JPRS 81784

16 September 1982

USSR Report

CYBERNETICS, COMPUTERS AND AUTOMATION TECHNOLOGY

No. 62

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.

PROCUREMENT OF PUBLICATIONS

JPRS publications may be ordered from the National Technical Information Service (NTIS), Springfield, Virginia 22161. In ordering, it is recommended that the JPRS number, title, date and author, if applicable, of publication be cited.

Current JPRS publications are announced in Government Reports Announcements issued semimonthly by the NTIS, and are listed in the Monthly Catalog of U.S. Government Publications issued by the Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402.

Correspondence pertaining to matters other than procurement may be addressed to Joint Publications Research Service, 1000 North Glebe Road, Arlington, Virginia 22201.

JPRS 81784

16 September 1982

USSR REPORT

CYBERNETICS, COMPUTERS AND AUTOMATION TECHNOLOGY

No. 62

CONTENTS

SOFTWARE

APPLICATIONS

CEMA INTEGRATION IN CREATION AND USE OF COMPUTERS

Moscow PRAVDA in Russian 29 Apr 82 p 4

[Text] Electronic computer technology... A world of large and small computers, integrated circuits, microprocessors. In a word, the leading edge of scientific-technical progress. Recently at the regular meeting of the CEMA Ispolkom questions of cooperation in the development and widespread use of microprocessor technology in the economy were discussed. Letters are being sent to the editor, in which the readers are asking for an explanation of what value computers have in the countries of socialist cooperation, and how cooperation is developing in this field.

Ulitsa Chaikovskogo, Number 11. The Coordinating Center of the Intergovernmental Commission for Cooperation between the Socialist Countries in the Field of Computer Technology. Mikhail Yevgen'yevich Rakovskiy, the director of the Center relates:

"The integrated program of socialist economic integration is providing for the hastened creation and effective use of computers. The application of computers is helping to raise the productivity of labor, to increase the return of funds, and the main thing, to free the individual from tedious, monotonous operations.

"In order to reduce expenditures of time and material expenses in computer production in various countries of socialist cooperation, the constant and precise coordination of work in this field is necessary. Our Coordinating Center is figuratively called the headquarters of the Intergovernmental Commission. It brings together and guides the efforts of many thousands of designers, scientists, and workers in the eight socialist countries that are participating in the creation of computers.

"I will cite more data," our speaker continues his discussion. "As a whole in the socialist countries billions of rubles worth of computer technology are now being produced per year. Already hundreds of thousands of people are occupied with the operation of the whole stock of computers. That's why the most important question today is how to increase the effectiveness of the application of computers, and to decrease expenditures for their production.

"This is not an easy task, and it requires improving scientific-research work, increasing the reliability of the systems that are being produced, and preparing qualified personnel.

"Over the 12 years of coordinated work by the CEMA member countries that signed the agreement on cooperation in this field in 1969, much has been achieved. A mechanism for mutual action has been laid down, a single technical policy has been worked out, and scientific-research and design work is being carried out in an agreed-upon manner. The Unified Computer System and the Mini-Computer System has been created. Each of these systems provides complete compatibility of computers, and opens wide horizons for specialization and cooperation.

"In all this work it is important to emphasize the leading role of the collectives of the Soviet Union which are being led by V. V. Przhiyalkovskiy and B. N. Naumov, the chief designers of the computers. Zh. Zhelezov (Bulgaria), Zh. Narai (Hungary), G. Merkel' (GDR), B. Pivovar (Poland), and I. Brany (CSSR), the chief designers of the socialist countries, also had quite a bit to do with the whole affair.

"The division of labor in this field continues to develop successfully. The level of unification and standardization of hardware and software is continuously increasing, and the quality of equipment and systems is being raised. Bulgaria, for example, is improving the output of external memory devices year by year. In the GDR, alongside of the Robotron association, the "Karl Zeiss Jena" enterprise that is famous worldwide is participating in the creation of memory devices. Czeckoslovakian-produced machines with speeds of 20,000 and 60,000 operations per second are being modernized at two large associations--"Tesla", and ZAVT. Hungary is continuing to master the production of small computers and displays.

"At present in the countries of socialist cooperation great significance has been attached to the output of mini- and micro-computers. The improvement of microelectronic technology has made it possible to put several hundred thousand transistors and other elements on one silicon crystal. Thanks to this micro-computers are being created on the basis of large-scale integrated circuits that are smaller that a match box. In productivity they lag somewhat behind the large machines, but at the same time they are comparatively cheap, and convenient for use and service.

"Microprocessors are called catalysts of scientific-technical progress. They are providing for the active resolution of social questions transferring to the machines what is its and leaving to the man what is his. These "smart babies" work particularly successfully in the so-called embedded systems of automated control - in machine tools with digital program control, in manipulators (robots), in steel melting furnaces, in welding machines, and in the spinning and sewing industries.

"In the near future," says the director of the center in conclusion, "micocomputers with the so-called 'professional intellect' will have enormous prospects. These simple and cheap machines will provide for a sharp increase in the speed of technical progress. Qualified personnel is necessary in order to have such a technology. Over the years of cooperation of the participant countries of the agreement over 220,000 specialists in electronics, computer technology, and computer mathematics have been prepared in higher educational institutions."

