) variable of a component must be assigned the specific type of data that is transmitted through the corresponding channel. In addition, these variables are assigned specific properties that describe the data transmission method (control variables, informational, with buffering, queues, etc.). Descriptions of input and output variables, just as the other descriptions placed in the description of the component, may be used in the programs executed in the given component. To facilitate automatic inclusion of these descriptions in a program, components having an identical description, are assigned a type. This type may be described separately, and its name may be used to simplify descriptions of components and corresponding programs. If the type of program does not correspond to the type of component in which it must be executed (for example, the program has its own input and output variables), then the correspondence between them is established by including additional information in the initialization statement.

Dynamic links are not explicitly described, but are formed during execution of exchange statements. Two types of exchange are differentiated: transmission of messages and exchange of data. Messages are transmitted using the statement: PEREDAT' SOOBSHCHENIYE X PROGRAMME A (K) [send message X to program A (K)], where A is the program name and K is the component name. During execution of this statement, a dynamic link is established between the given component and component K, and message X is sent through the corresponding channel and written in the message buffer for component K. Access to messages received for the given program A is possible only during operation of this program and is realized by using the statement: OBRABOTAT' SOOBSHCHENIYE [process message].

The structure of messages which components exchange between themselves is described in the control program and is common to all components and programs subordinate to it both directly and at lower levels of control.

Data is exchanged by using two statements: PEREDAT' DANNYYE X PROGRAMME A(K) [send data X to program A(K)] and PRINYAT' DANNYYE Y IZ PROGRAMMY B(M) [receive data Y from program B(M)]. These two statements correspond to each other. Establishment of a link and transmission of data X from component K to component M will occur at the moment when simultaneously, program B executes the corresponding data transmission statement in component M and program A executes the receive statement in component K. Data X is written in area Y. Before this moment, execution of each of these statements is in the wait state. To reduce the waiting period, the apparatus of message transmission with interrupts may be used. The corresponding statement has the form: PEREDAT' SOOBSHCHENIYE X PROGRAMME A(K) S PRERYVANIYEM [send message X to program A(K) with interrupt]. Execution of it halts the operation of component K and initiates execution of the interrupt program, which in this case must be described in program A. If the gap between execution of the exchange statements is expected to be long, then the components may be synchronized between themselves, exchanging messages on readiness to send and receive data.

Facilities for interaction of parallel processes are easily expressed in terms of the dynamic interaction of components in such models as, for example, Hansen's model [16], which we believe closest to our considerations.

Let us note that interaction with the use of dynamic links corresponds more to program realization, while static links are used more extensively in designing hardware. The combined use of both facilities is the basis for joint design of computer circuitry and software.

Common memory for components of one level is defined by descriptions of those variables that have the special OBSHCHAYA [common] feature. In addition, data from an information library is accessible to all programs. A call to common memory occurs during execution of a statement of assignment, checking conditions in conditional statements, etc. Conflict calls are queued. In addition, the OTKRYT' [open] and ZAKRYVAT' [close] statements make it possible for some program to seize certain sections of common memory for a time. Partial se.zure is also possible, for example, ZAKRYT' DLYA ZAPISI (CHTENIYA) [close for writing (reading)].

Control. Description of a large number of homogeneous components may be facilitated by use of arrays of components, the dimensions of which may depend on the actual parameters of the control program. Fully decentralized functioning of components corresponds to the dummy statement part of the control program. In this case, each component must be described along with its program and all components begin operating simultaneously at the time of the call of the control program. A program ends its operation at the time when all components end operation.

In the case of the nontrivial statement part, a control program may execute initialization of operation of components, interrupt their operation, track the end of operation of components, check conditions of their operation, etc. In addition, a control program may change the structure of links between components, and eliminate and introduce new components from a previously described reserve.

Still another facility for dynamic change of structure involves the dynamic call of programs. Any program may call any other program described at the same or higher level of the hierarchy. In some cases, a call is permitted with lower levels too. Dynamic calls may occur recursively. With a call of a parallel program from control, a change in the structure of the components occurs through addition of new components which are eliminated when the called program ends operation. Use of the apparatus of checkpoints makes it possible to organize interaction of coprograms.

Conclusion

This article presents approaches to constructing a family of coordinated algorithmic languages for design of multiprocessor systems circuitry and software, which are under development at the Institute of Cybernetics of the UkSSR Academy of Sciences. Several representatives of this family have now been developed. They include: the ALGORITM-80 language oriented to joint design of hardware and software, the MMP multimodular programming language that belongs to the parallel-algorithmic level, and the AMK arithmetic macroconveyor language pertaining to the problem-functional level and oriented to the problems of computer mathematics. The latter two languages are intended for programming recursive computers [17].

References

1. Glushkov, V. M.; Kapitonova, Yu. V.; and Letichevskiy, A. A., "Theoretical Principles for Design of Discrete Systems," KIBERNETIKA, No 6, 1977, pp 5-20.
2. Glushkov, V. M.; Kapitonova, Yu. V.; and Letichevskiy, A. A., "Use of Methods of Formalized Technical Tasks for Design of Programs for Processing Data Structures," PROGRAMMIROVANIYE, No 6, 1978, pp 31-43.
3. Glushkov, V. M.; Kapitonova, Yu. V.; and Letichevskiy, A. A., "Theory of Design of Circuitry and Software for Multiprocessor Computers," KIBERNETIKA, No 6, 1978, pp 1-15.
4. "Design of Reliable Software," in "Trudy rabochey konferentsii IFIP" [Works of the Working Conference of the International Federation of Information Processing], Novosibirsk, VTs SO AN SSSR [Computing Center of the Siberian Branch of the USSR Academy of Sciences], 1977.
5. Wirth, N. "Sistematicheskoye programmirovaniye. Vvedeniye" [Systematic Programming: An Introduction], Moscow, Mir, 1976, 183 pages.
6. Glushkov, V. M.; Kapitonova, Yu. V.; and Letichevskiy, A. A., "Data Structure Theory and Synchronous Parallel Computations," KIBERNETIKA, No 6, 1976, pp 2-15.
7. Gorokhovskiy, S. S.; Kapitonova, Yu. V.; and Letichevskiy, A. A., "Programming Facilities and Solving Logic Problems in Software Systems, Basic Concepts of the [word illegible] Language," KIBERNETIKA, No 3, 1974, pp 27-47.
8. Glushkov, V. M. and others, "Basic Instrumental Language of Programming," (Preprint of Institute of Cybernetics of the UkSSR Academy of Sciences, No 79-22), Kiev, 1979. "Programming in the L2B Language," (Preprint of the Institute of Cybernetics of the UkSSR Academy of Sciences, No 79-23), Kiev, 1979.
9. Glushkov, V. M.; Kapitonova, Yu. V.; and Letichevskiy, A. A., "Instrumental Facilities for Design of Programs for Processing of Mathematical Texts," KIBERNETIKA, No 2, 1979, pp 37-42.
10. Glushkov, V. M.; Kapitonova, Yu. V.; and Letichevskiy, A. A., "Avtomatizatsiya proyektirovaniya vychislitel'nykh mashin" [Automation of Design of Computers], Kiev, Naukova dumka, 1975, 232 pages.
11. Glushkov, V. M.; Kapitonova, Yu. V.; Letichevskiy, A. A.; Vershinin, K. P.; and Malevanyy, N. L., "Constructing a Practical Formal Language for Writing Mathematical Theories," KIBERNETIKA, No 5, 1972, pp 19-28.

12. Glushkov, V. M. "System for Automation of Proofs (SAD)," in the book "Avtomatizatsiya obrabotki matematicheskikh tekstov" [Automation of Processing of Mathematical Texts], Kiev, Institute of Cybernetics of the UkSSR Academy of Sciences, 1980, pp 3-30.

13. Kotov, V. V. "Parallel Programming Languages," Part 1, KIBERNETIKA, No 3, pp 1-12; Part 2, KIBERNETIKA, No 4, 1980, pp 1-10.

14. Glushkov, V. M.; Bodnarchuk, V. G.; Grinchenko, T. A. and others, "ANALITIK (Algorithmic Language for Description of Computer Processes with the Use of Analytic Transformations)," KIBERNETIKA, No 3, 1971, pp 102-134.

15. Shevchenko, V. P. "Parallel Computations in the Algebra of Component Objects," "Automated Extracts of Dissertations of Candidates of Physicomathematical Science," Kiev, KGU [Kiev State University imeni T. G. Shevchenko, Order of Lenin], 1980, 20 pages.

16. Hansen, P. B., "Distributed Processes: A Concurrent Programming Concept," CACM [Communications of the Association for Computing Machinery], Vol 21, No 11, 1978, pp 934-941.

17. Glushkov, V. M. and others, "Recursive Machines and Computing Technology," "IFIP Congress-74," Stockholm, 1974, pp 65-70.

8545
CSO: 1863/149

VOICE RECOGNITION/SYNTHESIS

MINI-COMPUTER VOICE COMMUNICATION SYSTEM

Moscow PRIBORY I SISTEMY UPRAVLENIYA in Russian No 1, Jan 81 pp 13-14

[Article by engineers V.K. Kucherenko and I.K. Kucherenko: "Mini-computer Speech Communication System"]

[Text] This article examines an experimental system of voice communication with mini-computer, consisting of subsystems for segmented recognition of speech forms and the segmented synthesis of speech transmissions. The speech communication system is SM-1 mini-computer based and operates real time.

A feature of the system described is the use of interval durations between nulls in the speech signal, and also between adjacent points of local extremes of the speech signal.

We will review several general questions arising in the development of the computer based speech communication systems, specifically a number of limitations which are being de-bugged for subsystems(routines) involving speech form recognition and also synthesis of speech communications, created for pratical application.

The most substantial limitation is associated with the recognition time for speech form and synthesis time for speech communications, both of which must be rather low in order to realize one of the basic advantages of speech communication: convenience and speed of data transmission from the user to the computer and conversely.

Another limitation for the speech form recognition routine results from requirements imposed on pronunciation conditions for voice communications and speakers [1].

Necessary requirements for the practical application of speech communication with computers in several small systems include also low cost of the speech communication system and small size.

Naturally, a sufficiently flexible and powerful computer system with the use of speech communications must permit simultaneous interface with several independent users working at their own terminals. One of the ways to effectively solve this problem linked with the continual drop in the cost of mini-computers is to supply each terminal with a micro-computer, whose functions include analysis and synthesis of the voice signal.

Moreover, in small local systems not linked to powerful computers, the use of speech communications systems is possible with micro or mini computers. However, use of the latter in developing efficient speech communications systems with computers is difficult, a situation explained by the relatively low rate of operations in the inexpensive micro computers and their low-volume current memories, which result in a considerable limitation of the recognition algorithm's computing efficiency and of the entire set of speech communications programs. Thus, posing the task of developing speech communications systems with computers for practical application leads specifically to the problem of building these micro and mini computer based systems. One of the important features of the speech communication system also is the capability for the current variation and expansion of the differentiated and synthesized speech communications compositions.

The article further examines the experimental speech communications systems with the mini computer developed by the authors, easily incorporating various types of mini and micro computers and operating real time with a high speed capacity of 150-200,000 operations per second for the processor.

The system described utilizes a SM-1 mini computer and an "Yelektronika-60" micro computer. In utlizing the micro computer, all archive and program sets forming the minimal configuration speech communication system are situated in the micro computer's current storage. Minimum memory capacity necessary for speech form recognition routines is 4K 16-bit words, which permits the recognition of several dozen commands. The minimum memory volume required for the speech communications synthesis routine is dependent upon the vocabulary and pronunciation quality of speech and amounts to 8--24K. In developing the micro "Yelektronika-60" computer based communication system, a series of problems had to be resolved which were associated with regeneration of the dynamic semi-conductor memory, which influences the quality of synthesized speech sounds.

Common to the speech communication recognition and synthesis routines are the hardware options developed by the authors for the input of the speech signal to the computer and output from the computer. After input to the computer memory, the speech signal undergoes a preliminary processing stage, which includes a logical averaging and a hysterisis filtering.[2].

Primary speech signal parameters in recognition of speech forms are form characteristics of histograms for interval duration distribution between null intersections and adjacent exteme periods of the speech signal in the analysis segment. To reduce the dimensions of the span for information characters and further segmentation, a claster analysis is employed [3] and the evaluations obtained by the authors relating to exactness of approximations of speech signal structure frequency.

The speech form recognition routine operates in dialogue mode in real time during instruction as well as in recognition and provides for verification of speakers and to differentiate with a high degree of reliability such similar sounding words as "sinus",,"minus", "Minkus", etc., where instruction in a majority of instances is satisfied by a single realization of the speech form.

We will examine the speech communication synthesis, which provides for realizing any phonetic system in the Russian language. The speech communications segmented synthesis routine includes, in addition hardware and software for input-output of of the speech signal,a selection of programs for automated segmentation of the voice signal with the capability for subseqent control of segmentation results and a set of programs providing for the creation and editing of the segment dictionary, and also search, selection, concatenation, and the reproduction of dictionary elements in the required sequence.

In formulating the elements of the segment dictionary, the voice material used to obtain the segment dictionary is subjected to automated segmentation with the claster analysis employed. The limits of the segments obtained during automatic segmentation are verified, if necessary, based upon graphic and tabular data on spectral and time characteristics of the speech signal. The control of the natural aural quality of the isolated segments and their intelligibility is achieved by monitoring the segments via a loudspeaker. The voice segments thus obtained are then used in constituting the segment dictionary.

Formation of the segment dictionary and synthesis of voice communications is effected with the use of control programs, the flow-chart for which is depicted in illustration 1. The control program organizes the dialogue between the user and the voice communication synthesis routine. During the dialogue process, the control program provides the user with the capability to create or edit the the segment dictionary, and to organize the input of written communication, its automatic analysis and translation into the aural communication. The control program realizes its functions through the "Synthesis" routine and the "Archive" routine, the flow-chart for which are depicted for one version of realizing the speech synthesis routines in illustrations 2 and 3 respectively.

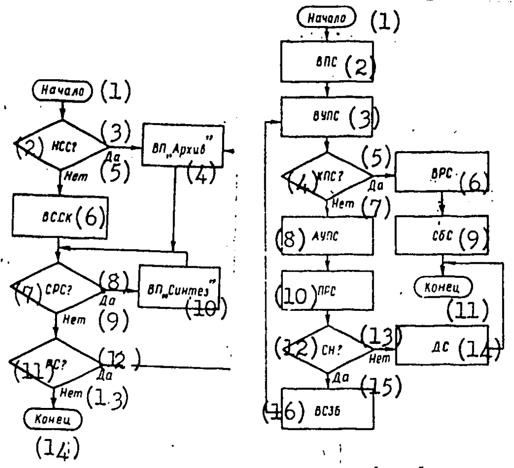


Illustration 1. Flow-chart for speech communications synthesis:
1. Begin 2. New segment dictionary: dump of Main routine "Archive" and "Synthesis" respectively 3. Yes 4. Main routine "Archive" ; 5. no 6. input of the segment dictionary archive and its catalogue; 7. Synthesis of speech communication; 8. yes 9. no 10. Main routine "Synthesis" 11. dictionary editing 12. yes 13. no 14. end

Illustration 2: "Synthesis" Routine Flow-chart

1. Begin 2. Input of written communication 3. isolation of the sequential segment of the written communication 4. end of written communication, 5. yes 6. reproduction of the synthesized speech communication 7. no 8. analysis of the current segment of the written communication and determination of the beginning of the next 9. collection of statistics on the segment dictionary 10. search for the required voice segment in the segment catalogue 11. end 12. segment located 13. no 14. diagnostic communication 15. yes 16. segment selection and output buffer loading

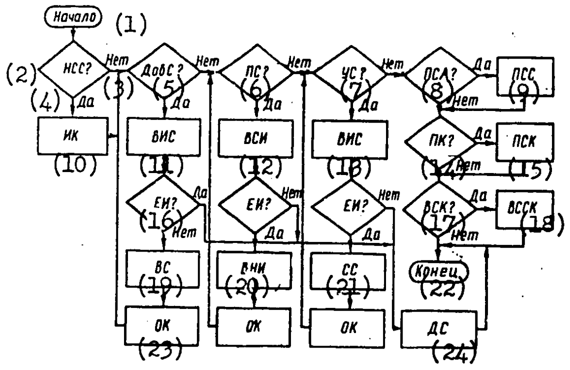


Illustration 3: "Archive" Routine Flow-chart
1. Begin 2. new segment dictionary 3. no 4. yes 5. addition of segment 6. segment re-designation 7. segment removal 8. segment re-distribution in the archive 9. segment re-distribution based upon statistics on the segment dictionary 10. intialize segment dictionary catalogue 11. input of segment name 12. input of old segment name 13. input of segment name 14. catalogue print 15. print of catalogue contents 16. is there such a name in the catalogue? 17. dictionary and catalogue output 18. input of segment dictionary archive and its catalogue 19. segment input 20. input of new segment name 21. deletion of segment and compression of segment archive(file) 22. end 23. catalogue update 24. diagnostic communication

The "Synthesis" routine is carried out by the input of the written communication, its analysis, search, selection of the appropriate speech segments, their concatenation, and also the loading of the output buffer with subsequent reproduction of its content via the speaker. If the required speech segment for the routine, "Synthesis" is not located, the diagnostic communication is output with the missing segment indicated. In this case, the user may add the necessary speech segment to the segment dictionary. At the end of its operation, the "Synthesis" routine assembles statistics of the freqency of the segment dictionary usage frequency. The utility of such information lies in that the segment dictionary is organized as a linear listing with dynamic distribution in memory, and therefore may shorten the time for selecting elements of the dictionary, optimizing their relative locations.

The "Archive" routine provides for the creation of a segment dictionary, and also edits it through input, redesignation, or elemination of segment dictionary elements. The capability for the output of current information on the segment dictionary catalogue, and also of diagnostic communications in the event of error location provides certain conveniences in operations with the speech synthesis routine.

The modular principle of construction for software on the routines for speech form recognition and speech communication synthesis permits creation of an easily modified and expanded system equipped with mini or micro computers for speech communication synthesis, depending upon the needs of the specific application.

We will cover certain possible areas of use for the speech communications systems described above.

Primarily, this automated data-information system and control systems of varying types are distinguished by both the number of recognized and synthesized speech communications and by demands or requirements for computer equipped speech communication systems. The employment of speech communications makes possible the communication or interface with the ASU or the data-information system and providing information to a broad contingent of individuals unfamiliar with computers. Here the speech communication may also serve to verify authorized access to information. The simplest in terms of speech communication with computers are the systems in which the voice material consists of a limited number of discrete communications.

Speech communication with computers may be employed also in ASU TP, for ARM servicing, etc.

Undoubtedly, speech communication with computer is an effective approach in the construction of eronomically optimum man-machine systems.

Literature

1. Martin, T. Practical Application of Speech Input To Computers. TIIER, April, 1976, v. 64, No 4.
2. Tu. Dzh., Gonsales R. Principles in Form Recognition, M.: Mir, 1978.
3. Kucherenko, V.K., Kucherenko I.K. Acoustical Isolation of Homogeneous Segments in Speech Signals.—From the book: Problems in Building Speech Comprehension Systems. M.: 1980.

8851
CSO: 1863/142

NETWORKS

UDC 681.324

CONCEPT OF A MODERN COMPUTING NETWORK

Riga AVTOMATIKA I VYCHISLITEL'NAYA TEKHNIKA in Russian No 2, Mar-Apr 81 (manuscript received 5 Dec 80) pp 3-14

[Article by E.A. Yakubaytis]

[Text] Computing networks are the highly efficient base of the modern data processing industry. The first stage of their development was the creation of centralized tree networks in which n user stations were connected to a single computer via communications channels. At the second stage the ability to join a small number of trees together was evidenced. And, finally, at the third, modern, stage distributed computing networks are being created whose topology is determined by the efficiency of the performance of all data processing processes.

The first large computing network in which the concept of distributed computing and data processing resources was well developed was the ARPA network [1]. A further development of this concept was reflected in the CYCLADES and CNET networks [1]. And, finally, appeared the DATAPAC, TELENET and TRANSPAC networks [1] in which the parameters and procedures for the exchange of information between computing complexes were standardized for the first time on the international level. In the Soviet Union distributed networks were created in Moscow [2], Novosibirsk [3], Riga [4] and a number of other science centers. An analysis of these generally different developments has demonstrated that they have a common basis which defines the concept of a modern distributed computing network. This article is devoted to a discussion of this concept.