With this the conversation at the Coordinating Center was ended. Let's go now to a different address to see how computers are working for socialist cooperation.

Kitayskiy Proyezd, Number 7. The Ministry of Power and Electrification. The Central Dispatching Administration (CDA) of the USSR Unified Energy System. We go up in the elevator and enter the spacious main dispatching center. Vladimir Germanovich Ornov is the head of the computer technology service.

"The Ministry of Power and Electrification," he relates, "is one of the earliest and most 'avid' users of computers. And this is understandable. Today energy flows, including to the brotherly countries, are reaching huge quantities, their speed is instantaneous, and the number of links that comprise the network of the international electrical system "Mir" is great. We couldn't get by without electronics in the control of such a complicated system. Hundreds of computers are now in operation in the Ministry of Power and Electrification, including practically all models of machines that have been produced in the socialist countries.

We are seated in front of one of the displays which bears the Unified System brand. Vladimir Germanovich presses several keys, and a schematic of the power flows across the western boarders of the USSR into Poland, Hungary, Czechoslovakia, Romania, and Bulgaria appears.

"Electricity is also good," explains our speaker. "Its quality not only depends on the maintenance of the necessary power flow, but also on its frequency. From the numbers on the screen it is clear that the computer is recording the normative, physical, and expected quantity of power and the frequency of the electrical current in the "Mir" system. It is simultaneously an advisor and forecaster, and aids in monitoring the flow of power, calculating its quality, ensuring the effectiveness of economic indicators, and, it goes without saying, making optimal decisions when breakdowns occur.

"Let's imagine for a second that in a certain large station in one of the European brother countries some kind of breakdown has occured, or let's say, a storm starts to pour on a power transmission line. The street lights are put out, factories come to a halt, the trains freeze... In order that this doesn't take place, a preventative electronic device is in operation. In a second along the wires of the international energy system help arrives from neighboring countries. In the same way computers react with the speed of lightning to peak loads in the morning and evening hours, when the use of energy sharpky rises.

...On the way out of the building of the CDA one's gaze lingers in the vestibule on the electronic display of the Hungarian firm "Vizinform". On it was a lit announcement about a meeting of the workers with ex-world chess champion M. Botvinnik called "Chess without energetics."

"Will an electronic partner for Karpov, a computer-grandmaster soon appear?" I jokingly ask our speaker.

"Apparently there's a long way to go," Vladimir Germanovich answered. But it seems like "micro-professors" are making advances for real. Computers have already been created that can read text and execute voice commands.

The saturation of contemporary life with computers places important questions before the brotherly countries: how to best provide servicing for a complicated technology, to prepare personnel for computer centers, to exchange experience?

Pyatnitskaya, 67 - yet another point on the map of the capital that interests us. The Bulgarian technical center of electronics and electro-technology in the USSR. Viktor Nikolov, the director, says that the workers at the center provide technical assistance in installation, servicing, and repairs under guarantee of Bulgarian computer technology, and cooperate in the creation of software. At present, practically all the systems that were purchased in Bulgaria are being serviced by Soviet specialists. Many of these studied at evening courses at the Moscow center. In order to familiarize people with the new Bulgarian products, exhibitions of equipment are being held here, applications packages are being demonstrated, and consultations are being organized.

Then V. Nikolov shows the instructional classrooms, where specialists are being prepared in all of the Bulgarian equipment that has been supplied to the USSR. Already around 4,000 people have been trained in the time that the center has existed. And if it used to be that mainly operators, programists, and service personnel came here from various places in the USSR, now skills are being acquired at the courses, it is mainly teachers who are raising their qualifications and then go on to train specialists.

The exchange of experience is also being resolved. Bulgarian engineers and technicians in their turn study, for example, in night courses in Moscow, Minsk, and Kazan'.

"What tasks has the collective of the Moscow technical center set before itself for the next few years?"

"Of course, the main thing is to preserve a reputation as a reliable firm, to ensure a high quality of service and preparation of cadres. And then, to cooperate with Soviet creators of electronics. Well of course we hope also to broaden purchases in the USSR of new Bulgarian computer goods."

The meeting ended with an inspection of the permanent exhibition. The tastefully constructed displays relate that before 1965 the production of electronic-computer technology in Bulgaria did not exist at all. The republic set out on the path of the development of a new branch. In the 1970's, 15 factories were erected here. The specialized "Izot" Association was organized.

Bulgaria has specialized in the construction of external memory devices. Its engineers, along with their Soviet colleagues, mastered the production of such complicated products, as central processors, magnetic tape and disk drives,

teleprocessing devices, and microprocessor equipment. And all of this is the result of the agreed upon actions of the socialist countries in the area of specialization and cooperation.

With this our short excursion from address to address, from person to person who can talk about how modern computers are serving socialist integration draws to a close. Of course, not everything on this path is smooth. Many problems still await their resolution or have only been solved in part. For instance, work on the further unification of products is going on, and the goal of eliminating parallelism in the production of computer technology is being pursued. There is slack for lowering expenses and increasing the quality of computers.