1. Model

The fundamental logical concept of a model of a computing network is the system--a hierarchical group of functions implemented in one or more computers, designed for solving specific network problems. Systems are linked to one another by means of conditional lines, called physical connections, forming the logical structure of the computing network. Each system is open-ended, i.e., satisfies the general requirements [5] of the interaction architecture of open-ended systems of the International Organization for Standardization (MOS [ISO]).

All systems are divided (cf. fig 1) into a number of layers, called levels. Each level performs in the computing network a specific functional task and provides service for the level above it. The digest of rules for the interaction of entities of like (identical) levels of various systems is called the protocol. The rules for the interaction of entities of adjacent levels of one and the same system are called the interlevel interface.

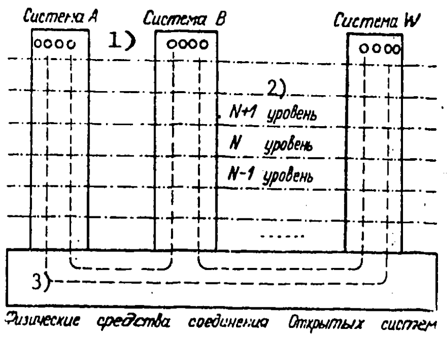


Figure 1. Interrelationship of Applied Programs in a Computing Network
Conventional symbols:

0	Applied program
---	Applied program line of interaction

Key:

1. System A
2. Level
3. Physical facilities for connecting open-ended systems

In the model of a computing network discussed by ISO (cf. fig 2) there are seven functional (logical) levels. The tasks performed by these levels are described in table 1. The top two levels of the system define the processes (the processes of representing data and of controlling applied processes and administrative control). The bottom five levels form the network method of access to these processes. The linking points between the network method of access and processes we will call ports.

Table 1. Functions Performed by System Levels

No of level	Name of level	Functions implemented by level
7	Applied	Execution of applied programs of users, control of terminals and operator stations, administrative control of network.

[Continued on following page]

6	Representation	Representation (interpretation) of the meaning (value) of information transmitted between applied programs, including conversion of instructions and data.
5	Contact	Organization and performance of communications contact between applied programs.
4	Transport	Transfer between processes of data files coded in any manner.
3	Network	Routing and addressing of information; control of data array flows.
2	Channel	Establishing, maintaining and breaking a connection.
1	Physical	Physical, mechanical and functional characteristics of connection.

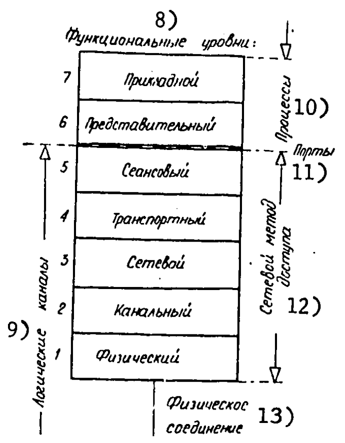


Figure 2. Hierarchy of System Levels

Key:

1. Physical
2. Channel
3. Network
4. Transport
5. Contact
6. Representation
7. Applied
8. Functional levels
9. Logic channels
10. Processes
11. Ports
12. Network method of access
13. Physical connection

The solution of diverse network problems is made possible by means of the specialization of systems, illustrated in table 2. Working, terminal and administrative systems are called user systems. All systems have a 7-level structure. However, in administrative and communication systems levels 4 to 7 are used only for administrative control. In the remaining systems these levels provide both for basic (involving the interaction of users' programs) and administrative control.

Table 2. Computing Network Systems

No	Name of system	Tasks performed by system
1	Working	Assignment of network resources: storage of data arrays, data search, execution of computing operations, simulation of processes, phenomena and entities, and development of software.
2	Terminal	Use of network resources: control of the operation of terminals, preparation of assignments, and interfacing with technological processes for the purpose of measuring and controlling them.
3	Communication	Routing of flows of arrays of information transmitted between working or terminal systems.
4	Administrative	Administrative control of computing network (gathering of statistics, operation records, issuing reports, diagnosis of failures, information regarding operation of network, etc.).

With the introduction of the specialization of systems it is not difficult to produce the logical structure shown in fig 3. Its nucleus is the logical structure of the data transmission network, consisting of communication systems linked with one another by means of physical connections. After fulfillment of the requirements for a standard interface, user (working, terminal and administrative) systems are connected to the data transmission network, as the result of which (cf. fig 3) the logical structure of the computing network is formed.

User systems satisfying the standards for interfacing with a data transmission network are connected directly to communication systems. If a user system does not conform to these standards, then it is connected to the communication system via a logic element of the network called an interface module. When communications contact is performed, the ports of processes located in various systems are connected by means of conditional lines called (cf. fig 2) logic channels. The transfer of data from one process to another takes place through them.

Data input/output (VV) logic modules linked with individual terminal systems complete (cf. fig 3) the synthesis of the logical structure of a computing network.

Computing networks are not always, as in fig 3, single-level. They can form (cf. fig 4) hierarchies in which any pairs of user systems interact regardless of on which level of the hierarchy they are.

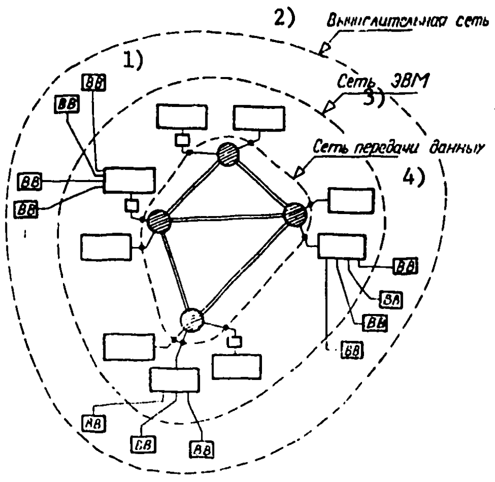


Figure 3. Logical Structure of Computing Network
Conventional symbols:

□	User system
●	Communication system
-⊠-	Interface module
ВВ-	Input/output module
—•—	Physical connection and point of standard interface

Key:

1. I/O unit	3. Computer network
2. Computing network	4. Data transmission network

Interface points at which communication systems are connected with user systems or interface modules must satisfy an international standard called Recommendation X.25. It was introduced [6] by the Consultative Committee on International Telegraphy and Telephony (CCITT) and has become widespread in many countries of the world. For example, in table 3 are given [7] some data on State data transmission networks created or to be created in Europe. Recommendation X.25 calls for the collective use of data transmission channels by the creation of virtual (logic) channels and by the transmission of standard packets through them.

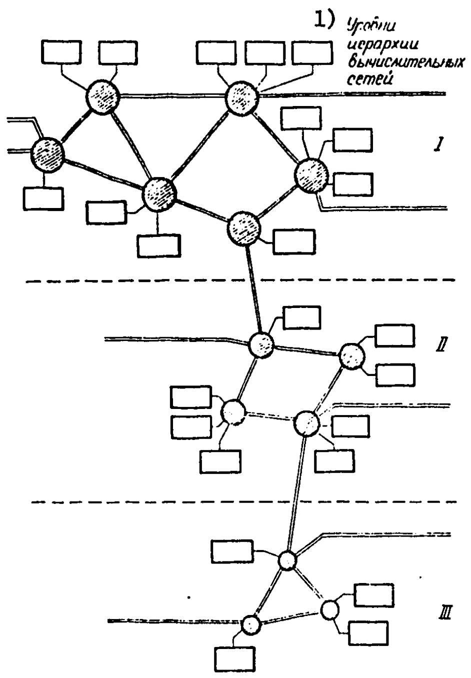


Figure 4. Hierarchy of Computing Networks

Key:
1. Levels of hierarchy of computing networks

Table 3. European Networks for Transmission of X.25 Packets

No	Country	Year of entry into service	Number of network centers	Number of terminal systems in 1980	Number of character-user stations in 1980
1	England	1979	3	100	1057
2	Belgium	1980	3	250	750
3	Holland	1979	3	1400	0
4	Denmark	1981	1	-	-

[Continued on following page]

5	Spain	1979	5	250	9000
6	Italy	1981	3	-	-
7	Norway	1981	1	-	-
8	Finland	1982	1	-	-
9	France	1978	25	3000	3000
10	FRG	1979	7	>500	>1500
11	Sweden	1981	1	-	-

A 7-level hierarchy of computing network protocols has been introduced in conformity with system levels (cf. fig 2). These protocols (cf. fig 5) have the same names as the levels do. Generally several different protocols can be at the same level. The protocols of different levels are quasi-independent of one another. Therefore, the replacement of a protocol at one level should not require the modification of protocols of other levels. In addition, the protocol of any one level is transparent to the protocols of higher levels, i.e., does not introduce distortions into their work.

The protocols of higher levels (4 to 7) characterize the procedures for direct (through the data transmission network) interaction of user systems with one another. Therefore, they are often called through ("end/end") protocols. Protocols of lower levels (1 to 3) describe the interaction of neighboring systems of any type. They, in addition, determine the standards for access to the data transmission network.

At the present time users widely employ programs of YeS [Unified Series] computers (SRV, OKA, KAMA, KROS, POISK, etc.). Therefore, at the first stage of development in computing networks it is advisable to use at higher levels the standards and procedures of YeS computer tree-type teleprocessing. At later stages, of course, special virtual protocols, for terminals, files, assignments, etc., will appear in networks.

The protocols of the physical, channel and network levels are described by CCITT Recommendation X.25. In keeping with it a data transmission network makes possible the transmission of packets, and user systems and interface modules, before transmission, the breakdown of messages into packets and, after transmission, the assembly of packets into messages.

2. Physical Structure

Each system considered in table 2 is implemented in a complex consisting of one or more computers (large, medium, small or micro), and each physical connection is accomplished by means of a data transmission channel. The functions of the interface module are performed by an interface converter which is constructed on the basis of one or more mini- or microcomputers. The input/output module is implemented by means of a terminal.

As a result, the logical structure is converted into the physical structure of the computing network, an example of which is illustrated in fig 6. Since not only one but several systems can be implemented in a complex, the number of types of complexes is considerably greater than the number of kinds of systems. Therefore, in a computing network are used not only working, terminal, communication, and administrative, but also terminal-working, working-communication, communication-user, etc., complexes. Communication complexes (together with the interface converters of any of them) form packet switching centers.

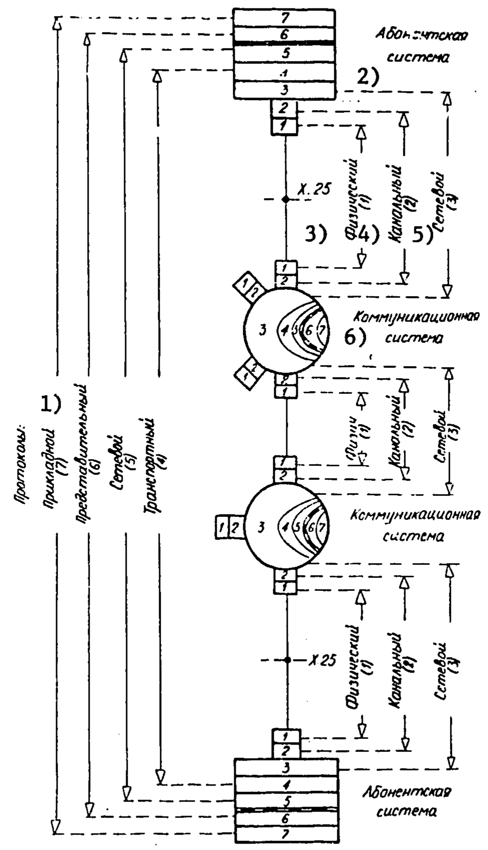


Figure 5. Hierarchy of Protocols for Basic Control of a Computing Network

Key:
1. Protocols: applied (7); representation (6); network (5); transport (4)

[Key continued on following page]

2. User system
3. Physical (1)
4. Channel (2)
5. Network (3)
6. Communication system

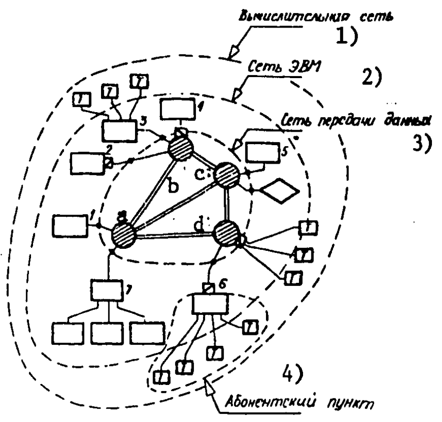


Figure 6. Physical Structure of a Computing Network (For designations of 1 to 7 cf. text.)
Conventional symbols:

Working complex consisting of several computers
Single-computer working complex
Working complex with interface converter
Terminal complex with terminals
Administrative complex
Packet-switching center (without interface converter and with interface converter)
Terminal complex with terminals and interface converter
Data transmission channel with standard interface point
Terminal

[Key on following page]

Key:

1. Computing network	3. Data transmission network
2. Computer network	4. User station

2.1. Data Transmission Networks

Each data transmission network consists (cf. fig 6) of packet-switching centers and data transmission channels connecting these centers with one another as well as with user complexes or interface converters. Large data transmission networks (cf. fig 7) can be multilevel. In these networks the higher the level, the larger the capacity of the packet-switching center and the higher the speed of data transmission channnels. The number of levels of a data transmission network and its number of centers and channels are determined by economics, the topology and reliability of the network and by its speed characteristics.

Points of the interface of packet-switching centers with user complexes and interface converters form the points of entry into the data transmission network. At these points are received or transmitted packets to be transmitted throughout the entire network without disassembly (or assembly). Packets are governed by CCITT Recommendation X.25. The entire data transmission network must be transparent, i.e., must transmit packets the data in which can be coded by any method.

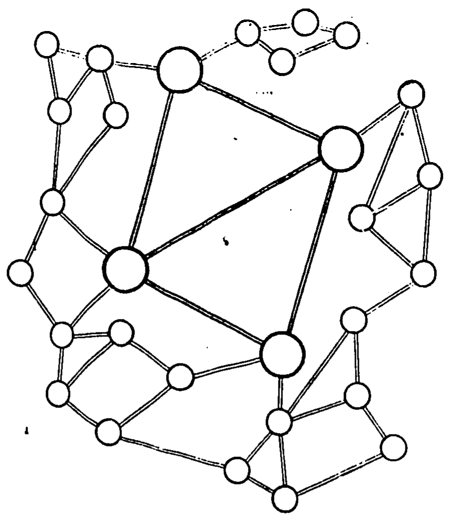


Figure 7. Two-Level Data Transmission Network

[Continued on following page]

Conventional symbols:

¤	Large-capacity trunk line packet-switching center
¤	Zone packet-switching center

A packet-switching center for making possible high data processing speeds is constructed on the basis of mini- and microcomputers and is a multiprocessor center. Only for very small data transmission networks or for the lower levels of large hierarchical networks in which this is permitted by stipulations for reliability can a packet-switching center be a single-processor one.

The entire processing of data in a packet-switching center is accomplished by means of a working storage alone. The external storage (magnetic disks) is used only for loading the software and for gathering statistical data.

In addition to packet switching, a switching center can provide for the performance of interface functions relating to the connection of non-standard user complexes or terminals. For this purpose interface converters constructed on the basis of mini- or microcomputers are included in the center.

A data transmission network, representing a packet-switching network, can also provide for message switching (for computers and telegraph equipment). For this purpose, in the inputs and outputs of the network (centers b and d in fig 6) are added interface subsystems which perform the disassembly and assembly of messages. The assembly/disassembly of messages is not carried out at intermediate centers (a and c in fig 6).

2.2. Working Complexes

Working complexes make it possible to offer the most varied computing and data processing resources to a wide range of users. Computers which have been created specially for the network are used most effectively in them. The YeS and SM computers already produced were not designed for networks. However, by taking special measures the successful implementation of working systems can be provided for in these computers.

Three methods exist for putting together a working complex constructed with one or more YeS or SM computers. The first of these consists in the fact that user complex 4 (cf. fig 6) is connected via a standard MPD [data transmission multiplexer] or PTD [remote processing station] to the interface converter of communication complex b . This converter, implementing the interface module, is in the guise of a standard user station for the MPD (PTD), and of a standard (for the network) user complex for communication complex b .

The second method is logically equivalent to the first. However, here the interface module is implemented not in the equipment linked with the communication complex, but in the PTD, mini- or microcomputer directly linked with user complex 2 (cf. fig 6).

The third method of interfacing the working complex consists in the fact that special programs making possible the creation of the network method of access (cf. fig 2) are written for the YeS or SM computer. Then, having added to network complex 1 (cf. fig 6) a network adapter which responds to standard X.25, it is possible to connect the working complex directly to communication complex a .

In the computing network is also possible the combined utilization of a YeS computer working with ordinary user stations and interacting with the network. This has been necessary primarily in those situations when the YeS computer complex already had ordinary teleprocessing before connection to the network.

2.3. Terminal Complexes

Terminal complexes are created on the basis of YeS or SM computers or microcomputers. For connection to the network each computer must have a network adapter and hardware satisfying the protocols of the computing network.

Network software is based on the utilization of standard operating systems and algorithmic languages of complexes. The chief software problem consists in organizing remote interaction between terminals used in the network with the resources of all working complexes present in the computer network.

Three groups of terminal complexes should be singled out: multiconsole, single-console and YeS computer user stations.

Multiconsole terminal complexes are the most effective means of simultaneous contact between teams of users and the resources of a computing network (e.g., complex 3 in fig 6). The largest of them are constructed on the basis of YeS computers. SM computers are used for creating the others.

Single-console complexes are created on the basis of SM computers or microcomputers. Each of them is designed for the work of a single user at a time.

The YeS computer user station is a terminal complex (complex 6 in fig 6) including one or more terminals. It interfaces with the data transmission network via an interface converter created on the basis of a mini- or microcomputer. This type of terminal complex is the least effective, but it makes it possible to use YeS computer character-user stations which the user already has.

2.4. Administrative Complexes

An administrative complex implements an administrative system forming the basis of the network control center. Each complex is constructed on the basis of one or more mini- or microcomputers and it interacts with all or a group set aside for it of user complexes, communication complexes or interface converters.

If a single administrative complex is used in a computing network, then all the equipment in it is duplicated for the purpose of ensuring high reliability. If there are several administrative complexes in a network, then one of them is the main one. When it goes out of order the functions of the main one are transferred to another administrative complex.

3. Topology of the Network

Let us discuss the principles of the creation of a distributed computing network by using the example illustrated in fig 8. There are five entities (enterprises, institutions, management and planning organizations) in which information is developed or is used. Furthermore, the storage, processing and output of multi-purpose information, regardless of its nature, must be accomplished by $m \geq 1$ complexes.

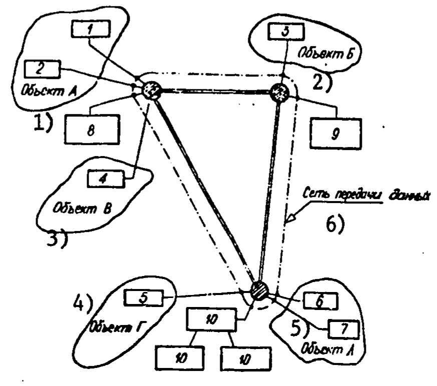


Figure 8. Topology of a Computing Network (cf. text for designations of 1 to 10)
Conventional symbols:

▭	User complex
●	Communication complex

Key:

1. Entity A	4. Entity D
2. Entity B	5. Entity E
3. Entity C	6. Data transmission network

Depending on the location of the points of the development and use of information, the information is divided (cf. fig 9) into local and global. Local information is information which is developed and used at one and the same entity. But if information is developed in one and is used at least in one other entity, then it is global. In turn, two groups must also be distinguished among local information. Information which can load one or more computers every day to a sufficient (economically permissible) extent we will call steady. The rest of local information we will place under the heading of peak.

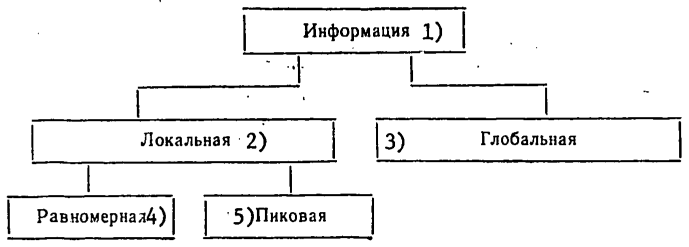


Figure 9. Classification of Information

Key:

1. Information	4. Steady
2. Local	5. Peak
3. Global	

Steady local information must be processed (cf. fig 8) by computing complexes 1 to 7, installed at those entities where it is developed and used. Large, small or microcomputers can be installed here depending on the amount of this information. As far as peak local and global information is concerned, it must be processed in computing complexes 8 to 10, the size, number and locations of which are determined by a feasibility study, taking into account the cost of equipment and communications channels, maintenance costs, reliability and efficiency requirements, etc.

As a result a set of user complexes, 1 to 10, originates, located at the necessary geographic points and interacting with one another (cf. fig 8) via a data transmission network.

4. Local Networks

Local computing networks, which can operate both independently and can represent users of large networks, have taken on ever greater importance. Here preferred development has been given to networks in which (cf. fig 10) a single packet-switching center is executed in the form of a monochannel to which network interface modules are connected. In turn, each module is connected to one or more user complexes. We will call a network of this sort a monochannel network.

Depending on the topology of the computing network, the required data transmission speed and the traffic, the monochannel is a twisted pair of telephone wires, a line, a coaxial cable, a light conductor, or the ether (radio). The length of a monochannel in small networks uniting a group of instruments or microcomputers is not greater than one meter. In large local networks containing dozens and sometimes hundreds of computers the length of a monochannel can reach several kilometers.

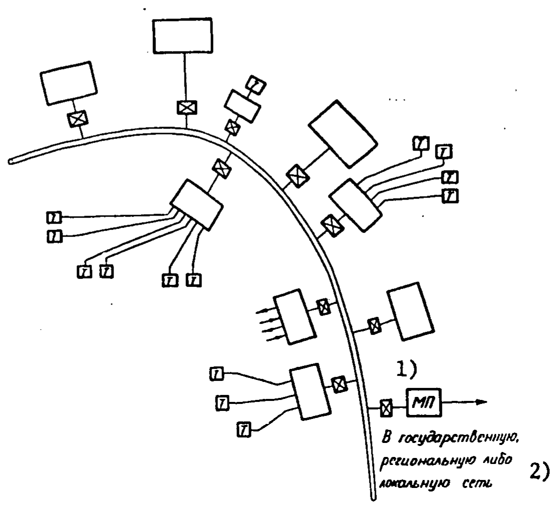


Figure 10. Monochannel Computing Network
Conventional symbols:

Working complex

Terminal complex with terminals

Terminal complex with measuring or control equipment

Network interface unit

Internetwork converter

Key:
1. MP [internetwork converter] 2. To State, regional or local network

The network interface module is most often created on the basis of a microcomputer and has the buffers required for transmitting and receiving packet sequences. This makes it possible to accomplish the transmission of information through a monochannel at a rate different from the data input/output rate in user complexes.

The method of access to the monochannel is of great importance in an independent computing network. With a small number of user complexes the network can have a supervisor unit which according to a selected cycle assigns complexes the right to occupy the monochannel. However, with a considerable number of user complexes this method becomes extremely inefficient. Therefore, other methods are employed in modern monochannel networks.

The most simple method of access to a monochannel, called in [8] the "classical," is governed by the following rules: 1) the network interface unit enters in each packet a control sum making it possible to check the contents of this packet; 2) the packet is transmitted through the monochannel to the network interface unit addressed as soon as the need for this arises; 3) the network interface unit addressed checks the packet received and informs the transmitting unit if there are errors in this packet; 4) with the existence of errors the packet is transmitted again upon the expiration of a randomly selected time interval.

Errors can appear in a packet because of the malfunctioning of equipment, noise, or, chiefly, the transmission into the monochannel of packets by several network interface units at once. The repeated transmission of packets over various time intervals makes it possible to eliminate conflicts which arise.

Losses of time for the repeated transmission of packets with heavy traffic can be rather great. Therefore, a number of methods of improving the "classical" method exist.

Considerable savings are produced by using the "slot" method, whereby a packet can be transmitted into the monochannel only upon the appearance of synchronization pulses from a central clock. The number of conflicts between transmitting units is reduced drastically if they also fulfill the rule: "listen to the monochannel all the time and do not transmit if it is occupied."

Experience has demonstrated that the creation of monochannel computing networks makes it possible to ensure the efficient processing of information at an enterprise, in an institution, at a scientific institute and in an industrial association.

Conclusion

Thus, a computing network is a distributed multicomputer association designed for gathering, storing, transmitting, processing and outputting any kind of information relating to the control and functioning of various sectors of the national economy. A network can involve the processing of a wide range of data or can be a special-purpose one designed for solving a narrow range of problems.

The size of a computing network is determined by the problems entrusted to it. Therefore, on one hand, it can include hundreds and even thousands of computers distributed over a large territory embracing several countries and, on the other, it can be local and have a total of a few small computers.

The main tasks performed in a computing network are as follows: the storage of data and the performance of data search operations; the mathematical modeling of processes, machines, equipment, natural phenomena and other entities; planning and the managerial control of various elements of the national economy; controlling production technological processes; the performance of bookkeeping and statistical reporting for the functioning of production units; control of scientific research and technical progress; the performance of computing (calculation) work; the distribution of stock; carrying out electronic library exchange work; controlling the operation of savings banks; controlling civil aviation; ticket sales and reservations for various kinds of transportation; controlling a network of trade centers; controlling energy resources; etc.

The users of a computing network, depending on the nature of the information in it, are production workers, management apparatus coworkers, scientific associates, and broad elements of the population. All the necessary methods of interaction between users and computing complexes are provided for in a computing network.

Bibliography

1. Packet Switching Report. Logica Ltd, September 1978. 340 p.

2. Drozhzhinov, V.I., Ilyushin, A.I., Myamlin, A.N. and Shtarkman, V.S. "Principles of the Design of the SYeKOP Experimental Multiple-User Computer Network" in "Voprosy kibernetiki. Problemy informatsionnogo obmena v vychislitel'nykh setyakh" [Problems in Cybernetics; Problems of Information Exchange in Computing Networks], Moscow, 1979, pp 18-33.

3. Marchuk, G.I., Kuznetsov, Ye.P., Moskalev, O.V., Metlyayev, Yu.V. and Efros, L.B. "O programme rabot po sozdaniyu vychislitel'nogo kompleksa (Tsentra) kollektivnogo pol'zovaniya v Novosibirskom nauchnom tsentre SO AN SSSR" [Program of Work on Creation of a Multiple-User Computing Complex (Center) at the USSR Academy of Sciences Siberian Division Novosibirsk Science Center], Preprint No 130, Novosibirsk, 1978, pp 1-34.

4. Yakubaytis, E.A. "Arkhitektura vychislitel'nykh setey" [Architecture of Computing Networks], Moscow, Statistika, 1980, 278 pages.

5. Reference Model of Open Systems Interconnection ISO/TS, 97/SC, 16, N 227. August 1979. 181 p.

6. CCITT Sixth Plenary Assembly. Geneva, 1976. Orange Book. Vol. 8. 2. Public Data Networks. Geneva, 1977. 217 p.

7. Public Data Networks. Eurodata Foundation. Lutyens House. Finsbury Circus, London, EC 2M 71.Y. 1979. 233 p.

8. Cotton, I. Technologies for local area computer networks. - Computer Networks, 1980, N 4, p. 197-208.

8831
CSO: 1863/161

UDC 681.324

DATA TRANSMISSION SYSTEM (SPD) OF THE 'EL'BRUS-1' MULTIPROCESSOR COMPUTING COMPLEX (MVK)

Riga AVTOMATIKA I VYCHISLITEL'NAYA TEKHNIKA in Russian No 2, Mar-Apr 81 (manuscript received 25 Apr 80) pp 15-30

[Article by V.S. Burtsev and V.I. Perekatov]

[Text] It is obvious that the intense development of teleprocessing is now at the stage when the term "fourth-generation teleprocessing system" must come in use as applied to large-capacity computing complexes and large computers.

Experience gained in creating the fundamentals of the "El'brus-1" MVK teleprocessing system and acquired as of the present time as the result of its development makes it possible for us to formulate certain traits which distinguish the fourth-generation system.

1. A teleprocessing system must represent a combination of general-purpose software and hardware making it possible to create practically any teleprocessing subsystem, including one whose key features were actually not drafted at the moment of the creation of these facilities.

2. The existence in the complex's structure of a problem-oriented program-controlled processor which solves a considerable portion of teleprocessing problems.

3. The modular structure of the special-purpose processor. Modularity is understood in the broader sense than simply the division of the processor into programmable and nonprogrammable facilities. It is possible to speak of "vertical" and "horizontal" modularity. The former assumes the existence of a certain set of functional modules making it possible to develop a processor for solving one more class of problems, e.g., a terminal module, a data transmission line module, a transport level line module, a rapid-channel module, a module for controlling peripherals and the like.

Horizontal modularity assumes the use in the structure of the processor of a variable number of modules of a single type (including modules of other types). Usually this number is determined by the number of communications lines serviced and by the rate of the information flow.

4. Teleprocessing software, including the software of the special-purpose processor, is designed on the basis of problem-oriented high-level languages.

The distinctive features cited do not pretend to completeness. They have been formulated chiefly in order to demonstrate that teleprocessing systems have entered a qualitatively new stage of development and new trends require their own generalization.

Teleprocessing systems characterized by the properties indicated above have opened up great possibilities. In particular, the question of the ability to use an MVK in a specific network architecture--closed- or open-ended--has been eliminated. By using universal hardware and software which are invariant with respect to the principle of organization of the network, system personnel can adapt an MVK to a network of any type. The problems which arise with this are of an ordinary technological nature and are to a great extent administrative.

The domestic system characterized by the fundamentally new features of fourth-generation teleprocessing is the "El'brus" MVK data transmission system. Able to serve as an example of a foreign fourth-generation system is the DATACOM system implemented in the B6700-7700 series of the Burroughs firm.

Fundamental Concepts

As the result of an analysis of the functions and operating features of an MVK SPD, two concepts have arisen which have been responsible for the beginning of a fundamentally new development.

The first of these--the concept of continuous adaptation--is based on the following considerations. The list of remote users of such a large computing facility as the "El'brus-1" MVK includes equipment of the most varied classes--from teleprinters operating in terminal systems to computing complexes interacting in computer networks. A characteristic feature of large teleprocessing systems is their practically continuous development--the elimination of obsolete components and the addition of new ones, the modification of technical and logical attributes, the modification of interaction algorithms, and the like. In connection with this, the adaptation of an MVK to remote users is regarded not as a one-time event, but as the way of life of the complex--a steady step-by-step process.

The second concept is the concept of primary processing. The fact is that in a number of applications an MVK is in the guise of a large in-line center for the processing of information arriving from communications lines. Fairly often the operations which thereby are performed on each received bit of information (e.g., a character) are independent, i.e., do not require the use of data received earlier. A typical example is the screening of data, which is carried out according to some simple criterion. In other cases processing is possible only with the presence of an entire group of received data (e.g., calculation of an arithmetic mean), but it can be performed gradually as each successive bit is received (the summing of data) and can be concluded immediately after reception of the last bit. Therefore, if the information load is considerable and the data transmission rate is relatively slow, it is pssible to improve the efficiency of an MVK by adding to its structure uncomplicated special-purpose equipment (a communications processor) which processes the next bit of information received regardless of whether the following bit is received.* In certain cases this equipment can, as reception takes place, "without hurrying" perform the entire computing task entrusted to the MVK as applied to these data.

In other words, primary processing should free the complicated equipment of the complex's central processors from the constant execution of simple operations on small-format operands; this work considerably reduces the equipment utilization factor.

On the basis of these concepts decisive practical decisions were made and implemented: 1) to develop a programming language oriented toward the teleprocessing base level and universal within the framework of the problems of this level; and 2) to add to the MVK's structure, as the equipment executing the base level's routine, a special-purpose data transmission processor (PPD).

The language has been given the name SETRAN (for "setevoy translyator" [network translator]). It makes it possible for a system programmer or user to specify the entire network of remote MVK users, as well as the data and routines required for implementing interaction with them, e.g., the line discipline, the algorithms for exchange with a terminal, the variant of the LAR-2 procedure, a specific primary processing process, etc.

To data transmission processors (specifically to their adapters) are connected communications lines which connect the complex with remote users. Thus, the network is, as it were, broken down into several fragments, each of which is served by its own PPD (cf. fig 1). The SETRAN translator routine is executed in the complex's central system. The program describing the entire network is converted by the translator routine into several code files, each of which represents the operating system (OS) of one of the PPD's. A specific OS is designed to serve the network fragment of its "own" processor. Upon initiation it is loaded into the local working storage of the PPD and the processor becomes adapted to the fragment.

Thus, for the personnel responsible for the creation of a specific teleprocessing subsystem for the "El'brus" MVK the problem is reduced basically to an adequate description of the network and of the basic teleprocessing processes in the SETRAN language. This made necessary the very careful coordination of the three key components: the language, the operating system procedures of the data transmission processor, and the PPD equipment. Actually, development took place in integrated fashion, which made it possible to optimize all three components.

*This processing principle has been given the name "conveyer" processing. With regard to transmission the conveyer principle can be formulated in the following manner: Each processed, i.e., prepared for transmission, bit of information is sent into the communications line regardless of whether the following bit has been processed.

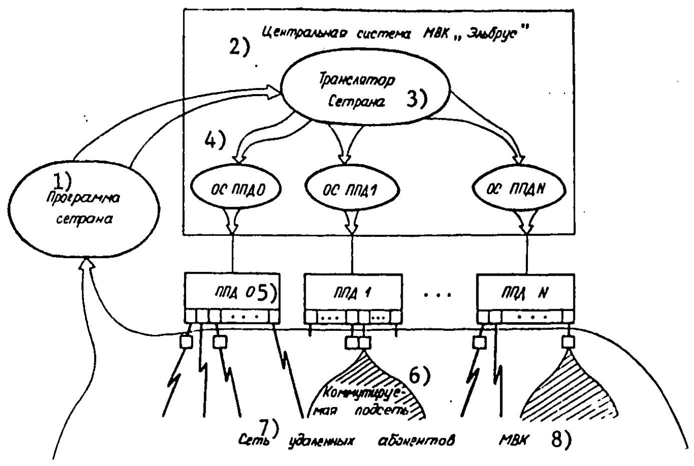


Figure 1. Principle of Adaptation of the "El'brus" MVK to a Network of Remote Users

Key:

1. SETRAN program	5. PPD0
2. Central system of "El'brus" MVK	6. Switchable subnetwork
3. SETRAN translator routine	7. Network of remote users
4. OS PPD0 [operating system of data transmission processor No 0]	8. MVK

SETRAN: Concept of the Programming Language

The SETRAN program consists of three sections which are described in succession: the global description section, the network section and the algorithm section. It is best to begin a first acquaintance with the language with the network section.

Network Section

In this section is described the combination of data processing modules participating in the MVK's operation through communications lines. A software-controlled remote unit interacting with the MVK through a communications line (in particular, a terminal) is an "external unit."

The second type of unit is the "line." We know, for example, that in working with terminals connected to a multipoint line it is necessary to perform operations for the line as a whole; an example of these operations is polling. In this sense a line can be singled out as a distinctive data processing unit coupled in a definite manner with external units--terminals.

System numbers of adapters, i.e., points for the connection of communications lines performing the role of contacts between the network and the complex, are assigned for units of the "line" type.

In the particular case of a single-point line the concept of an external unit and line unit often merge. Here the line unit can be used for software control of the cluster formed by the remote unit and the communications line connecting it with the complex.

Problems in basic teleprocessing sometimes reveal the need to control lines. For example, the information flows of two communications lines duplicate one another for the purpose of reliability; therefore, only one of them, to be selected according to a specific algorithm, must be admitted to the central system. A third type of unit--"internal units"--is added for problems of this kind. These can be regarded as virtual data processing modules located in the MVK itself.

Modules assigned in the network section form structures called "trees." In the structure of a tree one unit can be "nested" in another and, on the other hand, include several nested modules in itself. The implementation of SETRAN in the "El'brus" MVK permits six nesting levels.

As an example it is possible to take the section of a network illustrated in fig 2a. It includes four communications lines connected with the adapters of a single PPD having the system numbers 0:1, 0:7, 1:4 and 1:7. Through the first line the MVK interacts with a remote unit, TERM. To the second line are connected three terminals, POLTERM. The third and fourth lines are connected with information sources which duplicate one another. Therefore, the necessity arises of adding to the software a module which functions as a flow selector.

In the network section this section can be represented by three trees whose structure is shown in fig 2b. The first of these, a confluent tree, includes a single unit. The corresponding entry in the program has the following form:

```
* MODULE  # 0:1 TERM                     % Line module
  . . . . . .
  OVER
* MODULE 3  # 0:2 TREKHTLINIYA           %* Line unit.  After the procedure word
 [THREE-POINT LINE]                      MODULE is entered the number of nested
                                         modules--3. %*
  . . . . . . . . . .
     MODULE POLTERM1                     % External module
     . . . . . .
     OVER
     MODULE POLTERM2
     . . . . . .
     OVER
```

[Continued on following page]

```
        MODULE POLTERM3
        . . . . . . .
        OVER
    OVER
   *MODULE 2 SELEKTOR [SELECTOR]            % Internal module
    . . . . . .
        MODULE  # 1:4 ISTOCHNIK1 [SOURCE1]
        . . . . . . . .
        OVER
        MODULE  # 1:7 ISTOCHNIK2 [SOURCE2]
        . . . . . . . .
        OVER
    OVER
```

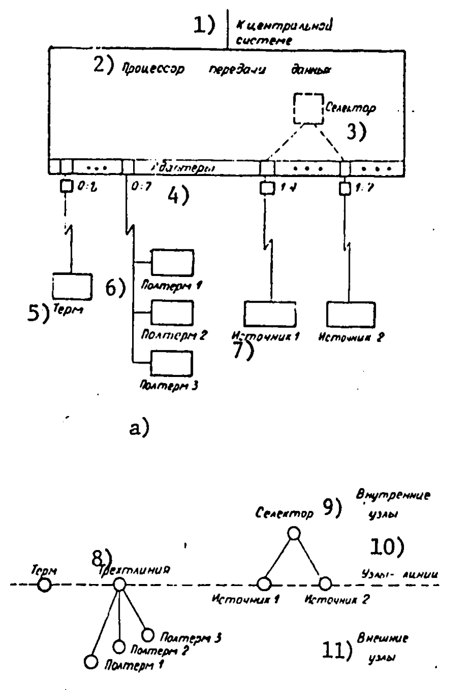


Figure 2. Principle of Representation of a Network of Remote Users in the SETRAN Program: a--real netwcrk; b--structure of modules in network section

[Key on following page]

Key:

1. To central system	7. ISTOCHNIK1 [SOURCE1]
2. Data transmission processor	8. TREKHTLINIYA [THREE-POINT LINE]
3. Selector	9. Internal modules
4. Adapters	10. Line modules
5. TERM	11. External modules
6. POLTERM1	

Here the dotted line replaces "contents of module." The fact is that each module is assigned specific program components. For example, for the TREKHTLINIYA module they could have the form:

```
* MODULE 3  # 0:7 TREKHTLINIYA
    MEMORY PAMLIN [MEMORY LINE] (POVTORY [REPEATS]: = 5)
    REGISTERS REG
    QUEUE OCHLIN [QUEUE LINE]
    PROGRAM DISPETCHER [SUPERVISOR] [OCHLIN, 1&&255];
           OPROS [POLLING]; VYBORKA [ACCESS]
    MODULE POLTERM1
           MEMORY PAMTERM [MEMORY TERMINAL]
           MESSAGE ZAG [HEADING] (SB)
           QUEUE OCHTERM [QUEUE TERMINAL]
           PROGRAM VVOD [INPUT]; VYVOD [OUTPUT]
    OVER
    MODULE POLTERM2
           KOP [COPY] POLTERM1
    OVER
    MODULE POLTERM3
           KOP [COPY] POLTERM1
    OVER
 OVER
```

This entry indicates that to the TREKHTLINIYA module are assigned the following: 1) its own memory, formed according to the PAMLIN mask determined in the global description section; 2) its own registers, formed according to the REG mask, also determined in the global description section; 3) the OCHLIN queue; 4) DISPETCHER, OPROS and VYBORKA routines whose texts are assigned in the algorithm section. These routines, of course, can be assigned to other modules; nevertheless, for the TREKHTLINIYA module they will be executed only in its context, i.e., will use variables from its own memory (registers) or its own queue; they are capable of activating another routine assigned by name only within the limits of their own module.