The pains of growth... There is no doubt that they will be overcome. Because the main thing has been laid down - an international collective of designers, engineers, workers, and technicians has been created. They have a single goal: to put electronics at the service of socialism, and to strengthen the integration of the brotherly countries.

9946
CSO: 1863/156

PROBLEMS IN TRAINING SPECIALISTS IN COMPUTER AIDED DESIGN

Moscow VESTNIK VYSSHEY SHKOLY in Russian No 3, Mar 82 pp 42-46

[Article by Professor V. I. Salyga and S. V. Kurov, "Improve the Training of Specialists in SAPR [Computer-Aided Design Systems]"]

[Excerpts] The Main Directions for Economic and Social Development of the USSR for 1981-1985 and the period to 1990 call for "expanding the automation of design and scientific research work by using electronic computers." In this connection, training personnel in computer-aided design (SAPR) [CAD] is assuming special importance and relevance.

Inspection results have shown that training of CAD developers is being successfully implemented in a number of the country's VUZ's. This effort is being carried out especially intensively and successfully at the MAI [Moscow Aviation Institute], the MIFI [Moscow Engineering Physics Institute], the MIEM [Moscow Institute of Electronic Machinery], the MVTU [Moscow Higher Engineering School] and the MEI [Moscow Power Engineering School]. Instructors at these VUZ's have established close business contacts with organizations and enterprises engaged in developing CAD systems.

Organizing training of CAD specialists has required VUZ's to reequip departments and laboratories with modern computers and peripherals. On this basis, implementation of an interVUZ program on development of training-research CAD systems has begun.

Analysis of the training of CAD developmental engineers has revealed not only the positive aspects, but also a group of basic problems. Raising the quality of training specialists in this field is fundamentally dependent on solving these problems. Thus, effective implementation of the training process is hindered by the lack of standard training plans and programs for the disciplines in this field. Essentially, each VUZ that trains CAD developers now has its own plan for this specialization, and the set of disciplines even within one specialty differ substantially for different institutes. With that, not one training plan for the specialties with regard to these specializations has been approved by the USSR Minvuz [Minstry of VUZ's]. At the same time, at the institutes inspected by the State Inspectorate, there were no standard training plans for the corresponding specialties with regard to specializations in the CAD field that had been developed and approved by the ministry. Thus, at the Kalininskiy Polytechnical Institute, which does not have sufficient experience in training CAD engineers,

instructors and appropriate equipment, training of CAD developers has been carried out for 3 to 4 years according to an unapproved training plan. Lack od due monitoring has led to CAD specialization disciplines corresponding more to the training of CAD system users than to developers. Personnel at the Moscow Institute of Electronic Machinery have been operating for five years according to unapproved training plans.

The required rates of development of CAD developmental engineers are also being hindered by low executive discipline at a number of VUX's that were inspected. Thus, despite the ministry's directive on developing CAD specialist training, it has been poorly implemented at the Polytechnical Institute, the Institute of Shipbuilding and the Institute of Precision Mechanics and Optics in Leningrad. When inspected, they had no approved training plans and programs for the disciplines of specializations in CAD development. There were no standard programs on a number of profile disciplines of specialties, on the base of which CAD developers are being trained, at the Chelyabinsk and Kalininskiy Polytechnical Institutes.

In a number of VUZ's it is not considered mandatory to meet to the full extent all requirements imposed on compilation and coordination of instructional documentation. Thus, when working programs were drawn up, the necessary coordination was not always made between the special disciplines and the disciplines of specializations, which led to unwarranted duplication of lectures. At some institutes, the working programs did not take into account the latest achievements in science and technology in CAD systems and in design and tooling-up for production.

It should be said that even our leading VUZ's are poorly engaged in the questions of improving the training of specialists in CAD development and in generalizing modern expertise in CAD.

In the VUZ's, organizational questions for training CAD engineers are still not definitively resolved, and the demand for these specialists in sectors of the national economy and individual regions is studied and analyzed inadequately. There has been essentially no effort in this direction at the Chelyabinsk and Kalininskiy Polytechnical Institutes. With that, as inspection results show, the corresponding regions are in need of skilled specialists to develop CAD systems.

It is now inconceivable that a modern engineer would lack knowledge in CAD methods, especially one working in computers and automated control systems. His professional potential for the future is determined primarily by the store of knowledge and skills gained in the department and school laboratories. In connection with this, evoking at least wonder is the attitude of some departments concerned with development of computers and automated control systems toward instruction in CAD principles and methods. Thus, at the Chelyabinsk Polytechnical Institute, only five percent of the lecture material for the course "Design of Computer Elements" has been devoted to the application of CAD facilities. These subjects are poorly covered in the lecture courses on the specialties of "Automated Control Systems" and "Applied Mathematics" at the Kiev, Chelyabinsk and Leningrad Polytechnical and the Leningrad Shipbuilding Institutes, as well as on the specialty of "Design and Production of Electronic Computer Equipment" at the Leningrad Institute of Precision Mechanics and Optics. As a result, only one fifth of the diploma and even fewer course projects ... on specialties associated with the application of automated control systems and computers at the institutes inspected covered CAD development.