The context of any POLTERM module is broader--in addition to the contents of its own module it includes the contents of the TREKHTLINIYA module; generally the context includes the contents of all modules placed higher with regard to a branch of the tree. The contents of POLTERM modules include also a message (SB) with a heading which is formed according to the ZAG mask assigned in the global description section.

In the description of the POLTERM2 and POLTERM3 modules it is indicated that their contents are copied from the POLTERM1 module. This abbreviates the entry.

In the example given the contents of a module include one queue and one message, but SETRAN in principle makes it possible to assign to a module several queues or messages. It is possible to employ unidimensional arrays of queues and messages; these are especially convenient in organizing buffer networks.

Unneeded components in the contents of a module can be absent. For example, it is possible to assign an internal module containing only a memory, or only queues, or a memory and queues. It will represent a common memory and common queues for the routines of nested line modules.

Queues and routines (for the purpose of activation) localized in nested modules also have access to routines executed in the context of a specific module, but for this purpose special facilities of the algorithm section are employed.

It should be mentioned that routines run in the context of a line module and of any nested external module can be executed by receiving and transmission operators in conformity with the adapter assigned in the line module. Several adapters may be assigned in this module; accordingly, these routines can work with several communications lines.

Let the DISPETCHER [SUPERVISORY] program perform the decision functions in controlling a three-point line. This program must be activated if the central system orders some kind of operation (input or output) for one or more POLTERM terminals. These orders in the form of message units are attached to OCHTERM queues belonging to external modules. In addition, a message unit oriented directly to the line can arrive, such as for establishing a connection, or for disconnection or for performing a test. Messages of this kind are included in the OCHLIN queue. All conditions for activation relative to the state of queues are assigned for the DISPETCHER program in square brackets. They can be understood as follows: If a module's routine is not executed and a message is referred to a vacant OCHLIN queue or to any vacant queue of any module located on the first nesting level, then the execution of this routine must begin.*

The language provides for various methods of assigning individual activating queues and groups of them.

The second method of activation assumes unconditional running of the module's routine immediately after initialization of the PPD.

By means of the third method one routine is started by a statement which is executed by another. For example, the VVOD [INPUT] routine can be started from

*In the 1&&255 construction the 1 digit indicates "first nesting level" and the 255 digit, "any queue of the module."

the OPROS [POLLING] routine, and the VYVOD [OUTPUT] routine from the VYBORKA [ACCESS] routine.

Thus, a structure corresponding to a multipoint line is assigned in the network section. With the presence of a task oriented wholly to the line, a routine is started in the module which in the programmer's concept is associated with the communications line. Having established logical connection with a specific terminal, this routine starts the required routine in the module which is associated with the terminal, and the latter routine performs direct exchange.

Everything said above must be regarded only as an illustration of several concepts and possibilities of the language. For example, the presence of a memory in POLTERM modules is justified only in those cases when in a given exchange session the results of a preceding session are utilized in some way. If this is not so, then the routines of each external module can operate with the data of a single common memory made available to the terminals by turns. This is the memory of the TREKHTLINIYA module.

Similarly, a message in which characters received from each terminal are stored can be described in the TREKHTLINIYA module.

And, of course, it is possible to describe the entire structure in the form of a single line module, whereby switching between terminals is accomplished by indexing structures belonging to this module.

It is necessary to pay attention to one important factor. Having assigned the tree of the TREKHTLINIYA module in the network section and considering specific routines of the algorithm section in the context of modules of this tree, the programmer can completely separate himself from his surroundings (in this case the TERM and SELEKTOR modules) and assume that the entire task for the TREKHTLINIYA tree is performed by a certain virtual processor which operates with components which are described in the contents of modules. It attends only to communications lines connected to line module adapters. This principle, certainly convenient in solving problems at the basic teleprocessing level, was used for the first time in the NDL B6700 language.

The description of the network in SETRAN also provides for the possibility of dynamic reconfiguration--the working replacement of the contents of modules and the switching of some fragment of the network for servicing by a standby PPD.

Global Description Section

"Masks" are defined in the global description section. A memory mask is assigned in the form of a sequence of descriptions of simple memory variables and of unidimensional arrays of memory variables. On the basis of this description the language's translator routine creates a certain representation (mask) of the region of the memory in which representations of the variables to be described have been packaged. Character locations of the mask are initialized in keeping with the description of variables. If, then, in the network section the programmer assigns a memory mask identifier to certain modules, then the translator routine will actually isolate for each of them the physical region of the local working storage of the PPD formed according to the assigned mask. All variables whose representations are localized in this region will appear in the module's context.

The programmer can influence packaging by assigning the precise location of a simple variable in the mask. Direct addressing is necessary for the data transmission processor operating system's interface and for routines of the central system. With specific principles for the equipment's structure, it makes it possible to increase the efficiency of the data transmission processor on account of the simultaneous retrieval of several operands from the memory.

Relative addressing is also possible, when the representations of other variables are localized within the limits of a memory location storing a representation of an encompassing variable.

In fig 3, for example, is illustrated how the translator routine performs packaging on the basis of the description given. The programmer uses variables with the identifiers INFORMATSIYA [INFORMATION], TIPINFORMATSII [TYPE OF INFORMATION], NOMABONENTA [NUMBER OF USER] and NOMPEREDACHI [NUMBER OF TRANSMISSION] in forming a heading, but when the need arises of transmitting into the communications line a structure already formed, he operates with the encompassing variable ZAGOLOVOK [HEADING].

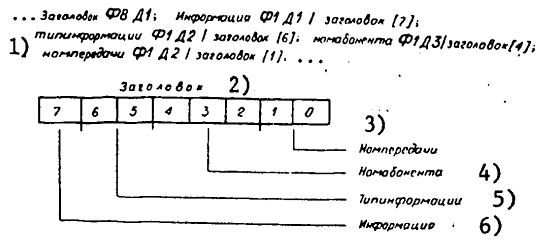


Figure 3. Principle of Combining Representations of Variables

Key:

1. ...ZAGOLOVOK F8 D1; INFORMATSIYA F1 D1 / ZAGOLOVOK [7]; TIPINFORMATSII F1 D2 / ZAGOLOVOK [6]; NOMABONENTA F1 D3 / ZAGOLOVOK [4]; NOMPEREDACHI F1 D2 / ZAGOLOVOK [1]; ...
2. ZAGOLOVOK
3. NOMPEREDACHI
4. NOMABONENTA
5. TIPINFORMATSII
6. INFORMATSIYA

The description can also not include addressing of a variable; in this case the location of the representation of the variable in the memory is determined by the translator routine.

A special type of simple memory variable is the semaphore. It is highly probable that a variable of one and the same mask must be initialized differently in the context of various modules. Let, for example, the maximum permissible number of repetitions with the unsuccessful transmission of a polling (or accessing) sequence be assigned by a variable with a POVTORY [REPEATS] indentifier from the PAMLIN mask. In view of the varying quality of communications lines, the number of repeats must be selected individually for each of them. This "tuning" of one or more variables of a mask as applied to a module can be carried out in the network section. Above in describing the TREKHTLINIYA module we wrote

... MEMORY PAMLIN (POVTORY: = 5) ...

This can be understood as follows: the isolation for the module of a memory formed according to the PAMLIN mask, but with this the variable with the POVTORY identifier must be initialized not by the value assigned for it in describing the mask, but by the value 5 .

A second type of mask is "message masks." A message is a region of the memory which is used for setting up interaction between routines run in modules of a single tree, as well as for interaction between any of them and routines of the central system. Having formed a message, one of the interacting routines transmits it to another; as a rule, this is done through a queue playing the role of a buffer between asynchronous processes.

A message usually includes a heading and a text section. In the typical case of a message unit for readout (from the central system) the text portion contains the text to be transmitted and the heading, certain parameters of the transmission procedure; in the case of a message-result concering read-in (to the central system) the text portion contains the text received from the line and the heading represents the hard-copy log of the communications section. Messages containing only a heading are often used.

For the SETRAN programmer the "message" is the message's identifier. Actually, the message identifier equals the message descriptor--a location of the local working storage with a fixed address, which contains a reference to the message itself and some information regarding it. Thus, under the same name the programmer can process an entire flow of messages keyed to the descriptor at various times.

In base level problems, as a rule, it is necessary to form or analyze a message heading. The programmer does this by using "message variables," whose representations are localized in the heading of the message keyed to the descriptor at a given moment. The SETRAN program can process messages with various heading structures; the only limitation consists in the fact that messages with an identical structure are to "pass" through a given descriptor.

The message mask, which includes descriptions of message variables, assigns the structure of the heading. In the network section each message identifier is assigned a message mask identifier (e.g., above for the POLTERM1 module we wrote ...MESSAGE ZAG (SB)... , where ZAG is the mask identifier and SB is the message identifier).

Thus, the translator routine "knows" how to gain access to the heading (via the descriptor) and how the region of the memory occupied by the heading is structured (from the description of the message mask); this makes it possible to perform operations with components of the heading described as message variables.

In communications processors is ordinarily used nonprogrammable line equipment which links the computer with communications lines.* It is at the lowest functional level of data transmission: It interacts directly with communications terminal equipment (in particular, with modems), it enables the required duration of unit intervals, it gathers bits into characters in reception, and, on the other hand, breaks up characters in transmission, it eliminates (adds) start and stop pulses, and the like. Usually its mode of operation with a given communications line (adapter) is determined by the state of several registers accessible to a programmable computer.

The software facilities of these registers are assigned in a "register mask" in the form of a sequence of descriptions of "register variables." The register mask identifier is indicated in the contents of the line module after the procedural word REGISTRY [REGISTERS]. By operating with register variables, any routine of the line's tree can control the state of line equipment as applied to "its own" adapter.

In the register mask are also assigned virtual registers which lack a hardware equivalent; their functions are implemented by the computer through interpretation. In other words, register variables are used in individual cases as a means of "soldering" into the software those functions which were not added to the line equipment because of a lack of opportunity or simply on account of shortsightedness.

It must be mentioned that in addition to memory, message and register variables a "special variable" has been defined in the language. It is designated by the procedural word SIMVOL [CHARACTER] and represents the software facility for the location through which exchange between the computer and line equipment takes place.

Each adapter is assigned its own special variable, to which is implicity assigned the representation of characters received through the adapter in execution (as applied to the adapter) of the reception statement; thus they become program accessible. In execution of the transmission statement the program-prepared representation of the special variable is implicity sent through the adapter as the next character to be transmitted; this operation can be performed more than once in a single statement.

The constant mask contains descriptions of simple constants and of unidimensional arrays of constants. In the network section each PPD is assigned an identifier

*This equipment is present also in the "El'brus-1" MVK PPD. It is called a group interface unit (GUS).

of a certain constant mask. As the result of the constant of this mask, any routine of the module run in a given PPD can be used.

The "mask" concept as applied to constants is rather arbitrary, for variants are possible in which a constant mask cannot be identified with the structure of a memory region or of registers. For example, in the first implementation of SETRAN the representations of simple constants whose word length is not greater than four bytes remain in the translator routine's tables and during generation are represented directly in the instruction code. However, in the typical case the constant mask includes descriptions of arrays and of long constants. Their representations are packaged in the local working storage; therefore, the translator routine, in passing a description of the constant mask, creates a character pattern of the region of the memory occupied by constants. The region of the memory structured according to this mask is actually isolated in the data transmission processor in translation of the network section.

In data transmission problems simple constants are especially important for designating long strings and characters which do not have proper graphics.

Constant arrays can be described in one of two forms in SETRAN. According to the first form all elements of the array are described in the form of a list. Thus it is convenient to add to the program code translation tables whereby the translation itself will be performed by an ordinary assignment statement, e.g.:

```
CHARACTER: = K.DKOI7[SIMVOL];
```

here K.DKOI7 is the constant array with the identifier DKOI7 from constant mask K. The array represents a table of translation from DKOI to KOI7.

In the case of the second form, instead of a literal indication of elements of the array is written the equation by which they are computed. It obligatorily contains a variable "input element" (VKh) for which a range of variation is assigned. Elements of the array are generated by the translator routine, substituting in the equation the next value of the input element lying within the limits of the range. Thus, the program in a number of cases does without routine work relating to the computation of factors to be assigned analytically.

For example, let it be required to find the remainder from dividing (modulo-two) by a generating polynomial a certain eight-bit code linked on the right with 16 zeros.

With the step-by-step method of cyclic anti-interference coding this must be done after the reception of each character with a newly formed eight-bit code. But it is possible, having sacrified the memory, to form a table of remainders for all possible eight-bit combinations. If the generating polynomial is given in the same mask in the form of a simple constant with an identifier of X16X12X5X0, then for the purpose of generating the table it is sufficient to describe the following array:

```
...[FROM 0 TO 255] OSTATKI [REMAINDERS] F8D2 = VKh&4<<0000>>MODMOD
                                             X16X12X5X0;...
```

The last component of the global description section is descriptions of line events. A "line event" is the software form of an interrupt transmitted by the line equipment computer. It can correspond to an actually generated interrupt or to some virtual interrupt created by interpretation by a "soldered in" module of the operating system.

Algorithm Section

The texts of module routines are described in the algorithm section. The addition to the language of subroutines and functions has been worked out and this has not caused fundamental difficulties. However, in the first edition of the translator routine subroutines and functions were not implemented because of the great overall amount of work.

Included among the number of algorithm section facilities which can be utilized in writing a module program are primarily universal constructions such as structures and situations, conditional and selection (CASE) statements, synchronization statements and somewhat developed Dykstra primitives.

Operations of 10 priorities have been defined for expressions. In addition to whole-number arithmetic operations, a number of logic operations have been added, as well as specific operations which are especially useful in data transmission problems, such as a shift (in particular, a cyclic), inversion, linking, embedding, division and modulo-two addition and formation of a parity-check sequence.

In computing expressions is employed the basic statement of the language according to which each construction used as a primary simultaneously has a "representation" and a "value." The representation of a construction is the continuous set of binary digits assigned to it: 0 and 1. The value of a construction is the value of the number assigned to it. By a number is meant a positive whole number or zero.

This dualism, making it possible to apply any operation (in particular, arithmetic and logical) to any construction, is quite fruitful in base level teleprocessing problems. It is possible to cite many examples. Let us choose the most simple.

Let the character received be stored in a memory location "belonging" to the single-byte variable P.SIMVOL (i.e., to a variable with the identifier SIMVOL [CHARACTER] from mask P). The character enters the communications line in KOI7 code. It is required to check whether it is the control character KB (KONETS BLOKA [END OF BLOCK]), whose code is assigned by the single-byte simple constant K.KB. If it is, then certain operations are to be performed. Otherwise, the character is to be translated into DKOI code and other operations are to be performed.

It will look like this:

```
IF  P.SIMVOL EKV [EQUIVALENT] K.KB
THEN. . . . . . . .
OR ELSE  P.SIMVOL:= K.KOI7D[P.SIMVOL]
. . . . . . . . . . . . . . . . . . . . . . . . .
OVER;
```

In the condition is employed the logical operation "equivalence" which checks the identity of the codes of both operands. It works with the representation of the P.SIMVOL variable.

In the OR ELSE alternative the translation of codes takes place. The translation table is the K.KOI7D constant array. In indexing the P.SIMVOL variable array we use its value. It makes it possible to select the element of the array corresponding to the same character in DKOI code.

Another example. Let there be organized on the basis of the byte variable P.SCHETCHIK [COUNTER] a modulo-16 counter which registers the number of the information block to be transmitted. This number must be transmitted to the user before the block in DKOI code.

In computing the current value of the counter an arithmetic addition operation is performed, utilizing the value of the variable P.SCHETCHIK:=*+1; but conversion into DKOI code is performed by means of logical addition of the variable with a string having the code 11110000:P.SCHETCHIK:=*OR <<11110000>>. Here the representation of the variable if utilized.

A number of operators and standard functions make it possible to arrange for working with queues and messages: to attach and detach a message, to transfer all messages from one queue to another, to obtain information on a message and queue, to move the indicator of the current byte in a message, to store the next received character in a message byte or to select the next message byte for transmission, to interrogate a region of the memory for a message (descriptor), and to transmit a message to routines of the central system or to the free memory region bank.

This software is not fundamentally new--it is included in one form or another in DC-ALGOL B6700, for example. In SETRAN some of it has been considerably developed or modified; that which did not require modification was taken without changes.

The language necessitated the addition of several specific statements and standard functions. Included in their number can be standard functions making it possible to work with queues and routines of nested modules, or the standard function making it possible to convert a DKOI string into a binary number and vice-versa.

SETRAN is oriented toward the conveyer method of data processing. Therefore, a special position among specific facilities of the algorithm section is occupied by the reception operator and transmission operator. By utilizing them the programmer organizes work with communications lines, which is in the final account the key function of the SETRAN program and of the data transmission processor.

Reception and transmission operators are fairly complicated; therefore, within the scope of this article it is best to demonstrate their function in a brief example.

```
TO NETZAGOLOVKA [NO HEADING], PEREPOLNENIYE [OVERFLOW], OSHCHETNOSTI [PARITY
ERROR], OSHKSB [BLOCK CONTROL CHARACTER ERROR], APDNEVNORME [DATA TRANSMISSION
EQUIPMENT NOT UP TO STANDARD]
     % *The reception and transmission operators are combined.  It is possible to
        build structures on their basis.  Here is given is description of situa-
        tions % *.

RECEIVE [1] MESSAGE SB [FROM P.NACHSEGMENTA [BEGINNING OF SEGMENT] TO P.KONSEGMENTA
[END OF SEGMENT]]
     % *Reception is carried out from the adapter having a relative number of
        1 in the list of all line module adapters.  Then within the framework of
        the same operator characters received will be stored in the message
        segment whose boundaries are assigned by the value of the variables
        P.NACHSEGMENTA and P.KONSEGMENTA.  Thus, it is possible to apply simultan-
        eously to a single message several reception operators filling various
        segments % *.

(SBOYAPD [DATA PROCESSING EQUIPMENT MALFUNCTION], KTOTAM [WHO THERE], DETEKTOR-
KACHESTVA [QUALITY DETECTOR], CHETNOST' [PARITY])
     % *These are "foreseen line elements," i.e., interrupts which can arrive from
        the line equipment instead of the next received character.
        Entered receiving medium. % *

* FOR P.ETALONZAGOLOVKA [STANDARD HEADING]
  CYCLE
  . . . . . . . .                          % *The dotted line indicates omitted
                                           constructions. % *

: ...NETZAGOLOVKA! [NO HEADING]
REPEAT;
     % *Operator for start-stop cycle with a heading.  Its function is to compare
        the sequence of characters (headings) received from the communications
        line with a certain standard, which in our case is assigned by the repre-
        sentation of the P.ETALONZAGOLOVKA variable.  Each step of the start-stop
        cycle begins with the reception of the next character.  At the end of the
        cycle's medium the character received (possibly transformed in the cycle's
        medium) is compared with the corresponding byte of the variable.

        With a lack of agreement the sequence of operators specified after the
        colon is executed; in our case the operator for a structural change with
        the NETZAGOLOVKA situation completes the execution of the entire reception
        operator.  If the entire heading is received successfully, then the start-
        stop cycle operator is ended naturally % *.

TO VES'BLOK [ENTIRE BLOCK]
* CYCLE
```

```
     % *The simple start-stop cycle operator which will receive the information
        block is begun. The structure is created on its basis; the VES'BLOK
        situation is described % *.

. . . . . . . . . . . . . . . . . . . .

IF SIMVOL [CHARACTER] EKV [EQUIVALENT] K.KB THEN VES'BLOK! [ENTIRE BLOCK] OVER;
     % *In case of the reception of a character equivalent to the constant
        K.KB, we leave the cycle according to the VES'BLOK situation % *.

. . . . . . . . . . . . . . . . . . .

STORE |PEREPOLNENIYE! [OVERFLOW]
     % *The character received is stored (possibly processed in the medium of the
        start-stop cycle) in the "current byte" of the SB segment. If the segment
        is already filled, then a structural change is performed in accordance
        with the PEREPOLNENIYE situation and the reception operator is concluded % *.

REPEAT;
DELAY;
     % *Start-stop delay operator. The routine stops before reception of the next
        character % *.

IF CHARACTER ⌉ = P.KSB THEN OSHKSB! [BLOCK CONTROL CHARACTER ERROR] OVER
     % *The character received is a block control character (KSB). It is compared
        with the P.KSB variable "in which" the KSB of the block received is
        generated. With lack of agreement a structural change in accordance with
        the OSHKSB situation takes place and execution of the reception operator
        is concluded; otherwise, the reception operator is concluded naturally % *.

(SBOYAPD : APDNEVNORME!,[(DATA TRANSMISSION EQUIPMENT MALFUNCTION : DATA TRANSMIS-
SION EQUIPMENT NOT UP TO STANDARD!,]
     % *The "reaction to line events" entry begins. With the origin of a SBOYAPD
        line event a structural change is performed according to the APDNEVNORME
        situation and the reception operator is concluded % *.

DETEKTORKACHESTVA [QUALITY DETECTOR]: . . . . . . . . . . ,
     % *With the origin of a DETEKTORKACHESTVA line event a certain sequence of
        operators is executed and the program returns to the same point of the
        reception medium where it was before the origin of the line event, i.e.,
        awaits the next character % *.

CHETNOST' [PARITY]:. . . OSHCHETNOSTI! [PARITY ERROR]
     % *In case of an error in parity of the character received a certain sequence
        of operators is executed and reception is concluded as the result of a
        structural change in keeping with the OSHCHETNOSTI situation % *.

VSEPM [RECEPTION OVER]
     % The reception medium and the entire reception operator are completed.
```

```
WITH NETZAGOLOVKA: [NO HEADING]. . . . . . . . . ,     % *Attachment of operation
     PEREPOLNENIYE: [OVERFLOW]. . . . . . . . . ,        in keeping with situations
     OSHCHETNOSTI: [PARITY ERROR]. . . . . . . . ,       concluding execution of
     OSHKSB: [BLOCK CONTROL CHARACTER ERROR]. . ,        the reception operator % *.
     APDNEVNORME [DATA PROCESSING EQUIPMENT
      NOT UP TO STANDARD]:. . . . . . . . . . . ,

VSESIT [SITUATIONS OVER]
```

The transmission operator is practically symmetric; by employing the start-stop cycle and start-stop delay operators and other facilities of the algorithm section in its medium it is possible to describe the transmission of a complicated information structure.