A typical deficiency for the majority of VUZ's is the poor systems engineering direction of the training process. Little attention is paid to the principles of design and research of structures and to development of the components of software and the information base for these systems.

Practical laboratory training that incorporates the principles for mastering modern CAD facilities must play a special role in training CAD specialists. Yet in many VUZ's inspected, the theoretical training of students in CAD systems has not been reinforced to the required extent by meaningful laboratory and practical activities. Thus, there has been actually no laboratory work on CVD at the Kalininskiy Polytechnical Institute. In some VUZ's, the content of practical laboratory training does not fully correspond to the problems of training CAD developers. As inspection results have shown, this situation is largely due to poor equipping of training laboratories with modern computers and peripherals, microprocessors and CAD facilities, including displays and plotters. In the same Kalininskiy Polytechnical Institute, there are no displays at all, and in some VUZ's, the equipment available does not meet even half the requirements.

8545
CSO: 1863/193

RENTING OF COMPUTER TIME

Moscow TRUD in Russian 13 Apr 82 p 2

[Article by G. Sidorova, engineer: "Machine Time Is Being Rented"]

[Text] In the early days, when computers with the peculiarities of their designers went into production, there were not enough of them to supply all of the enterprises and organizations that needed them. Now the demand is being satisfied. Computers have become a mandatory possession of enterprises and institutes. There are so many of them, that it is a rare "Advertisements" section in the newspaper VECHERNYAYA MOSKVA that goes without an announcement: "Organization is renting out computer time..." Renting time... How is this coming about? Does a task for the computer have to be sought? In other words, are expensive computers sitting idle?

We called several of the numbers indicated in the announcements.

"The Main Computer Center of Minrechflot RSFSR [RFSFR Ministry of the Maritime Fleet] is renting out time on any day of the week..."

"The Main Information-computer Center of the USSR Ministry of Petroleum Refining and Petrochemicals Industry is renting out computer time..."

As it turned out, even the Computer Center of the Statistical Administration of Moscow is also putting computer time up for rent. Perhaps it's possible somewhere, but here the computer should have plenty to do.

Renting, renting, renting...

According to the current national norms, computers are supposed to do useful work, fulfilling tasks, for 15 hours per day. This load has been established by USSR Gosplan and the CSA [Central Statistical Administration] of the USSR.

What does it take in order to get one's own computer center? As S. N. Bushev, the head of the Main Administration for Computer Work of the USSR CSA, explained to us, not very much. Ministries are given the right to plan the creation and start-up of computer centers themselves. If, in their opinion, this or that organization needs to acquire its own computer center, an order for a computer is sent to USSR Gosplan. And, like mushrooms after rain,

separate computer centers and computer technology departments in scientific-research institutes appear. To what does such a "demographic explosion" in the computerized information sciences family lead? Computer technology is dispersed across ministries and branches without taking into account real possibilities and needs. Why, for example, in Tashkent, do the Main Information-Computer Center of USSR Glavsredazirsovkhozvodstroy [Main Central Asian Sovkhoz Water-building Administration] of the Ministry of Land Reclamation and Water Resources and the Information-Computer Centers of UzSSR Goskomvodstroy [expansion unknown] and of the Republic Ministry of Land Reclamation and Water Resources coexist side-by-side? After all, in practice they are solving the same problems, with which one branch center in the region could have coped.

> To have your own computer center is prestigious. Stocking up on the technology, the ministry and department seem to be competing with one another. And so in the localities they are contending for clients. After all, the norms for loading must be fulfilled. And any means to achieve this are fine. Those who are renting out time are not interested in what kinds of tasks are being solved on the computer by the renter. Maybe someone wants to calculate how many door handles there are in his institution? Please, the computer is at your service. If only the accounts were settled.

According to data of the USSR CSA, the nationwide average daily loading of large- and medium-scale computers is 11.3 hours. We emphasize - this is the average. In certain computer centers, however, computers are doing useful work only 5-6 hours. No one in these organizations is answering for the low effectiveness of the use of computer technology. No one has to. There is no single centralized organization in the country which would regulate its use.

But after all, isn't someone nevertheless responsible for this? Yes, again it's somehow the ministries themselves. It's well-known that when one has to be responsible for oneself, objective causes that diminish one's responsibilities are always found. Probably for this reason in certain branches and departments the computer centers are constantly amongst those that are lagging, which in no way can reflect favorably on their existence.

Unfortunately, there isn't yet any data for the average loading of computers for the whole year of 1981. But in the first half of last year for the USSR Ministry of Land Reclamation and Water Resources it was all of 8.3 hours. And it was close to that level in the preceeding year. For example, an ES-1022 computer in the information-computer centers Ministries of Land Reclamation and Water Resources in Belorussia and Kirghiziya do useful work all of six hours, and the amount for the even more powerful ES-1033 in a computer center of Glavrissovkhozvodstroy (Alma-Ata) [expansion unknown] was even less.

The causes? B. G. Shtep, the deputy minister of the USSR Ministry of Land Reclamation and Water Resources, feels that the problem is that the computer technology is primitive. We won't bother to argue with Boris Grigor'evich,

but on account of this we got interested in the opinion of specialists of the USSR State Committee on Science and Technology (SCST), Gosplan, and the USSR CSA. They noted that the Unified Series of computers, yes and even the Minsk-32s that are being replaced, are capable of solving all planning-economic and engineering-calculating tasks. But at the USSR Ministry of Land Reclamation and Water Resources, basically only tasks of an accounting nature are entrusted to the computer, whereas the circle of problems with which the ministry is concerned is very large. They are: construction and operation of units, the distribution of water, the development of design work. And so the main cause, if you will, is the fact that the task load from the ministry itself is too low.

G. Ye. Vodolazhskiy, the head specialist of the department of computer technology of USSR Gosplan in the amelioration branch, agrees with this. "The USSR Ministry of Land Reclamation and Water Resources," he adds, "is hardly the only ministry where monitoring of the use of computer technology is carried out outside the bounds of the central apparatus. It is assigned to the All-Union "Soyuzvodproyekt" Association [expansion unknown]. Naturally, in working with the computer subdivisions there is no organizational head.

This example only confirms it an extra time: it is time to change the existing order for the distribution of technology and the creation of computing subdivisions. Here is what B. G. Senyaninov, the assistant head of the Main Administration of computer technology of the SCST, said about this subject:

"USSR Gosplan and the SCST should plan the creation of new and the development of existing computer centers according to the requests made by the ministries and departments."

In our view, it would also make sense to take into account the data of the CSA: how much technology is concentrated within separate regions, and how effectively it is being used. But for the present the way in which technology is distributed remains as before. The dispersion of computers throughout the country continues.

And even in those regions, where there already is enough. How can the computer be helped so that it is not left "unemployed?"

A partial solution was found in Kiev. A dispatching service has already been operating there for five years. It redistributes computer time among organizations that do not have their own computers. It is convenient.

N. Ya. Gezel', head specialist of "Ukrremdorproyekt" [expansion unknown], one of the regular customers, admitted that it would be difficult for the institute to get by without the dispatcher service.

"On the average we need about four hours of computer time per day, and on different computers," he was relating. "It used to be necessary to look for days to find someone who would give computer time. The service does this very quickly, and in addition, selects a computer subdivision that is situated not far away."

At the present stage a dispatching service is necessary. Over five years the computer subdivisions in Kiev with its help have sold 76,000 hours of computer time for the sum of 5.5 million rubles. In the city idle time for computers because of the absence of work was slashed more than 40 percent.

The USSR Interdepartmental Council on Questions of the Improvement of Management in the National Economy approved the experience of the experimental information-dispatcher service for computer centers in Kiev. Taking into account this experience, the Ukrainian filial of the USSR NII CSA [The USSR Scientific-Research Institute of the Central Statistical Administration], the Institute of Cybernetics of the Ukrainian SSR Academy of Sciences, the All-Union Scientific Research Institute for Problems of Organization and Management of the SCSR, the Main Administration of computer work of the USSR SCA, and the computer center of the Statistical Administration worked out a standard working design for computer center dispatcher services (CCDS).

> Dispatcher services will work like middlemen between computer centers and organizations that need computer time. The services should pay for themselves. A calculation based on the example of Kiev showed: two percent of the calculated selling price of completed work (for each - from the computer center and from the user) is fully sufficient, in order that the service can not only exist, but also grow. In this five-year plan it is planned to create CCDS in a few more cities. The services need housing, transportation, and carefully worked out forms of mutual accounting with the users. But there is the apprehension that, although their first steps will be along a well-trodden path, there will be barriers. The fact of the matter is that the Kiev service until now has been working "unselfishly", not receiving any payment at all for services from its users.

Almost all the specialists with whom there was a chance to converse came together on one point: Dispatching will help to use computers more effectively. But the main thing, I think, is in the fact that it has the ability to hasten the creation of collective use computer centers (CUCC).

What is a CUCC? If one were to characterize them very briefly: they are centers which are outfitted with high performance computers and hardware, through the simultaneous use of which many users receive access to computing and information resources. We already have centers of collective use working in Leningrad, Minsk, Riga, Tallinn, Tomsk, Tyumen', Tula, and also in

Novosibirsk. They service budget organizations, industrial enterprises, and agricultural enterprises alike. It is much cheaper for users to make use of the resources of the CUCC, than if they had had to provide themselves with their own computer centers.

In the 11th Five-Year Plan the number of collective use computer centers will increase. Machines of new power will be needed. At the same time there is no ban on the creation of one's own computer centers both for ministries and departments, and even for potential aspirants for computer technology. It has come about that USSR Gosplan should approach the question of the distribution of computers even more attentively and critically, especially in those regions where, in the next few years, collective use centers will be.

9946
CSO: 1863/156

UDC 681.3.14./21.

SYNTHESIS OF AUTOMATA ON BASIS OF CIRCUITS WITH MATRIX STRUCTURE

Kiev UPRAVLYAYUSHCHIYE SISTEMY I MASHINY in Russian No 1, Jan-Feb 82 (manuscript received 3 Nov 80, after revision 21 Jul 81) pp 66-70

[Article by Valeriy Anatol'yevich Sklyarov, senior teacher, Minsk Radio Engineering Institute]

[Text] Introduction

Works [1, 2] have reviewed the methods of synthesizing microprogram automata (MPA's) on programmable logical matrices (PLM's) which are included in the class of circuits with matrix structure. This class also includes read-only (permanent) memory units [1]. The present article considers methods of synthesizing MPA's on the basis of read-only memory units [ROM's] and PLM's.