Data Transmission Processor

In keeping with the general principle of modularity maintained in the "El'brus" MVK, each specific complex, depending on the number of communications lines served and the rate of the information flow, can contain up to 16 PPD's [data transmission processors]. The data transmission processors interact with the central system of the MVK on the basis of a common working storage. The PPD gains access to the central working storage through an input/output processor (PVV) which controls all external information flows of the MVK. Up to four PPD's are connected to a single PVV (cf. fig 4).

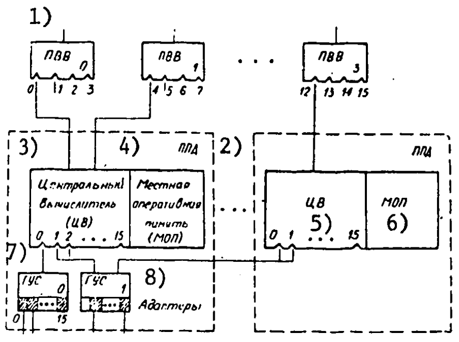


Figure 4. Structure of "El'brus" MVK Data Transmission Processor

Key:

1.	PVV	5.	TsV
2.	PPD	6.	MOP
3.	Central computer (TsV)	7.	GUS [group interface unit]
4.	Local working storage (MOP)	8.	Adapters

The data transmission processor can be connected directly to two PVV's. At a given moment it works only with one of them, but when a PVV goes out of order the information flow of the data transmission processor can be switched immediately to a standby PVV.

The data transmission processor has a modular structure. It consists of a central computer (TsV), a local working storage (MOP) and group interface units (GUS's) whose number is determined by the number of communications lines served.

Group Interface Units

The key functions of a GUS are as follows: to receive data and convert it into character form for subsequent processing by the SETRAN program being run in the central computer; in transmission its function is the opposite--to convert characters prepared by the program to a form in which they are to be transmitted through the communications line and then to ensure their entry into the line.

The transmission of data is usually performed in a serial bit-by-bit manner; therefore, the function of a GUS can be defined more concisely: the assembly of characters in reception and their disassembly in transmission. Assembly-disassembly algorithms are determinate; only parameters can be changed--the type of data transmission (synchronous/asynchronous), the rate (bits/s) and the character format. Accordingly, a GUS is designed as a nonprogrammable unit in which the parameters of the "soldered in" algorithms are assigned by the central computer, i.e., by the SETRAN program working with register variables.

The structure of a GUS includes 16 adapter positions. An adapter of any type can be installed at any of them. Designwise an adapter represents a single standard element of the "El'brus" MVK. At the present time adapters of three types are used: 1) a telegraph, for direct interfacing with the line (an S1-TG interface); 2) the S2-100 (V24 series 100) interface adapter; and 3) a parallel interface adapter (S3 and S2-200). All of these perform only the functions of an electrical interface with the line or communications terminal equipment. The Kh-25 adapter, in which some logic is provided, is under development.

An adapter position is tuned to a specific mode by installing the appropriate adapter and by means of software tuning of control fields which assign exchange parameters.

The serial synchronous mode makes possible operation with speeds up to 9600 bits/s. An increase in speed is possible, but the maximum permissible number of adapter positions in a GUS is thereby reduced. Two character formats--a main and additional--are worked with directly. The additional format is used for the remainder of a coded program in cases when its total length is not a multiple of the word length of the main format.

One of the variants of the synchronous mode is designed especially for the cycle phasing procedure. With a steady reception mode there is a variant with the automatic exclusion of synchronization characters and with the transmission mode a variant with the automatic generation of these characters. The code of a synchronization character can be any one--it is assigned by the SETRAN program.

The serial asynchronous mode is implemented for the range of standard speeds from 50 bits/s to 4800 bits/s and for all start-stop formats used in practice. Its distinctive feature is the automatic (without the participation of the central computer) readout of reception and transmission time-outs between characters. The length of a time-out is assigned by the program.

The generation of an OTBOY (BREAK) pulse represents a separate mode. Its duration is software controlled.

Possible emergency and nonemergency situations which arise in reception and transmission result in the generation of special interrupts transmitted to the central computer. They are processed as line events in the program.

To GUS's have been added special gear making it possible to implement a program-controlled interface with communications terminal equipment (e.g., a modem). By their means the control of interface circuits has been brought up to the level of the SETRAN program, in which the appropriate procedure, in particular, the establishment of a connection, is described.

A single group interface unit can be connected directly to two central computers. In case one central computer goes out of order the information flow of the GUS can be switched to a standby by means of special reconfiguration routines.

Central Computer

The central computer is a special-purpose processor designed basically to process information flows according to the conveyer method.

A main feature of this equipment is the fact that it was developed under the great influence of the conceptually combined software, i.e., was designed for its implementation from the beginning. Here special attention was paid to character-by-character processing--the execution of standard operations which inevitably or often correspond to the processing of each received or transmitted character.

The hardware-controlled change of context must be singled out here. The fact is that operation according to the conveyer method implicitly assumes the recurrence of routines carrying out character-by-character processing. It is made possible by the fact that for each communications line (adapter) is created a data module making it possible to organize execution of the routine in the context of precisely this line. Let us call it a "line context module" (BKL). A standard component of a BKL is the address of the instruction from which it is necessary to begin execution of the routine with the arrival of the next character.

Ordinarily all BKL's are stored in a local working storage in the form of a regular structure. The efficiency of the processor is improved markedly if the BKL for the line served is transferred to registers with a short access time during the processing of a character. Accordingly, after the completion of processing the BKL must be returned to the MOP from the rapid register zone. In implementations of the conveyer method known to the authors (e.g., in the DCP B6700-7700) the transfer of BKL's is performed by the operating system, i.e., is software controlled. This takes a considerable amount of time and inevitable overhead costs originate.

In the "El'brus" MVK PPD a change of line context is made possible by the hardware: The process of requesting and returning the BKL is performed automatically and coincides with processing of the next character. Thus, the central computer begins the operations with the next character without preparatory operations, immediately after completing processing of the preceding.

Software requirements have, of course, been reflected in the instruction set. The four-byte instructions of the central computer are oriented toward the execution of SETRAN operations on byte operands and operands representing a group of bytes. They make it possible to speed up access to program constructions localized in the local working storage and the storage of the central system and, in particular, to organize working with queues and messages. By means of specific instructions a very simple mechanism for requesting procedures has been implemented, making it possible to structure not too badly the object code of the SETRAN program.

Conditional transfer instructions are oriented toward the most typical software situations--the arrival from GUS's of interrupts of a specific type and code, a change in the state of flags and the like.

Advanced hardware control has been added to the data transmission processor, which practically eliminates the possibility of malfunctioning or failure without their instantaneous indication. As the result of an emergency situation the hardware automatically causes special reaction routines representing part of the operating system of the PPD. The program reaction to malfunctions makes it possible in principle to maintain the survivability of the PPD right up to extreme emergency situations.

Conclusion

Results achieved in the creation of the data transmission system representing the basis of the entire "El'brus" MVK teleprocessing system have been described in fairly complete form in this study. It, of course, will be subjected to changes in the process of gaining experience in using it. However, if allowance is made for the fact that the "El'brus" MVK SPD [data transmission system] is in the overall course of the development of fourth-generation teleprocessing systems, it can be considered a fundamental basis for the creation of further studies in this area.

8831
CSO: 1863/161

UDC 621.391:621.395:681.142

NEW BOOK DISCUSSES CONTROL SYSTEMS FOR COMMUNICATIONS NETWORKS

Moscow SISTEMY UPRAVLENIYA SETYAMI in Russian 1980 (signed to press 21 Jun 80) p 138

[Table of contents of book "Control Systems for Networks", edited by Professor V. G. Lazarev, doctor of technical sciences, and N. Ya. Parshenov, candidate of technical sciences, USSR Academy of Sciences, Izdatel'stvo "Nauka", 1,650 copies, 140 Pages]

[Text] Table of Contents

11176
CSO: 1863/137

UDC 621.391

HYBRID SWITCHING SYSTEMS

Moscow SISTEMY UPRAVLENIYA SETYAMI in Russian 1980 (signed to press 21 Jun 80) pp 52-62

[Chapter by V. N. Koshelev of book "Control Systems for Networks", edited by Professor V. G. Lazarev, doctor of technical sciences, and N. Ya. Parshenkov, candidate of technical sciences, USSR Academy of Sciences, Izdatel'stvo "Nauka", 1,650 copies, 140 pages]

[Excerpts] Work began in the late 1960's and early 1970's to build hybrid switching systems. The need to build integrated communications networks and hybrid switching systems arises, first, from the fact that the development of technology and data processing shows a growing need for information exchange among different kinds of users. In the second place, as studies [6, 9, and 10] show, integrated communications networks are more economical than other types. This economy is achieved by fuller use of expensive main channels.

In conformity with the recommendations of the International Telegraph and Telephone Consultative Committee, data transmission equipment is classified by transmission speed into the following groups: low-speed — 50, 100, and 200 bauds; medium-speed — 600 and 1,200 bauds (speeds of 1,800, 2,400, 3,600, and 4,800 bauds are also permitted); high-speed — 19,600 and 48,000 bauds and higher.

Information in a communication network may also have different characteristics. There are short messages transmitted in a mode close to real time, long messages in real time, and large arrays of data. Information transmission may be performed with improved reliability or without, with due regard for the priority of the user and information or without. The different characteristics of user terminals and information being transmitted require that communications networks have different characteristics and that the switching center have different functional features.

Thus, the future communications network should be suitable for serving different groups of users, which will require that the network have specific characteristics that differ from existing networks. These include a broad range of data transmission speeds, the possibility of transmitting communications in a wide range of lengths, and flexible protocol that makes it possible to work in real time or relative time scales among different types of users.

Integrated communications networks are designed and built on the basis of already-existing networks. Combinations of switching procedures in integrated communications networks are employed with due regard for maximum integration of functions and joint use of the resources of switching procedures.

The problem of constructing a general-use integrated communications network covers a number of questions: the architecture of the network, linking the network and the user, network control, consolidating connecting lines, and constructing switching centers.

The main question in constructing an integrated communications network, which in large part determines the solution to the other questions, is defining the degree of integration of the communications networks. Figure 1 below shows different degrees of integration for channel and packet switching networks.

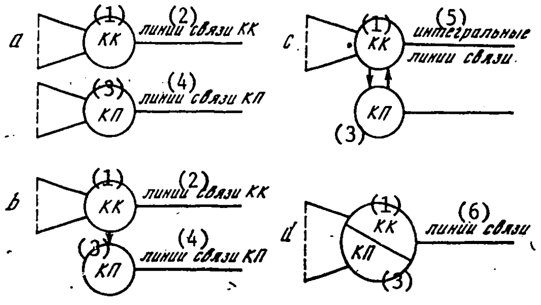


Figure 1.

Key: (1) Switching Channel (KK);
(2) Communications Lines of Switching Channel;
(3) Packet Switching (KП);
(4) Packet Switching Communications Lines;
(5) Integrated Communications Lines;
(6) Communications Lines.

Figure 1a shows the divided structure of communications networks that exists at the present time. Each network here has its own set of users and communications lines. The communications networks do not have common equipment.

The first degree of integration (Figure 1b) offers users the possibility of employing all the resources of the communications networks that make up the integrated system. In this case the switching equipment at the switching center and the communications lines remain separated for all switching methods. Users are connected to the communications network through one of the switching systems. In this case it is the channel switching system.

The second degree of integration (Figure 1c) has maximal merging of the functions and resources of the switching systems being combined. It is based on one system and minimizes the individual equipment of the other. For example, if the integrated communications network is constructed of channel switching and packet switching systems based on the channel switching system, the individual equipment of the packet switching system is minimized. With this degree of integration the connecting lines are shared, while the switching and control systems remain separate.

Finally, the third degree of integration offers the possibility of access to the equipment of the communications network for users in all categories (see Figure 1d) with shared use of communications lines and merging the switching and control functions for all the switching systems being consolidated. This permits shared use of the entire memory volume of the memory unit, the power of center processors, and the software system of the switching center.

Development of the switching center is a key question in building an integrated communications network. When selecting the structure of the switching center it is necessary to consider the similarity and differences of the processes of establishing connections for different switching procedures. The similarity consists in the existence of three phases of processing a demand to establish a connection and transmit information: the phase of establishing the connection; the information transmission phase; the signoff phase. The procedure for initiating the communications channel is also the same for all methods of switching. The difference is that with some switching methods (channel switching, and sometimes also packet switching) a real communications channel is established, while in other cases (packet switching and communications channels) the channel is virtual. This leads to a difference in the organization of hardware and software. The software includes additional procedures to organize the distribution of memory to store packets and messages, as well as the programs needed to process them. The hardware includes the additional memory volume of the center memory units and the additional power output of the center processor. The additional power expenditure of the center processor is reduced by constructing and separating the modules for processing and storing packets and messages from the center processor. This greatly increases the flexibility of the hybrid switching system.

Furthermore, the additional power expenditure of the center processor is reduced by constructing a multiprocessor structure in the switching center [4].

The next important question of constructing the integrated communications system, which determines the choice of the switching center structure, is selecting the method of hybridization. Several techniques are known: separate receipt and processing of information transmitted by different switching techniques; joining the information of the data transmission network and telephone network users into "envelopes" which are transmitted on the main line, and transmitting information with communications channels and switching channels with transduction in the hybrid switching center (but for communications channels complete messages are transduced, whereas with switching channels it is particularly meaningful segments).

At the present time hybrid switching systems for integrated communications networks are being developed and introduced intensively. Some of the most recently developed systems are the EDS and EDX systems which are being introduced in the national network for transmission of discrete information of West Germany [4, 11, 12]; the APB-20 and T-200 in Switzerland [8, 16]; the Unified Switching Center and DS-714 in England [7, 14]; the DDX-2 in Japan [13]; and, the M-3200 in the United States [5, 15]. Let us consider a few of them.

Conclusion

Our analysis of the experience of many large companies in the areas of designing, developing, and introducing hybrid switching networks enables us to formulate the following basic requirements, which must be the basis for construction of such networks:

1. Optimization of the processes of controlling blocks of transmitted information;

2. The existence of general network protocol that defines the principles of control of the network and interaction between network centers and users;

3. Selection of optimal switching procedures depending on the characteristics of the information, users, and condition of the network;

4. Insuring a high degree of reliability in switching centers;

5. Insuring the possibility of expanding the capacity of switching centers;

6. Ability to operate in both real and relative time scales;

7. Developing software and a system of commands that maximally simplify programming and the entire cycle of software work in the process of information exchange.

11176
CSO: 1863/137

UDC 681.32:681.3.06.001.57

A METHODOLOGY FOR THE SIMULATION MODELING OF THE COLLECTIVE-USE COMPUTER CENTER OF THE SIBERIAN DEPARTMENT OF THE USSR ACADEMY OF SCIENCES

Riga AVTOMATIKA I VYCHISLITEL'NAYA TEKHNIKA in Russian No 1, Jan-Feb 81 pp 3-14 manuscript received 25 Dec 79

[Paper by Yu.I. Mitrofanov]

[Text] A number of systems analysis problems came up during the design of the collective-use computer center of the Siberian Department of the USSR Academy of Sciences (VTsKP) [CUCC] being created in the Novosibirsk Scientific Center, the solution of which proved to be possible only by means of simulation modeling. In terms of algorithm structure and functioning, the CUCC is a typical network of computer complexes [1] and belongs to the class of large complex systems characterized by stochastic functioning.

After studying the possible language tools for the simulation modeling of the CUCC, ALGOL-60 was chosen with the procedural expansion up to the level of a timewise modeling language for discrete systems (the set of simulation modeling procedures for discrete systems - the KIMDS) [2, 3]. The reasons for such a selection are treated in section 1. The methodology for the simulation modeling of discrete systems using the tools of the complex, its composition and structure which was realized in the set of simulation modeling procedures for discrete systems are treated in section 2. The organizational principles based on the set of simulation modeling procedures for discrete systems for the CUCC simulation models are treated in section 3. The composition and structure of two simulation models of the CUCC are discussed in section 4 as well as some results of studying it.

1. *Language tools for simulation modeling.* Simulation modeling as a technique for studying real and conceptual systems provides the possibility of constructing models which a theoretical description of the systems being modeled with an level of detailing. It is specifically this fundamental property that in many cases determines simulation modeling as the only possible study technique (especially for complex systems), despite its well known drawbacks: the labor intensity of the development of simulation models, the complexity of debugging

them and the large amount of computer time required to perform experiments with models on a computer. The development of the means of automating simulation modeling and computer hardware is constantly reducing the relative weight of the deficiencies of this method as compared to its advantages, thereby providing for a constant expansion of the practical utilization of simulation modeling.

The labor intensity of the development of simulation models and their debugging depends substantially on the programming languages used for the models, in particular, on the existence of convenient and economical tools in these languages for representing the systems being modeled in models of the structural and functional properties. When developing the means of automation for simulation modeling, as a rule, the fact that the potential users of these tools are usually specialists in the systems being studied is taken into account, where these specialists do not have professional training in the general case either in programming or in the methodology of simulation modeling. For this reason, the modeling tools should basically allow for the reduction of the task of developing a simulation model to the convenient and natural depiction of objects and relationships in the system being modeled by the appropriate objects and relationships in the modeling system being employed. In this case, the methodological principles for the construction of models are governed by the systems being used for the automation of the modeling. For example, discrete system modeling languages are being used at the present time which are oriented towards the depiction of events, processes, operations, etc. [4]. Universal algorithmic programming languages (AYaP) by virtue of their primary function neither contain the specific tools necessary for the representation of time relationships in the models, where such time relationships occur in the systems being modeled, nor embody a methodology for the construction of simulation models. At the same time, the broad capabilities of these languages as regards the representation of functioning algorithms for the elements of the systems being modeled have made it possible to use them as basic languages in the development of a number of specialized modeling languages (SYaM) (for example, SIMULA and SIMPL/1).

At the present time, both universal algorithmic programming languages and specialized programming languages are used in simulation modeling practice for discrete systems (the advantages and drawbacks of their use for these purposes were treated in [4]). The reasons for the use of an algorithmic programming language in simulation modeling are actually deeper than those usually cited (the lack of translators from the modeling languages to those accessible to a computer user, difficulties in operating modeling systems, the lack of high quality documentation for users of modeling systems). In comparing algorithmic programming languages and specialized modeling languages from the viewpoint of their usage in simulation modeling, the following must be taken into account: 1) The ideas and content of the basic concepts of simulation modeling are rather simple, and understanding them does not as a rule cause any difficulties; 2) The existing modeling systems are oriented to one degree or another towards definite classes of modeled ssytems, something which generates differences of a methodological nature in the depiction of the modeled systems in the models; for this reason, in particular, recommendations for users regarding the choice of specific modeling systems prove to be useful [8]; 3) The striving of designers of modeling languages to make them universal, on one hand makes it difficult to study these languages and their practical applications by users, who are not programming specialists, and on the other hand, reduces the efficiency of the simulation models as programs; 4) In the development of simulation models for complex systems, along with the complexity of the functioning algorithms for the system components, the weight of the tools made available by the modeling language for the description of the algorithms increases with respect to the tools for the systemic organization of the models.

Thus, the utilization of universal algorithmic programming languages (as a rule, with the creation of the requisite tools within the framework of these languages) in the case of simulation modeling of complicated discrete systems is natural at the present stage of development of specialized modeling languages, which is on the whole still not satisfactory.

2. *The KIMDS Simulation Modeling Complex.* In structural terms, the set of simulation modeling procedures for discrete systems is a complex of procedures for various purposes [3]. An independent component of the complex is the master program, which organizes the operation of the simulation models as a whole. All of the procedures of the complex can be broken down into several sets: synchronization and control procedures; statistical data retrieval and processing procedures during modeling; random quantity generation procedures; procedures for working with data files; and service procedures.

More than 60 procedures for different functions are incorporated in the complex at the present time. The structure of the complex makes it possible, on one hand, to expand the composition of the complex through the inclusion of the requisite procedures, and on the other hand, to utilize only the requisite procedures of the complex in the development of specific simulation models.

The States of Processes and the Control of Their Development. We shall consider the basic principles for the organization of simulation models of discrete systems using the example of a simulation model I for a system S. We call a structural unit of the simulation model a program process. Corresponding to each element E_i, $i = 1, 2, \ldots, n$, of the system in the simulation model is a program process P_i, which is a description of the given element in the ALGOL-60 programming language and formatted as a procedure. Consequently, n program processes P_i, which are descriptions of the elements E_i, are incorporated in the model I.

Messages are represented in simulation models by requirements of various classes, while the message sections are represented by requirement sections. The requirements in the models are represented by their numbers, and for this reason, the motion of requirements in the models is essentially the motion of the numbers representing them among the various units of the models. Setting the requirements in a section and selecting them from the sections in the models is accomplished by accessing the *put* and *get* procedures. We shall assume that all of the processes in the model are numbered, and the number of the process P_i is designated as ν_i.

We shall draw a distinction between the main and auxiliary possible states of the program process P_i and designate them as $\rho^{(f)}$, $s = 1, 2, \ldots, 10$. The main states are the active and the passive states ($\rho^{(1)}$ and $\rho^{(2)}$ respectively). The remaining states are auxiliary ones. When running a simulation model on a computer, each program process realizes a certain sequence of discrete events on the model time axis, which represents the corresponding elementary process. The realization of the events is accomplished by program processes only at the point in time where are present in the active state. If some condition must be met to realize the next event, then the process is in the passive state until this condition is met. We shall term the state $\rho^{(3)}$ the stop state. It is a special case of the passive state, when the condition for the activation of the process is the direct action of another process which is in the active state. The introduction of the state $\rho^{(j)}$, $j = 4, 5, \ldots, 10$, was needed for auxiliary purposes related the operational algorithms of the modeling system as a whole, and in particular, to make an analysis to see that the events being realized in the simulation models are noncontradictory.

Synchronization and control procedures, as well as operators for accessing them are used in the bodies of the processes to assure the representation of the parallel development of elementary processes in a model and provide for the synchronization of the development of the program processes as well as control their development.

We shall define the content of the synchronization and control procedures, presupposing for the sake of definiteness that the operators for accessing them are used in the body of the process P_i: delay (ρ, ψ, ω) delays the development of the process P_i (shifts to the passive state) for ϕ units of model time or until the condition ρ is met; hold (ϕ, ω) delays the development of the process P_i for ϕ units of model time; wait (ρ, ω) delays the development of the process P_i until the condition ρ is met; halt (ν ω) stops the process at the number ν (shifts to the stop state); leav(ω) stops the process P_i; start(ν, ω) activates the process at the number ν (shifts to the active state), which is in the stop state. The realization of the algorithm of the process with the number ν will be started with the first segment of the process algorithm; push(ν, ω) activates the process with the number ν, which is in the stop state (the realization of the algorithm for the process with the number ν will be started with the segment of the algorithm, ahead of which it was stopped); finish shuts down the modeling process, i.e., terminates the execution of the simulation model on the computer.

Service Tables and the Master Program. The operation of the set of simulation modeling procedures for discrete systems and the simulation model as a whole is controlled by the master program, which, just as the procedures for synchronization and control, uses information on the processes during operation which is contained in tables of process states, time markers, return points, prototypes and copies. In these tables, the numbers of the elements which contain the information pertaining the processes correspond to the numbers ν_i of the processes P_i. The codes of the process states $\rho^{(s)}$, $s = 1, 2, \ldots, 10$, are the values of the elements of the table of process states.

We shall use θ to designate the current value of the model time, and τ for the point in the model time at which the next event should be realized in the process P (we shall call this moment the time marker of the process P). We shall use the character $[\infty]$ to designate a positive real number which is the representation of $+\infty$ in the digital computer, on which the modeling is performed (an experiment with a simulation model). For processes which are in an active state, $\tau = \theta$; in a passive state $\tau > \theta$ (in particular, we can have $\tau = [\infty]$, and in the stop state, $\tau = [\infty]$. The process P can shift from the active state to the passive one only as as a result of the execution of the operator of one of the synchronization procedures: delay, hold or wait. With the execution of the first two operators, the point in time τ for the process P is computed from the formula $\tau = \theta + \phi$; with the execution of the operator for the wait procedure, τ takes on the value of $[\infty]$. The transition of a process P with a number ν to the stop state can occur either from the active state in accordance with the operator for the leav procedure, or from the active and passive states with the operator for the halt procedure, carried out by another process. With the transition of the process P to the stop state, τ assumes the value of $[\infty]$. The values of the elements of the time marker table are the points in time τ for all of the simulation model processes; these values fall in a range of $[\theta, [\infty]]$. It stands to reason that the values of τ for each process change in the course of running the routines for the simulation model on the computer.

During the execution of the operators of the synchronization and control procedures, used in the body of the process P, the control is transferred to the master program. To provide for the return of the control to the process P, the master program should have data on the "point of return to the process" or the number of the algorithm segment which should be executed as the next one and which directly follows the operator. The point of return to the process is governed by the parameter ω in the synchronization and control operators. The value of this parameter is entered by the synchronization and control procedures in the table of return points, i.e., the value ω is assigned to the table element corresponding to the process P.

We shall call two process P_i and P_j, i. $j = 1, 2, \ldots, n$, $i \neq j$, equivalent if they differ only in the values of the internal parameters. We shall designate the number of equivalent processes of the k-th type as e_k, $k = 1, 2, \ldots, m$, where m is the number of types of equivalent processes, while we designate the set of k type equivalent processes as $F_k = \{P_{s,k}\}$, where s $\{1, 2, \ldots, n\}$.

It is sometimes expedient to have only a single program for process representation for equivalent processes of a set F_k in a simulation model, where we shall call this representation the prototype of the k-type processes. We shall call the k-type processes themselves the copies of the prototype in this case. With such an approach to the organization of simulation models, it is sufficient to have m prototypes in a model, something which in many cases makes it possible to substantially curtail the volume of a simulation model and a program.

Tables of prototypes and copies are used to refer to a specific process. The numbers of the prototypes or the types of processes are the values of the table of prototypes. The numbers of the copies of the prototypes or the internal numbers of the processes which form the set F_k are the values of the elements of the table of copies. For example, if the process P_i with a number ν_i belongs to F_k, then the

ν_i-th element of the table of prototypes is equal to k, while the ν_i-th element of the table of copies is equal to j $\{1, 2, \ldots, e_k\}$

The sequence of actions performed by the modeling complex is governed by the organization and functional content of the master program and is conditionally broken down into six steps. In each step, the master program executes the corresponding actions, the major ones of which are: the transfer of control to the active process, the processing of the auxiliary markers, the control of the realization of the coupling of the processes, checking that the conditions for process activation are met, calculating the time markers, checking for the presence of active processes, searching for the process with the minimal time marker τ_{min}, searching for processes for which the time markers are $\tau = \tau_{min}$ and correcting the model time.

Statistical Data Retrieval and Processing During Modeling. For the sake of brevity, we shall call the set of procedures for statistical data retrieval and processing during simulation modeling of discrete systems, together with the information base, the system S. This system provides for the retrieval and processing of data defined by the user prior to starting the simulation model (prior to the performance of an experiment with the model on a computer). In this case, the user has the capability of controlling the data retrieval and processing process by means of specifying the appropriate control data: for example, specifying the point in model time, beginning at which the data retrieval and processing will be carried out. This capability has been realized for the purpose of providing for data retrieval only after the completion of the transient operational mode of the model, i.e., when the model enters the steady-state mode. Based on the values of the confidence level and confidence interval specified by the user, the system S can terminate (at the request of the user) the retrieval of certain data, if an estimate with the requisite precision is obtained. In the case of a limited volume of immediate access memory of the computer, in which the experiment is performed, the user can organize the retrieval of the requisite data in a sequential mode. In this case, the retrieval and processing of certain data will be accomplished after the completion, processing and output of other data. The system S, in accordance with the information specified by the user, can calculate estimates of the mean values and dispersions of the random lengths of the indicated sections and the random lengths of the model time intervals, the boundaries of which are determined by the user in the specified initial data. The indicated system can also generate histograms of the random interval length and random section length distributions, as well as provide for the capability of routing a requirement with an arbitrary indicated number. In the routing mode, during the modeling process the numbers of the sections and points in model time of the setting of the indicated requirement in a section and its selection from the section are printed out. This makes it possible to trace the motion of a requirement in the model during the modeling process, something which substantially facilitates the debugging of the models.

The composition structure and some of the algorithms of the system S have been treated in the literature [6, 7].

3. *The organizational principles of simulation models for CUCC's.* The Model of a CUCC. The computer resources of a CUCC are the computer complexes (VK),

tied together by a data transmission network (SPD), formed by data transmission centers and channels. Computer complexes are broken down into mainframe (BVK) and peripheral (PVK) complexes. The network centers are designed around communication processors, and each computer complex is directly coupled to one of the centers. We shall assume that a center, the computer complex directly coupled to it and the computer complex terminals tied to this form a link. We call all of the elements of a certain link and the users employing the terminals of this link local with respect to each other, while the elements and users of another link are termed remote with respect to the given link.

A user can interact with the network in two ways: one-time and an interactive mode, according to the operational mode of the specific computer complex)batch processing or time sharing). In this case, a user feeds out a message from a terminal, where the addressee is indicated in the message. The addressees can be either local or remote computer complexes and users. The message sources can be only the users. Three kinds of messages come in to each computer complex: internal, external and remote. Internal and external messages come from local users and are distinguished by the fact that the former are completely processed by the given computer complexes, while the latter are transmitted for processing by remote computer complexes. We shall call external messages addressed to BVK's [mainframe computer complexes] the base messages, while those addressed to users are callled communications messages. Remote messages come in from remote users. A computer complex upon receiving a message addressed to it or to its local user, generates a notification which is routed to the address of the source of the given message. The results of the message processing in the network by one of the computer complexes (local or remote) are routed to the message source in the form of a notice.

When switching the packets in a data transmission network, all of the messages and notices incoming to it are formatted as packets of a fixed size. Only the immediate access memory of the communications processors is used as the buffer memory at the junction centers.

The model of the environment of the CUCC. The environment of a CUCC is created by its users, for whom the category, type and form of the user are defined as the characteristics, where these determine the basic user characteristic: the ranking of the user.

Two categories of users are introduced in the models corresponding to a definite external priority level of the administrative network, a priority which is assigned to messages of the given users for transmission and processing in the CUCC. There can be two types of CUCC users: one-time and interactive users, corresponding to their modes of interaction with the network, as well as two kinds, determined by the kind of dominant network resource used during the processing of the user messages. Network resources are broken down into computational and informational (communications resources are also included in the latter). We then associate users with the first and second kinds, during the processing of the messages of which, primarily computational and informational network resources resepctively are employed. It follows from the classification given here that network users can belong to one of eight ranks.

Requirements and their attributes. Four classes of requirements are represented in the CUCC models: messages, notices, packets and notifications. They all have a main attribute, the marker, and a supplemental attribute, the complement. The requirements in the models are represented by their numbers, which are generated and erased by the model processes. The number is a unique characteristic, which identifies the requirement in the model. The class of a requirement can change in the process of processing and moving it in the model, but the number of a requirement always remains constant. The movement of requirements in the model is depicted by the movement of their numbers.

The attributes contain the basic characteristics of a requirement which are represented by indicators. The marker has a composite structure and consists of five indicators, which define the code of the requirement class, the number of the priority level (during requirement processing and transmission), the number of the user rank to which the given requirement "belongs", the number of the message based on which the requirement was "generated" and the address of the requirement source. The complement consists of five indicators, which define the address of the requirement addressee, the code for the addressee criterion (this indicator defines and is the addressee of the requirement of the computer complex or user), the number of packets (in the message or notice), the packet number and the packet criterion code (this indicator determines the association of a packet with a message or notice).

The markers and complements of requirements are contained in tables of markers and complements. The marker and complement codes are the values of the elements of the corresponding tables; the numbers of the elements are equal to the numbers of the requirements. The marker and complement codes are used by and can be corrected by the model processes during requirement processing and transmission in the network model.

The structure of the models. The simulation models of CUCC's include four types of program processes which depict definite CUCC components: π_s depicts the set of users (the environment) of a computer complex, including the users of various categories, types and kinds (the process generates a message flow which depicts the overall message incoming to the computer complex from the users of the given set); π_c depicts the computer complex of a CUCC (the process simulates the execution of the main functions of the computer complex of analyzing, processing and transmitting the messages, notices and notifications incoming to the computer complex); π_n depicts a junction center (communications processor) of the data transmission network (the process simulates the execution of the main functions of a data transmission network junction center of analyzing, processing, preparing for transmission and transmitting the messages, notices, packets and notifications incoming to the junction); π_l depicts a simplex data transmission channel of the data transmission network (the process simulates the transmission of messages, notices, notifications and packets via the channel).

The trajectories of movement of the various classes of requirements between the model processes during their functioning depict the trajectories of requirement movement among the components of the CUCC which are depicted by the corresponding processes.

The queues which are realized in the models based on the principle of processing the requirements located in them are broken down into dynamic and static types (queue listings). Dynamic queues are the input queues to the processes, while static queues are the internal queues of the processes. The requirements from a dynamic queue are processed as processes without interruptions, i.e., until all of the requirement found within the queue are processed. The processing of requirements from a static queue can be interrupted, specifically: upon completing the processing of the next regular requirement, a process can terminate the processing of the requirements from a given queue and changeover to the performance of other functions, in particular, begin the processing of requirements from other queues. Setting and selecting the requirementsfrom the queues is accomplished in accordance with the "first come - first served" principle, with relative priority, i.e., without interrupting the processing of a lower priority requirement upon the arrival of a higher priority requirement. The queues in the model are realized in the form of coupled listings.

4. *Simulation modeling of a CUCC.* Some results of a study using simulation modeling of one of the organizational variants of a CUCC, which we shall call the VTsKP-G [CUCC-G = Collective-Use Computer Center - G] are treated in this section [8, 9].

The CUCC-G includes 13 computer complexes: 10 peripheral and 3 mainframe complexes. All of the peripheral computer complexes are based on the M-7000 computer, while the base mainframe computer complexes, the BVKYe, BVKB and BVKE, are based on the Unified System of Electronic Computers, the BESM-6 computer and the "Elbrus" multiprocessor computer. All of the computer complexes are tied together by the data transmission network. The functions of the data transmission network junctions in the CUCC-G are partially realized by the M-7000 computers incorporated in the computer complex, and partially by microcomputers. The topology of the data transmission network is shown in Figure 1, where Z_i, $i = 1, 2, \ldots, 13$, designates the links of the CUCC-G, while the lines correspond to duplex data transmission channels, each of which is designed in the form of two simplex data transmission channels (KPD), one of which is the input channel relative to a specific junction, while the other is the output.

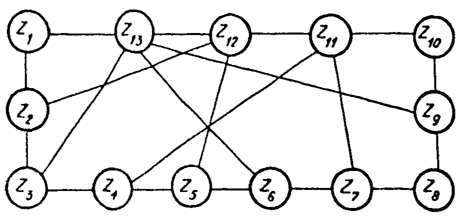


Figure 1. The topology of the data transmission network of the CUCC-G

The carrying capacities of all data transmission channels are [illegible]6 Kbit/sec. The links Z_i, 1, 2, ..., 10, correspond to the peripheral computer complexes, while Z_{11}, Z_{12} and Z_{13} are the BVKYe, BVKB and BVKE respectively (there are no users having direct access to the BVK [base mainframe computer complex; computers in the links corresponding to the BVK's). The major algorithms and some of the parameters of the CUCC-G and its environment were treated in [10].

The CUCC-G was studied through experiments with simulation models: the IM-2 and IM-3. The IM-2 model depicts the CUCC-G as a whole, while the IM-3 model depicts the CUCC-G data transmission network. Various modifications of the processes π_s, π_c, π_n and π_l are used in these models. The processes in the models are combined into blocks. The block B_i, $i = 1, 2, \ldots, 13$, depicts the link Z_i of the CUCC-G

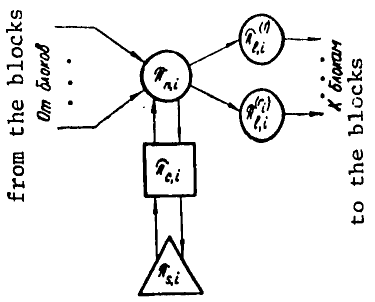


Figure 2. The configuration of the block B_i of the IM-2 model

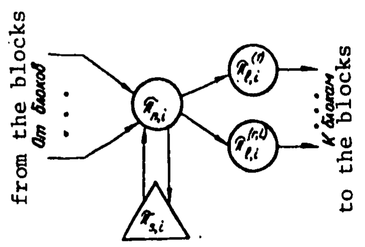


Figure 3. The configuration of the block B_i of the IM-3 model.

with the output data transmission channels. The blocks B_i in the IM-2 and IM-3 models differ in terms of the composition and content of the processes included in them. One each of the type π_s, π_c and π_n processes is incorporated in the B_i of the IM-2, as well as r_i type π_l processes, where $r_i \geq 1$. Incorporated in the B_i of the IM-3 are the same types of processes, with the exception of π_c, the basic characteristics of which are depicted in modifications of the π_s and π_n processes. The structural configurations of the B_i blocks in the IM-2 and IM-3 models are shown in Figures 2 and 3.

One of the systems analysis problems solved by means of simulation modeling was the study of the impact of various methods of organizing data transmission via the data transmission link on the quality of CUCC-G functioning: message switching and packet switching. The results of the relevant experiments with the models are treated below. An algorithm was realized in the models for routing the messages and packets, which provided for their transmission between the links of the CUCC-G (between the centers) via minimal routes (with the least number of intermediate junctions). If several minimal routes exist for the transmission of messages and packets from Z_i to Z_j, $j \neq i$, $i,j = 1, 2, \ldots, 13$, then they are transmitted with equal probability via one of them. The addressing of external messages to various CUCC-G computer complexes was specified in the modeling by the distribution of the probabilities $d = (d_0, d_1, d_2, d_3)$, where d_0, d_1, d_2 and d_3 are probabilities of the addressing of external messages to the following respectively: the peripheral computer complexes (users), the BVKYe, BVKB and BVKE. A distribution of d = (0.03, 0.93, 0.02 and 0.02) was specified in the experiments with the models and the existence of two categories of messages was depicted in the CUCC-G (two categories of users) corresponding to the level of their external priority (20 percent of the messages were of the first category, and 80 percent were of the second).

It was presupposed in the switching of the packets (the packet size was fixed) that in the case of incomplete filling of a packet with data, it was filled completely with "zeros", the transmission of which produces an additional load on the data transmission channel. This way of depicting the transmission of incomplete packets was used to partially take into account the additional loading of the data transmission channel produced by the repeated transmission of packets in which errors were detected which were caused by interference in the channels, since the actual mechanism for detecting errors in packets and providing for repeated transmission of packets received with errors which is present in the CUCC-G was not reflected in the models for the sake of simplicity.