We will call the PLM with n inputs, m outputs, and q intermediate lines the PLM (n, n, q) [1]. The ROM can be described by two parameters [1] — the number of inputs n' and the number of outputs m'; we will call these read-only memory units ROM (n', m').

Suppose we have given MPA S [1], $A = \{a_1, \ldots, a_M\}$ — the set of internal states of the automaton, $X = \{x_1, \ldots, x_L\}$ and $Y = \{y_1, \ldots, y_N\}$ are the sets of input and output variables respectively. We will assign the MPA using direct transfer tables [1].

The automaton S can be constructed on one PLM(n, m, q) [2] if $(L + R) \leq n$, $(N+R) \leq m$, $B \leq q$, where $R = \text{int}\log_2 M$ is the bit configuration of the code of internal states and B is the number of lines in the transfer table of the MPA. In practice the first and second cited conditions are usually not met [2, 3]. The present article considers synthesizing such automata.

The Structure of an Automaton on PLM and ROM

Suppose $(L+R) \leq n$, $(N+R) > m$, $B \leq q$, $(N+R) \leq m'$, $b \leq m$, $b \leq n'$, and $b = \text{int}\log_2 B$.

Figure 1 below shows a diagram of the Mili automaton, where $\varphi_1, \ldots, \varphi_R$ are the functions of excitation of memory elements of register RG; $\tau_1, \ldots, \tau_R$ are feedback signals from memory elements of the MPA; and, the binary code of the number of the line in the MPA transfer table is transmitted along the lines $z_1, \ldots, z_b$.

Any line i of the transfer table of the Mili MPA assigns a transfer from state a_{im} (initial state of the MPA) to state a_{is} (transfer state of MPA) through the action of input signal $X(a_{im}, a_{is})$ and outputs the signal $Y(a_{im}, a_{is})$ of a certain subset of set Y [1]. To determine the number of line i of the transfer table in a deterministic automaton it is sufficient to know the initial state a_{im} of the MPA and the input signal $X(a_{im}, a_{is})$. Knowing the number of line i of the transfer table it is possible to determine uniquely the transfer state a_{is} of the MPA and the subset of output signals $Y(a_{im}, a_{is})$.

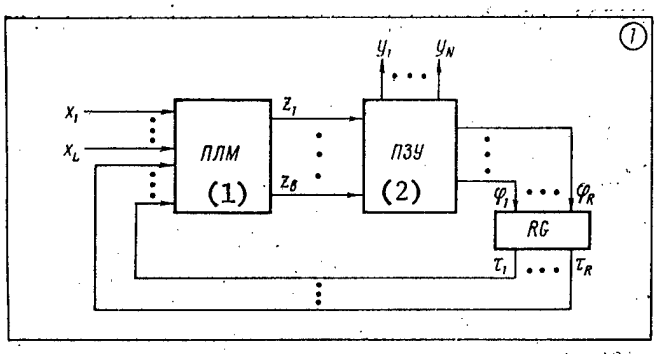

Figure 1. Structural Diagram of Mili Automaton on PLM and ROM.

Key: (1) PLM;
(2) ROM.

Depending on the values of the variables $x_1, \ldots, x_L$ and $\tau_1, \ldots, \tau_R$ the PLM in Figure 1 forms the code of the line number transmitted by lines $z_1, \ldots, z_b$. The functions $z_1, \ldots, z_b$ (lines and functions formed on lines are indicated by the same symbols here) are represented in random disjunctive normal form (DNF) and therefore can be realized efficiently on TLM's [1]. The functions $\varphi_1, \ldots, \varphi_R$ and $Y(a_{im}, a_{is})$ are represented in complete disjunctive normal form and can be realized efficiently on ROM's [1].

Synthesizing Automatons on PLM's and ROM's

Suppose $(L+R) \leqslant N$, $(N+R) > m$, $B \leqslant q$, $(N+R) > m'$, $b \leqslant m$, and $b \leqslant n'$. An MPA with the calculated parameters can be constructed in two ways. The first is to expand the ROM (n', m') by outputs [1, 2]. The second way is specially coding sets of output signals $y_1, \ldots, y_N$ [4, 5] and using a supplementary standard circuit for recognition of them.