Considering the fact that $d_1 >> d_2, d_3$, the volumes of the base messages in the BVKB and BVKE in the experiment with the IM-2, the sizes of the base messages addressed to these BVK's and the sizes of the notices generated by these complexes were all specified as equal to the first moments of the corresponding distribution functions. For the purpose of reducing the time required to lock the model into the steady-state mode when generating the initial state of the model (during each experiment), the initial setting of the messages was accomplished in the queue of messages awaiting processing in the BVK, where the number of messages was equal to the average number of messages in these queues, determined by means of analytical modeling [11].

The fact that the average length of time the messages were present in the base computer complex of the CUCC-G amounts to about 52 seconds was taken into account in the experiments with the IM-3 when specifying the initial data for the modeling. This value was obtained by means of analytical modeling of the CUCC-G and was confirmed in simulation modeling using the IM-2.

The average response times of the CUCC-G for external messages and the load factors of the devices were taken as the basic parameters of the CUCC-G, which characterize its operational quality. In particular, the average response times for base messages, especially messages addressed to BVKYe's, were of the greatest interest, since the utilization of the problem oriented program systems of an information and reference nature which were realized in the BVKYe and which are under development at the present time, was depicted in the modeling. A study of the response times for base messages was made by means of the IM-2. Also taking into account the fact that the switching method basically influences the time characteristics of data transmission through a data transmission channel, it was important to determine the extent of this influence. For this reason, the average transmission through a data transmission system, which is the sum of the average duration of time that a message and the notice corresponding to it are present in the data transmission system was studied as a function of the switching method and packet size (in the case of packet switching) in the experiments with the IM-3.

We shall introduce symbols for the estimates of the average response times of the CUCC-G and the load coefficients of the data transmission channel, in which S = Ye, B and E is the indicator of the BVK, k = 1, 2 is the indicator of the category (the external priority level) of the messages: $\rho_c^{(b)}$ is the estimate of the average response time of the CUCC-G for base messages, addressed to the BVKS; $\rho_{k,s}^{(b)}$ is the estimate of the average response time of the CUCC-G for k-th category base messages, addressed to the BVKS; ρ is the estimate of the average transmission time through the data transmission system; ρ_k is the estimate of the average transmission time through the data transmission system for k-th category messages and notices; $\psi(i,j)$ is the estimate of the average load coefficient of a data transmission channel, which is the output channel for the i-th junction center and the input one for the j-th junction.

The results of two series of experiments are considered below, the first of which (the experiment $Э_1^{(2)}$, $Э_2^{(2)}$, $Э_3^{(2)}$ was performed using the IM-2 model (see Table 1), while the second (experiment $Э_1^{(3)}$, $Э_2^{(3)}$, $Э_3^{(3)}$) was performed with the IM-3 model (see Table 2). In eperiments $Э_1^{(2)}$ and $Э_1^{(3)}$, a message switching method was

TABLE 1

Таблица 1

	BVKYe БВКЕ			BVKB БВКБ	БВКЭ BVKE
	$Э_1^{(2)}$	$Э_2^{(2)}$	$Э_3^{(2)}$	$Э_1^{(2)}, Э_2^{(2)}, Э_3^{(2)}$	$Э_1^{(2)}, Э_2^{(2)}, Э_3^{(2)}$
$\rho_c^{(b)}$ (c)	35	41	23	900	400
$\rho_{1,c}^{(b)}$ (c)	9,8	8,4	6,1	70	60
$\rho_{2,c}^{(b)}$ (c)	42	51	29	1100	530

TABLE 2

	$Э_1^{(3)}$			$Э_2^{(3)}$			$Э_3^{(3)}$		
	z_1	z_7	z_8	z_1	z_7	z_8	z_1	z_7	z_8
ρ (c)	8,81	3,65	6,23	6,91	6,56	6,87	5,24	5,27	4,87
ρ_1 (c)	6,67	—	—	6,16	3,22	3,73	3,33	2,77	2,83
ρ_2 (c)	9,34	—	—	7,06	7,22	7,89	5,8	5,79	5,33
$\psi_{(7,11)}$	0,128			0,225			0,162		
$\psi_{(11,7)}$	0,624			0,775			0,64		

realized in the models, while in the remaining experiments, a packet switching method was used. In this case, the size of the packet S_p was specified as 512 bytes in $Э_2^{(2)}$ and $Э_2^{(3)}$, while it was specified as 256 bytes in $Э_3^{(2)}$ and $Э_3^{(3)}$.

For the base messages addressed to BVKB's and BVKE's, the difference in the values of the estimates obtained in the experiments falls within the range of precision in their determination, and therefore the values of the estimates given in Table 1 are the average values for all three experiments. As can be seen from Table 1, the estimate $\rho_E^{(b)}$ and $\rho_{k,E}^{(b)}$ depend both on the switching method and on the packet size. The nature of and the reason for this dependence become understandable if one considers the fact that the average length (number of data transmission channels) of a minimal route from a peripheral computer complex to the BVKYe is 1.8, while the average sizes of the base messages for BVKYe's and notices for these messages are equal to 400 and 2,000 bytes respectively. For this reason, when switching packets with S_p = 512 bytes, the impact of reducing the average transmission time through the data transmission system due to the simultaneous transmission of the packets of one method via the several series data transmission channels of its route (the primary effect) proved to be less than the impact of increasing the transmission time caused by the transmission of the "zeros" with incomplete filling of the packets (the second effect). Reducing the size of a packet (S_p = 256 bytes) stengthened the first effect and weakened the second, and provided for a substantial reduction in the response time of the BVKYe as compared to the case of message switching.

Estimates are given in Table 2 for the transmission time through the data transmission system for external messages generated in links Z_1, Z_7 and Z_8. As follows from the nature of the distribution d, the interchange of the peripheral computer complexes and the BVKYe produces the primary load on the data transmission system (the messages in the forward direction and the notices in the return direction). It can be seen from the topology of the CUCC-G (see Figure 1) that the length (number of data transmission channels) of the minimal path to the BVKYe (Z_{11}) for three peripheral computer complexes is equal to one, for six such complexes it is two, and for one peripheral computer complex it is three (these three types of peripheral computer complexes are represented in Table 2 by links Z7, Z_8 and Z_1 respectively).

It follows from an analysis of the results of the experiments with the IM-2 and IM-3 that packet switching provides better functional performance than message switching, and that the optimal packet size is 256 bytes. Reducing the packet size down to 128 bytes is not expedient, since it causes a substantial increase in the load on the processors at the data tranmission system junction centers with the execution of programs for packet processing and selecting their route.

Estimates of the overall average response time of the CUCC-G, $\rho_1^{(1)}$, $\rho_2^{(1)}$ and $\rho_3^{(1)}$ for switching messages and switching packets with S_p = 512 and S_p = 256 bytes respectively can be derived from Table 1 using the distribution d: $\rho_1^{(1)} = 58$, $\rho_2^{(1)} = 64$ and $\rho_3^{(1)} = 46$ seconds. At the same time, we obtain $\hat{\rho}_1^{(1)} =$ $= 58.2$, $\hat{\rho}_2^{(1)} = 58.9$ and $\hat{\rho}_3^{(1)} = 56.9$ seconds from Table 2, for example, for Z_8 (a characteristic case for a CUCC-G, if one considers the topology of the data transmission system) taking into account the assumptions used in the model that the average time a message is present in the base mainframe computer complex is 52 seconds. Thus, the dependence of the CUCC-G response time on the switching method and the packet size, determined by the IM-3, is significantly less than the dependence found by means of the IM-2. The major reason for this difference is the smaller amount of detail in the IM-3 as compared to the IM-2 as regards reflecting the impact of the switching method and packet size on the rates of the message flows from the users to the CUCC-G, and consequently, on the time messages are present in the BVK (especially in the BVKYe, if one takes into account the form of the distribution d).

BIBLIOGRAPHY

1. Marchuk G.I., Moskalev O.V., "Metodologiya sozdaniya territorial'nogo vychislitel'nogo tsentra kollektivnogo pol'zovaniya SO AN SSSR" ["The Design Methodology for the Collective-Use Territorial Computer Center of the Siberian Department of the USSR Academy of Sciences"], KIBERNETIKA, 1977, No 6, pp 73-77.

2. Mitrofanov Yu.I., "O komplekse protsedur imitatsionnogo modelirovaniya diskretnykh sistem" ["On the Set of Procedures for the Simulation Modeling of Discrete Systems"], in the book, "Voprosy Kibernetiki" ["Questions in Cybernetics"], 1978, No 42, pp 29-34 (Moscow, Scientific Council on the Comprehensive Problem of 'Cybernetics' of the USSR Academy of Sciences).

3. Mitrofanov, Yu.I., Ivanov, A.N., "KIMDS - kompleks protsedur imitatsionnogo modelirovaniya obobshchennykh diskretnykh sistem" ["The KIMDS: A Set of Procedures for the Simulation Modeling of Generalized Discrete Systems"], PROGRAMMIROVANIYE, 1978, No 5, pp 74-83.

4. Kivia F.J., "Yazyki modelirovaniya" ["Modeling Languages"], in the book, "Mashinnyye imitatsionnyye eksperimenty s modelyami ekonomicheskikh sistem" ["Computer Simulation Experiments with Models of Economic Systems"], Moscow, Mir Publishers, 1975, pp 397-489.

5. Shannon, R., "Imitatsionnoye modelirovaniye sistem - iskusstvo i nauka" ["Systems Simulation Modeling: The Art and Science"], Moscow, Mir Publishers, 1978, 420 pp.

6. Mitrofanov YuI., Ivanov A.N., Yaroslavtsev A.F., "Sistema sbora i obrabotki statisticheskikh dannykh dlya imitatsionnykh modeley diskretnykh sistem" ["A Statistical Data Retrieval and Processing System for Simulation Models of Discrete Systems"], in the book, "Imitatsionnoye modelirovaniye sistem (Sistemnoye modelirovaniye -4)" ["Systems Simulation Modeling (System Modeling 4)"], Novosibirsk, Computer Center of the Siberian Department of the USSR Academy of Sciences, 1976, pp 61-80.

7. Mitrofanov Yu.I., Ivanov A.N., Yaroslavtsev A.F., "Printsipy organizatsii monitora kompleksa KIMDS" ["The Organizational Principles of the Monitor for the Set of Procedures for the Simulation Modeling of Generalized Discrete Systems"], Preprint No 163, Novosibirsk, Computer Center of the Siberian Department of the USSR Academy of Sciences, 1979, 45 pp.

8. Kvashnin G.A., Mitrofanov Yu.I., Posdnyakovich L.V., "Issledovaniye kharakteristik VTsKP SO AN SSSR metodom imitatsionnogo modelirovaniya" ["An Investigations of the Characteristics of the Collective-Use Computer Center of the Siberian Department of the USSR Academy of Sciences by Simulation Modeling"], in the book, "Tezisy nauchnykh soobshcheniy Vsesoyuznoy konferentsii 'Vychislitel'nyye sistemy, seti i tsentry kollektivnogo pol'zovaniya'" ["Abstracts of the Scientific Papers of the All-Union Conference on 'Collective Use Computer Systems, Networks and Centers'"], Part 3, Novosibirsk, Computer Center of the Siberian Department of the USSR Academy of Sciences, 1978, pp 147-151.

9. Mitrofanov Yu.I., Kvashnin G.A., "Nekotoryye rezul'taty imitatsionnogo modelirovaniya vychislitel'nogo tsentra kollektivnogo pol'zovaniya SO AN SSSR" ["Some Results of Simulation Modeling of the Collective-Use Computer Center of the Siberian Department of the USSR Academy of Sciences"], in the book, "Tezisy dokladov Vsesoyuznoy konferentsii 'Vychislitel'nyye seti kommutatsii paketov'" ["Abstracts of Reports to the All-Union Conference on 'Packet Switching Computer Networks'"], Riga, Zinatne Publishers, 1979, pp 201-204.

10. Mitrofanov Yu.I., Kurbangulov V.Kh., "Issledovaniye VTsKP SO AN SSSR (modeli i rezul'taty)" ["A Study of the Collective-Use Computer Center of the Siberian Department of the USSR Academy of Sciences (Models and Results)"], Part 2, Preprint No. 144, Novosibirsk, Computer Center of the Siberian Department of the USSR Academy of Sciences, 1979, 43 pp.

11. Kurbangulov V.Kh., Mitrofanov Yu.I., Shul'ga N.P., "Analiz i optimizatsiya VTsKP SO AN SSSR" ["Analysis and Optimization of the Collective-Use Computer Center of the Siberian Department of the USSR Academy of Sciences"], in the book, "Tezisy nauchnykh soobshcheniye Vsesoyuznoy konferentsii 'Vychislitel'nyye sistemy, seti i tsentry kollektivnogo pol'zovaniya'", Part 3, Novosibirsk, Computer Center of the Siberian Department of the USSR Academy of Sciences, 1978, pp 190-194.

8225
CSO: 1863/126

TRAFFIC DISTRIBUTION ALGORITHM FOR AUTOMATED PLANNING SYSTEM

Moscow PRIBORY I SISTEMY UPRAVLENIYA in Russian No 1, Jan 81 p 11

[Article by A.Ye. Os'minin, candidate of technical sciences, and S. V. Aleksakhin, engineer: "Traffic Distribution Algorithm in the ASPR Data Transmission System"]

[Excerpt] The development of the second phase of the automated planning calculations system (ASPR) envisages the creation of an integrated storage system and data processing based on computer centers (VTs) of Gosplans, ministries and departments combined in a data transmission network (SPD). In the designing of this network, great significance was attached to optimum traffic distribution along possible data delivery routes in accordance with a selected criterion.

Henceforth, the term "data delivery route" will indicate the sequence of data transmission channels which transmit information from the commutation center (TsKS), linked with the VTs--the source of data, that is, the relay is made to the TsKS, linked with the VTs, the data recipient. The criterion for traffic distribution is, as a rule, the average data delivery (yield) time.

The works cite a number of methods for traffic distribution which are heuristic based, providing the capability to obtain the optimum result (minimal average data delivery time) only in certain specific cases.

This article proposes an algorithm providing for optimum traffic distribution by routing in a SPD with arbitrary structure.

COPYRIGHT: Izdatel'stvo "Mashinostroyeniye". "Pribory i sistemy upravleniya", 1981

8851
CSO: 1863/142

PUBLICATIONS

ABSTRACTS FROM THE JOURNAL 'AUTOMATION AND COMPUTER TECHNOLOGY'

Riga AVTOMATIKA I VYCHISLITEL'NAYA TEKHNIKA in Russian No 2, Mar-Apr 81 pp 97, 99

UDC 681.324

CONCEPT OF A MODERN COMPUTING NETWORK

[Abstract of article by E.A. Yakubaytis]

[Text] A model and the architecture of a modern distributed computing network are discussed. A study is made of the logical elements of the network (system), the hierarchy of protocols and problems with which the network is confronted. An analysis is made of problems relating to the design of the logical and physical structures of a computing network.

UDC 681.324

DATA TRANSMISSION SYSTEM (SPD) OF THE 'EL'BRUS-1' MULTIPROCESSOR COMPUTING COMPLEX (MVK)

[Abstract of article by V.S. Burtsev and V.I. Perekatov]

[Text] The concepts of continuous adaptation and primary processing are formulated, which determined the approach to the design of the "El'brus-1" MVK data transmission system. The implementation of these concepts required the creation of a special-purpose high-level language making it possible to describe the network of MVK user stations and processes of interaction with them. The data transmission processor added to the MVK's structure is functionally oriented toward the execution of the lanuage's programs. A number of the results obtained, in the opinion of the authors, are of fundamental interest in the teleprocessing field.

UDC 519.713

APPROXIMATE ALGORITHMS FOR PARALLEL DECOMPOSITION OF AUTOMATA

[Abstract of article by Yu.V. Pottosin and Ye.A. Shestakov]

[Text] Three algorithms are suggested for the parallel decomposition of a partial finite automaton into a two-component network, based on approximate algorithms for minimizing the number of states of the automaton. The results are presented of testing them on a computer for two algorithms programmed in the LYAPAS-M language.

UDC 681.3.06:51

CONVERSION OF A MICROPROGRAM INTO A CODED FLOWCHART OF AN ALGORITHM

[Abstract of article by S.S. Guseva]

[Text] A method is suggested for automating the coding of an algorithm for describing the functioning of an operating unit by means of a set of control and information signals. Here the original algorithm is written in the form of a microprogram in a structural-functional microprogramming language. Formalized rules are presented which formed the basis of the algorithm for the configuration of microinstructions from micro-operations of individual blocks of the microprogram. The characteristics of the computer program complex are given.

UDC 681.32:519.713

SYNTHESIS OF A MICROINSTRUCTION DECODER UTILIZING A PROGRAMMABLE LOGIC ARRAY (PLM)

[Abstract of article by G.F. Fritsnovich]

[Text] A formalized procedure is suggested for synthesizing the logical structure of a microinstruction decoder which is optimum for implementation by means of a programmable logic array (PLM). The problem is solved of designing a microinstruction code of minimum length which makes it possible to reduce to the maximum the number of intermediate PLM lines required for implementing the decoder. Painting the nodes of a finite unoriented graph is employed in solving the problem of minimizing the length of the microinstruction code.

UDC 681.324

ANALYSIS OF BUFFERING PROCESSES IN TELEPROCESSING SYSTEMS

[Abstract of article by M.D. Broytman and B.Ya. Ettinger]

[Text] The process of the entry, buffering and processing of incoming messages is described by means of Markov models of two-phase queueing systems with lock-out, which makes it possible to take into account the influence of the processing phase on the size of the buffer pool. The results of calculations for models are presented for static and dynamic buffering algorithms. A comparative evaluation of algorithms is made in terms of the capacity of the required buffer storage with a specific probability of its overflow. Data are presented which make it possible to select the optimum size of a block with block-by-block buffering. A comparison with the results of a calculation for a corresponding single-phase model has demonstrated the considerable influence of the second phase on the system's efficiency indicators.

UDC 621.391:519.1

ONE ALGORITHM FOR DETERMINING THE MINIMUM INTERCEPTING SETS OF EDGES OF THE GRAPH OF A COMMUNICATIONS NETWORK

[Abstract of article by P.K. Pavint'yev]

[Text] An algorithm is suggested for determining the minimum intersections of the graph of a communications network based on the theory of structural numbers and making it possible at the same time to find the total number of trees, the number of minimum intersections and the importance of the graph's edges.

UDC 621.374.33

INVERSE THRESHOLD CHARACTERISTIC OF A DISCRIMINATOR OF INSTANTANEOUS VALUES AND ITS UNIVERSAL ROLE IN DYNAMICS CALCULATIONS

[Abstract of article by R.Ya. Stasha]

[Text] An analysis is given of the dynamic properties of a discriminator of instantaneous values (DMZ) as a separate element of automation systems. Expressions are obtained for calculating the dynamic threshold of a DMZ. It is demonstrated that discriminators of instantaneous values of the first order are totally characterized by the derivative of the inverse threshold characteristic. Knowledge of this characteristic is sufficient in order to determine unambiguously the response of a DMZ for any input signal.

UDC 621.317.755.018.756:621.372.542.29

DISCRETE STROBOSCOPIC CONVERTER WITH A CONTROLLED TRANSFER CHARACTERISTIC

[Abstract of article by I.M. Blyumenau]

[Text] A stroboscopic converter with a comparator of instantaneous values based on a symmetric transistor flip-flop is discussed. It is demonstrated that its transfer characteristic corresponds to a system of the first order, whereby the position of the positive terminal is determined by the magnitude of the gating current. The results of experimental studies are given and the areas of application of the converter are indicated.

UDC 681.327:681.324

STOCHASTIC ANALYTICAL MODELS OF COMPUTING SYSTEM INPUT/OUTPUT CHANNELS

[Abstract of article by A.A. Dimitrov, B.M. Kagan and A.Ya. Kreynin]

[Text] A procedure for constructing stochastic models of computing system (VS) input/output channels is discussed. A study is made of models of selector, block multiplex and byte multiplex channels. The models are constructed on the basis of multiphase, multichannel priority queueing systems with warmup. The procedure for calculating models of channels is illustrated by examples of channels of YeS computers and others.

UDC 681.324

RESEARCH PROCESS AS A SUBJECT FOR AUTOMATION

[Abstract of article by A.N. Vystavkin]

[Text] The stages of scientific research (review of published data, theoretical study of the model, preparation and performance of experiment, processing and interpretation of experimental results, etc.) are presented in the form of a generalized block diagram with decision making blocks and feedback. On the basis of an analysis of the block diagram general concepts relating to the creation of systems for automating scientific research are detailed: In particular, the need for overall automation (i.e., of all stages) of scientific research is specified, a demonstration is given of the evolutionary nature of automation systems from simple single-processor to multicomputer hierarchical systems and then networks uniting science centers, and several conclusions are drawn regarding the required structure of computing systems and networks for the automation of scientific research. Criteria are suggested for the effectiveness of automated systems for scientific research. The block diagram, advanced concepts and criteria suggested can be applied in modeling systems for the automation of scientific research.