Suppose $(L+R) > n$, $(N+R) > m$, $(N+R) \leqslant m'$, $b \leqslant m$, and $b \leqslant n'$. We will represent the MPA circuit for this case as shown in Figure 2 (next page). To construct the MPA

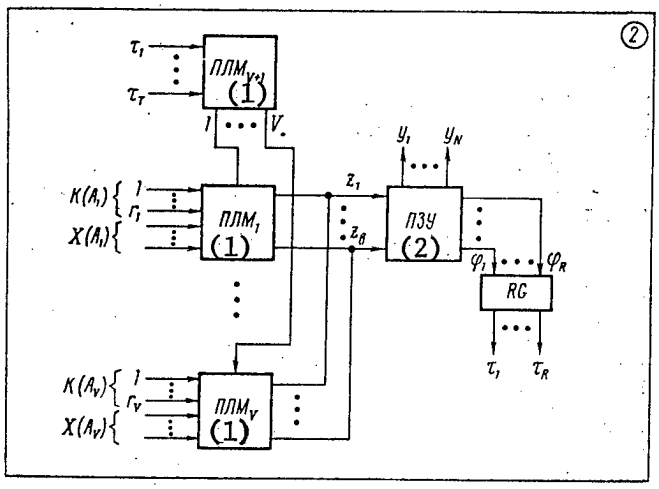

Figure 2. Structural Diagram of Automaton on PLM's and ROM's for $(L+R) > n$: $K(a_b)$ — r_d-Bit Codes of Automaton States Belonging to Block A_d of Decomposition π_A, d = 1, ..., V.

Key: (1) PLM;
(2) ROM.

---

on this circuit, the decomposition $\pi_A = \{A_1, \ldots, A_V\}$ is formed on the set of its states in such a way that $X(A_1) \cup \ldots \cup X(A_V) = X$ and either $X(A_i) \cap X(A_j) = \varnothing$, i, j = 1, ..., V, $i \neq j$ or their intersection has the minimum power. In this case $X(A_i)$ is a subset of the set X of input variables that significantly affect transfers from state belonging to block A_i of the decomposition π_A. After obtaining the decomposition π_A the MPA transfer table is divided into subtables whose number is equal to V. All the transfers from states belonging to block A_d are included in each subtable d. Then each subtable is realized on a separate PLM.

Works [1, 2] consider the method for obtaining π_A and constructing the MPA on PLM's according to this decomposition. When this method is used each PLM d must have at least $R+|X(A_d)|$ inputs ($|X)A_d)|$ is the power of set $X(A_d)$), but in this case it is not clear how to construct the MPA on PLM (n, m, q) if $n < (R+|X(A_d)|)$.

It is possible to reduce the number of inputs of each PLM by introducing an additional (V+1) PLM. We should note that this PLM realizes the functions of a standard decoder. We will encode the states in each decomposition block π_A with a binary code of length $r_d = \text{intlog}_2|A_d|$, d=1, ..., V; $r_d < R$ is almost always true. We also encode each decomposition block π_A with a binary code of length $T = \text{intlog}_2 V$. We represent the code of each state of the MPA in two parts: the T-bit code of the number of block π_A which includes the selected state, and the r_d-bit code with which this state is encoded in block d of decomposition π_A. Then the (V+1) PLM will have T inputs and V outputs. Using the code of the number of block π_A it will permit selection of the PLM_1, ..., PLM_V corresponding to this block. Usually $T \leq n$. If $V > m$, then the (V+1) PLM can easily be expanded in terms of outputs [2].

Introducing the (V+1) PLM is efficient, but it does not solve the problem of constructing the MPA on PLM (n, m, q) if the condition $(1+r_d+|X)A_d)|) > n$ is fulfilled for even one d. We will solve this problem by reducing the quantity $|X(a_d)|$ by introducing additional states [6].

Suppose for construction of the MPA transfer table using the techniques proposed in [1] the marking of the algorithm graph-diagram (AGD) is accomplished using markers $a_1, \ldots, a_m$. We will use an additional mark to indicate the input of the conditional apex to which there is a transfer from another conditional apex that does not pass through a statement apex, considering that this input is not marked (the latter is considered only when constructing a Mili MPA).

Figure 3 below shows the layout of an AGD from work [7] for constructing the transfer table of a Mili MPA. To obtain this table all the transfer paths among markers

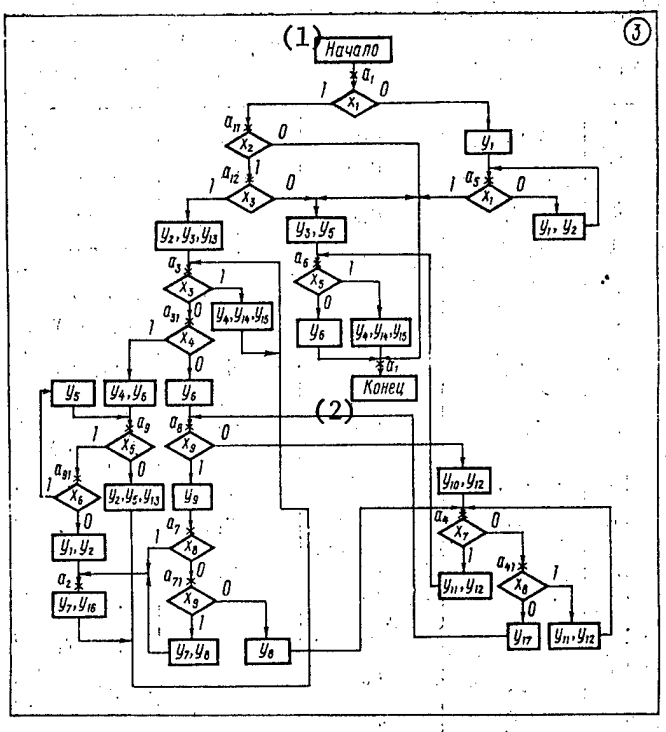

Figure 3. Example of the Layout of an Algorithm Graph-Diagram

Key: (1) Beginning;
(2) End.