UDC 519.873

METHOD OF TESTING DIGITAL EQUIPMENT

[Abstract of article by N.D. Ryabukha]

[Text] A method is discussed for testing a digital unit represented by a finite automaton by means of two testing automata, making it possible to reduce the testing equipment as compared with the familiar method. A second testing automaton depending on the tested automaton is added in place of the rollout unit or output decoder circuit. A procedure for synthesizing testing automata is presented, based on a specific example.

8831
CSO: 1863/161

TABLE OF CONTENTS FROM THE JOURNAL 'CYBERNETICS'

Kiev KIBERNETIKA in Russian No 1, Jan-Feb 81 (signed to press 13 Feb 81) p 153

[Text] CONTENTS Page

8545
CSO: 1863/149

UDC 62-50

CYBERNETICS AND COMPUTER ENGINEERING: DIGITAL CONTROL SYSTEMS

Kiev KIBERNETIKA I VYCHISLITEL'NAYA TEKHNIKA: DISKRETNYYE SISTEMY UPRAVLENIYA in Russian No 49, 1980 (signed to press 5 Sep 80) pp 98-100

[Annotation, table of contents and abstracts of articles in the book "Cybernetics and Computer Engineering: Digital Control Systems. Republic Level Interdepartmental Collected Papers", editor-in-chief V.M. Glushkov, 1,000 copies, 100 pages]

[Text] The solution of urgent problems in digital control systems using specialized computers and devices, as well as questions of the analysis and synthesis of optimal digital control systems, taking into account determinate and random external perturbations, are analyzed. A number of papers are devoted to the practical utilization of digital systems to control physical experiments and technological processes.

The collection is intended for scientific workers, graduate students and students in the senior courses of the higher educational institutes.

Table of Contents

UDC 62-505

THE SYNTHESIS OF OPTIMAL ADAPTIVE TRACKING SYSTEMS

[Abstract of article by Kuntsevich, V.M. and Lychak, M.M]

[Text] The problem of synthesizing an optimal tracking system for a certain unknown external signal which changes with time, generall speaking, in an arbitrary fashion, is analyzed. Optimal control is sought from the solution of a particular minimax problem. A procedure is described for the construction of estimates of the system parameter vector, as well as an estimate of the value of the unknown external signal. Figures 2; references 6.

UDC 62-50

FILTERING A MULTIPLE VALUED PROCESS IN LIMITED NOISE

[Abstract of article by Bakan, G.M.]

[Text] A nonstatic variant of the filtration problem is treated for an evolving process with indeterminate elements (a multiple valued process) which is observed under conditions where it is exposed to limited interference of an unknown nature. A nonstatic analogue of Bayes' formula is derived, and sub-optimal analogs of a Kalman-Busy filter are constructed based on this analog. Figures 1; references 6.

UDC 62-50

ON ONE MATHEMATICAL MODEL FOR DIGITAL CONTROL SYSTEMS WHICH ELIMINATE THE PROBLEM OF STRUCTURAL LIMITATIONS

[Abstract of article by Quntsevich, V.M. Lychak, M.M. and Sukennik, A.A.]

[Text] The problem of constructing a mathematical model for the process of digital computer control of a linear dynamic object. As a result, a discrete model of such a control system was constructed, which describes the motion of the controlled object at discrete points in time. In contrast to the well known discrete models, a special choice of the difference phase space is realized, which elminates the problem of structural limitations. References 7.

UDC 621.317.001.2:681.3.06

THE DETERMINATION OF THE WEIGHTING FUNCTION OF A LINEAR OBJECT WHERE A RANDOM PROCESS GENERATOR IS USED AS THE TEST SIGNAL SOURCE

[Abstract of article by Konson, Ye.D. and Malinin, L.M.]

[Text] An optimal processor procedure for the results of an active experiment, which uses a steady-state random process with a specified correlation function as the test signal is proposed for the solution of the problem of linear object identification. The procedure makes it possible to boost the precision of the identification problem solution. Figures 1; references 7.

UDC 62-50

A TECHNIQUE FOR CALCULATING THE OPTIMAL CONTROL FOR NONLINEAR DISCRETE OBJECTS USING AN ORTHOGONALIZATION PROCEDURE WHERE LIMITATIONS ARE PRESENT

[Abstract of article by Boychuk, L.M.]

[Text] A method of designing an optimal control program for a nonlinear discrete object is analyzed where limitations are present on its input and output quantities. The basis for the computational algorithm is an optimization method of the differential descent type with controlled processes for processing the limitations and the movement to the extremum. In this case, it is necessary to solve a special sub-definite system of linear algebraic equations at each iteration of the calculation, the coefficients of which are determined by the limitation gradients (the equations of the objects and the limitations on its input and output values), while the right sides of the equations are determined by the disparities of the indicated limitations. The use of an orthogonalization procedure in the the solution of such a system of equations makes it possible to substantially reduce the requisite immediate-access digital computer memory, especially when taking into account the structure of the limitations which is characteristic of discrete control objects. References 7.

UDC 62-50

ON THE NUMERICAL SYNTHESIS OF OPTIMAL CONTROLLERS FOR NONLINEAR DIGITAL CONTROL SYSTEMS

[Abstract of article by Valeyev, K.G. and Finin, G.S.]

[Text] Auxiliary information on the theory of integral manifolds of difference equations are presented. Nonlinear projectors of the solutions are introduced and some of their properties are studied, which substantiate the proposed technique for the numerical solution of Bellman's equation for the synthesis of an optimal controller. The principle of an optimal manifold is proposed, the use of which makes it possible to reduce to the problem of synthesizing optimal control in the general case to an interpolation problem. Figures 3; references 7.

UDC 62-504-4:519.2

MARKOV PROCESSES IN AUTOMATED CONTROL SYSTEMS WITH PULSE-FREQUENCY MODULATION OF THE SECOND KIND

[Abstract of article by Dubas, V.I.]

[Text] A procedure for studying systems with pulse-frequency modulation of the first kind is generalized to systems with PFM of the second kind. The transformation of a one-dimensional or multi-dimensional Markov process realized by PFM-II is analyzed. Algorithms are given for the calculation of the transition distribution densities of the output signals based on the known statistical characteristics of the input signal and the specified modulation law. References 9.

UDC 62-503.4

ON THE SOLUTION OF THE PROBLEM OF SYNTHESIZING ASYMPTOTICALLY STABLE NONLINEAR PULSE CONTROL SYSTEMS

[Abstract of article by Andreyeva, Ye.V. and Volosov, V.V.]

[Text] Questions of the dynamics of closed nonlinear pulse control systems for stable and neutral linear steady-states objects are studied. The problem of synthesizing the modulation laws governing the control pulses is solved, where these control pulses are optimal in terms of Lyapunov function damping. References 6.

UDC 62-50

THE IDENTIFICATION OF STOCHASTIC OBJECTS

[Abstract of article by Makarchuk, M.M.]

[Text] A method of determining the characteristics of sotchastic objects is developed based on the representation of models of the objects in the form of an orthogonal series of nonsteady-state periodic functionals with random coefficients. Tables 1; references 9.

UDC 62-503.4

THE STABILITY OF NONLINEAR PULSED SYSTEMS FOR THE CASE OF CONSTANTLY ACTING PERTURBATIONS

[Abstract of article Movchan, L.P.]

[Text] Questions of the stability of nonlinear pulsed systems are analyzed for the case of constantly acting perturbations. A stability theorem is proved for the case of constantly acting perturbations for asymptotically stable nonlinear pulsed control systems. References 8.

UDC 62.50

AN ALGORITHM AND SPECIFIC FEATURES OF THE DIGITAL COMPUTER REALIZATION OF A TECHNIQUE FOR THE SOLUTION OF NONLINEAR PROGRAMMING PROBLEMS WITH CONTROLLED PROCESSES FOR THE PROCESSING OF LIMITATIONS AND THE MOVEMENT TO THE EXTREMUM

[Abstract of article by Boychuk, L.M. and Shkvarun, N.M.]

[Text] A modified digital variant of a technique for the solution of nonlinear programming problems with controllable processes for the processing of limitations and the movement to the extremum, as well as the specific features of the digital computer realization of the technique are treated. Figures 5; tables 1; references 7.

UDC 62.52

ON ONE RECURRENT ALGORITHM FOR ADAPTIVE CONTROL OF A STATIC OBJECT

[Abstract of article by Zhitetskiy, L.S.]

[Text] The problem of constructing a stochastic analog for a recurrent finite converging adaptive control algorithm for a static object where the control is based on perturbations is analyzed for the case of a discrete set of control actions. Some of the specific features of the adaptation procedure are discussed. The sufficient conditions for algorithm convergence which is most likely true after a finite number of steps are formulated and proved. Figures 3; references 7.

UDC 62-50

THE IDENTIFICATION OF THE PARAMETERS OF THE MATHEMATICAL MODEL OF A COMPLEX FRACTIONATING COLUMN

[Abstract of article by Bakan, G.M. and Tarnovskiy, Yu.P.]

[Text] The applications of Bayes method to the problem of estimating the parameters of a mathematical model for a complex fractionating column is analyzed. Parameter estimates are calculated from observations of the output and intermediate coordinates, given the condition that the observations of the output coordinates are made an integer number of times less often than the observations of the intermediate coordinates. A recurrent form of the algorithm is derived for the solution of the problem for a model which linearized with respect to the parameters being estimated. Figures 1; references 5.

UDC 620.178.5.05-52

MODULAR PROGRAMMING AND STRUCTURE OF THE SOFTWARE FOR A VIBRATION TESTING AUTOMATED CONTROL SYSTEM

[Abstract of article by Trachuk, T.S. and Tunik, A.A.]

[Text] The construction principles for the software of digital automated control systems for vibration tests of mechanical systems based on modular programming are treated. The major operational modes of the automated control system for the vibration tests as well as the program modules which provide for the execution of these modes are described. A structure is proposed for a control system which determines the operational sequence of the program modules and the links between them. The software considered here is realized using series produced control minicomputers. Figures 4; references 5.

UDC 681.355

A SPECIALIZED PROGRAMMABLE PROCESSOR FOR A DIGITAL CONTROL SYSTEM FOR A SPECIAL MATRIX OF A RANDOM VECTOR PROCESS

[Abstract of article by Petrovskiy, A.A.]

[Text] The digital control system for the spectral density matrix of a three--dimensional random vector process is analyzed. The place of the specialized processor in the system is determined and its structure is described. The precision in the generation of the spectral density by means of the special processor is analyzed and a way of scaling the weighting function of the digital filter is proposed, which makes it possible to reduce the error in the spectral density generation without increasing the bit length of a machine word. Figures 4; references 8.

UDC 621.391:621.311

ON THE QUESTION OF THE DETERMINATION OF THE NOISE IMMUNITY OF DIGITAL SIGNAL TRANSMISSION

[Abstract of article by Skryl', V.F.]

[Text] The problem of determining the probability of unit symbol distortion in a binary steady-state or periodic steady-state channel without a memory is analyzed, where the distribution of the interference in the channel differs from a normal distribution and is specified in the form of magnetograph recordings. An algorithm is proposed for the solution of the given problem on a digital computer using an analog-digital converter. Figures 2; references 5.

8225
CSO: 1863/150

TABLE OF CONTENTS FROM THE JOURNAL 'AUTOMATION AND COMPUTER TECHNOLOGY'

Riga AVTOMATIKA I VYCHISLITEL'NAYA TEKHNIKA in Russian No 1, Jan-Feb 81

[Text] Table of Contents

8225
CSO: 1863/126

UDC 621

'INFORMATION SELECTION AND TRANSMISSION' NO 61

Kiev OTBOR I PEREDACHA INFORMATSII in Russian No 61, 1980 (signed to press 27 Nov 80) p 101

[Table of contents from collection "Information Selection and Transmission", edited by the Interdepartmental Collections Editorial Board, consisting of L. Ya. Mizvuk (editor-in-chief), Ya. Ye. Belen'kiy, B. I. Blazhkevich, Yu. N. Bobkov, V. I. Gordnenko, F. B. Grinevich, V. V. Gritsyk (deputy editor-in-chief), V. G. Zubov (secretary), N. I. Kalashnikov, V. P. Marinets, M. A. Rakov, and A. N. Svenson, Physico-Mechanical Institute, Ukrainian SSR Academy of Sciences, Izdatel'stvo "Naukova dumka", 1000 copies, 101 pages]

[Text] CONTENTS Page

2174
CSO: 1863/156

UDC 621

'INFORMATION SELECTION AND TRANSMISSION' NO 62

Kiev OTBOR I PEREDACHA INFORMATSII in Russian No 62, 1980 (signed to press 4 Dec 80) p 118

[Table of contents from collection "Information Selection and Transmission", edited by the Interdepartmental Collections Editorial Board, consisting of L. Ya. Mizvuk (editor-in-chief), Ya. Ye. Belen'kiy, B. I. Blazhkevich, Yu. N. Bobkov, V. I. Gordnenko, F. B. Grinevich, V. V. Gritsyk (deputy editor-in-chief), V. G. Zubov (secretary), N. I. Kalashnikov, V. P. Marinets, M. A. Rakov, and A. N. Svenson, Physico-Mechanical Institute, Ukrainian SSR Academy of Sciences, Izdatel'stvo "Naukova dumka", 1000 copies, 118 pages]

[Text] CONTENTS Page

2174
CSO: 1863/156

UDC 532.546

NEW BOOK ON COMPUTER MODELING OF FILTRATION IN WATER ENGINEERING WORKS

Kiev RASCHET NA EVM NESTATSIONARNOY FIL'TRATSII V RAYONAKH GIDROTEKHNICHESKIKH SOORUZHENIY in Russian 1980 (signed to press 30 Jul 80) pp 2-5, 182

[Annotation, foreword and table of contents from book "Computer Calculation of Nonsteady-State Filtration in Regions of Water Engineering Works", by Aleksandr Aleksandrovich Dobronravov, Vitaliy Semenovich Kremez and Vladimir Stepanovich Siryy, Izdatel'stvo "Naukova dumka", 1,000 copies, 184 pages]

[Text] Solutions are given in this monograph for the profile and plan-view problems of nonsteady-state filtration in inhomogeneous soils for hydraulic and hydromechanical formulations, obtained by a finite-difference technique with complex boundaries of the filtration regions and boundary conditions. Numerical solution algorithms are described. An approximate method is presented for the solution of problems of nonsteady-state spatial filtration, based on the combined use of computers and modeling devices. Filtration calculation examples are given for regions of water management and land improvement facilities.

The book is intended for scientific workers and engineers engaged in filtration research and calculations, related to water engineering works construction, as well as for graduate students and students of advanced courses in the relevant specialties. Some 60 figures, 16 tables and bibliography: pp 173-181 (161 citations).

Foreword

The construction of water engineering works and land development systems occupies an important place in the development of the national economy of the USSR. The design of large water reservoirs and tailing dumps, as well as irrigation and drainage systems, brings about a change in the position of the free surface of the ground waters and the pressure heads in soil layers close to the surface of the ground. The increase in the ground flow level, which is due the filling of a water reservoir or tailings dump, can lead to ground water saturation of populated areas and the formation of boggy agricultural land. Highly mineralized water from tailing dumps, by intruding into ground waters, can lead to the pollution of water sources, surface water currents and to the salinization of agricultural lands with a subsequent loss in their fertility. For this reason, an important task in the project planning of water reservoirs and tailing dumps is the study of water filtration and the development of antifiltration and drainage measures which prevent or reduce the negative impact of water reservoirs and tailing dumps on the environment. The operation of various drainage and water intake systems also leads to a change in the levels and pressure heads of ground waters. Vertical boreholes have become widespread for water intake and water lowering purposes, where systematic vertical drainage is frequently employed in irrigated and drained massifs for water level reduction. In a number of cases, it is necessary to study the movement of ground waters to circular wells or close to excavations having a shape close to circular.

Inhomogeneously layered strata with alternating poorly and quite penetrable layers are characteristic of the natural conditions for the construction of water works. The water permeable portions of drainage trenches, canals, boreholes, etc., can be located in the various layers of an inhomogeneous water bearing stratum. For example, the drainage can be a combination type: the upper layer of the soil can be drained by trenches, while the subjacent layers can be drained by straight-line rows of boreholes.

Infiltration from the ground surface due to natural precipitation and a watering of irrigated ground, located close to water works, exerts a significant influence on the position of the free surface of the filtration flow. Since water engineering and land development facilities, depending on the economic needs and climatic conditions during various times of the year, are operated in different ways, the movement of ground waters as a consequence of their operation is, as a rule, of a nonsteady-state nature.

The existing solutions for nonsteady-state filtration problems, both analytical and numerical techniques, contain a number of substantial limitations on the form of the filtration region, the nature of the ground inhomogeneity, the variations in the boundary conditions, etc. For this reason, the further development of filtration calculation methods which maximally take into account natural factors, the arrangement of the water permeable portions of a structure and the nature of their operation is of great practical importance for the correct forecasting of the levels of ground waters, the determination of filtration losses from water reservoirs and tailing dumps, as well as using these for the basis of the development of efficient and economical antifiltration and drainage devices.

Methods of numerical solution with computers for problems of planar, plan-view and axially symmetric nonsteady-state filtration with the free surface in inhomogeneously layered soils with alternating poorly and quite permeable layers, as well as soils with arbitrary inhomogeneities, which were developed by the author over the past years in the Institute of Fluid Mechanics of the Ukrainian SSR Academy of Sciences are presented in the book.

Solutions of problems of planar nonsteady-state filtration are obtained in both hydraulic and fluid mechanics formulations. The application of the hydraulic model yields less precise values of the pressure heads and rates of flow of the filtration flow, however, it simplifies the calculations and saves a considerable amount of computer time. The utilization of the fluid mechanics model makes it possible to find more precise values of the filtration flow parameters and construct a detailed hydrodynamic grid, something which is particularly important in studying the flow in the near-drain region of the structure, but requires a greater operational speed and considerable volume of the computer memory. For this reason, one must work from the requisite precision of the results and the capabilities of the existing computers when choosing the mathematical model for filtration calculations.

A number of new problems of plane and axially symmetric filtration in inhomogeneously layered soils with alternating poorly and quite permeable layers are treated in a hydraulic formulation in the first chapter of the book. A method is developed in the second chapter for smoothing local singularities for the solution of problems of planar nonsteady-state filtration in a hydromechanical formulation in soils with arbitrary inhomogeneities. An approximate method is presented in the third chapter for solving problems of nonsteady-state spatial filtration, based on the joint utilization of a computer and modeling devices (EGDA) [electrohydrodynamic analogs]. The solutions of problems of planar nonsteady-state filtration in regions containing water basins of a complex configuration are analyzed. Examples are given for the calculation of filtration in regions of water management and land development facilities.

The problem solutions treated in the first and second chapters were obtained by a finite-difference technique with arbitrary boundaries of the filtration region and specified conditions placed on them. The finite difference equations are solved by a matrix run-through method (Chapter 1) and a block interation method (Chapter 2). Special attention is devoted to the approximation of the boundary conditions and the description of the numerical solution algorithms. Examples are given for the calculation and investigation of the filtration cases considered here. All of the calculations were performed on BESM-4M and BESM-6 computers using programs compiled by the authors in the ALGOL and FORTRAN algorithmic languages.

Unfortunately, the limited volume of the monograph did not allow for the presentation of texts of larger universal programs, which were used to run the calculations. However, there is no need in a number of cases of solving nonsteady-state filtration problems in their entirety, as is done in the monograph. For this reason, the authors hope that the detailed description of the numerical solution algorithms presented in the book will allow specialists with a mastery of programming languages to easily compose programs for the solution of specific problems in algorithmic languages convenient for them, which are oriented towards the capabilities of the computers available to them.

The first chapter of the monograph was written by V.S. Kremez, the second chapter and §3 of the third chapter were written by V.S. Siryy, while §1, 2 of the third chapter were written by A.A. Dobronravov.

The authors would like to express their sincere gratitude to professors N.N. Belyashevskiy and A.A. Glushchenko for a number of criticial comments made during the examination of the manuscript.

Table of Contents

8225
CSO: 1863/151 - END -