---

a_i on the AGD are determined [1]. The transfer paths are formed in such a way that the condition $|X(a_i)| < 1$ is met for all i. Clearly this is possible with the layout we are considering. The transfer path to the additional marker always passes through conditional apexes only; therefore, an empty set of output signals forms on the corresponding line of the table. For graphic purposes additional markers

have double indexing. The first index indicates the number of the primary markers from which the path travels to the additional marker, while the second index is the number of the additional marker. With the resulting transfer table, taking account of remarks made earlier, it is possible to construct an MPA on PLM's with any values $n \geqslant 2$, m, q.

Now let us consider how to construct, using the table, an automaton which is optimal in the sense of minimum expenditure of PLM (n, m, q). For the sake of simplicity here, we will not consider constraints related to intermediate and output lines. But if there are not enough output and intermediate lines in certain PLM's, the methods of PLM expansion [2] can be used. We present below an approximate algorithm for construction of an MPA in which the input variables are distributed to PLM's in such a way that, as far as possible, each variable x_i is linked to the input of just one PLM. This leads to a situation where the total number of inputs for all PLM's is minimal. The algorithm includes the following actions.

Step 1. From column a_m of the transfer table any state of the MPA is selected for which $|X(a_i)| = \max$ (clearly, $|X(a_i)|$ is equal to either zero or 1).

Step 2. The set $A^d = \{a_i, a_j |$ of a_1 selected in Step 1 is formed; a_j is written in column a_m of the transfer table, $X(a_i) = X(a_j)$, i, j = 1, ..., E, $i \neq j\}$; E is the number of primary and additional markers on the AGD for the first execution of algorithm d = 1.

Step 3. If $X(A_a)|+|X(a^d)|+\text{intlog}_2|A^d \cup A_d| \leqslant n-1$ (for the first execution of the algorithm for each d, $A_d = \emptyset$ and $|X(a_d)| = 0$), then all states from set A^d are included in block A_d of the decomposition π_A; if this is not the case we remove elements from A^d until this condition is met, after which the remaining elements from set A^d are included in block A_d. In this and subsequent points of the algorithm, when including MPA states in block A_d of decomposition π_A, from the transfer table of the automotan we cross out all lines which contain in column a_m states belonging to a_d.

Step 4. If $|(X)A_d|+\text{intlog}_2|A_d| < n-1$ and $A^d = \emptyset$ or $|X(A_d)|+\text{intlog}_2|A_d| = n - 1$, we try to expand set A_d by certain states A_{im} entered in column a_m of the transfer table for which $X(a_{im}) = \emptyset$. We expand A_d as long as condition $|x(A_d)|+\text{intlog}_2 |A_d| \leqslant n - 1$ continues to be true. Then we move to Step 7. If $|X(A_d)|+\text{intlog}_2|A_d| < n - 1$ and $a^d \neq \emptyset$, then we go to Step 5.

Step 5. If transfers of type $A_iX(a_ia_{ij})$ a_{ij} (transfer from state a_i to state a_{ij} occurring under the influence of input signal $X(a_i, a_{ij})$) or type a_{ij} $X(a_{ij}, a_{ik})$ a_{ik}) occurs, then we call the state a_{ij} in the first case and state a_{ik} in the second generated states, while state a_i in the first case and states a_{ij} in the second are called generating states. We should note that considering the designations given in Figure 3, generated states must always have double indexing where the first index coincides with the first index of the corresponding generating state. The generating and generated states can be replaced by a single state because the marker of the generated state is always complementary. When constructing an MPA on PLM's we will try to consolidate as many generated and generating states as possible without increasing the total number of inputs in all PLM's. This reduces the number of internal states of the MPA and the number of lines

in the transfer table. In set A_d we will select all generating and generated states and consider that if the transfer $A_i X(a_i, a_{ij}) a_{ij}$ and $a_i \in A_d$, $d > 1$, a_{ij} A_g, $g = 1, \ldots, d - 1$ occurred in the initial transfer table of the MPA, then state a_i ceases being a generating state. By analogy, if $a_{ij} \in A_d$, $d > 1$, $a_i \in A_g$, $g = 1, \ldots, d - 1$, then state a_{ij} ceases to be a generated state. We include the generated and generating states from set a_d in set B_d. If $B_d = \emptyset$, then we must designate $A^d = \emptyset$ and go to Step 1; if $B_d \neq \emptyset$, we go to Step 6.

Step 6. It is necessary to define the variable X_k in order to make the decision on consolidating several (possibly all) generating and generated states. The variable x_k appears in the set $X(A_d)$ after consolidation of the states and inclusion of the result of consolidation in set a_d and set $c_k = \{a_j | a_j$ — written in column a_m of the transfer table, $X(a_j) = \{x_k\}\}$, because as many elements of set C_k as possible should also be included in A_d to reduce the number of inputs in all PLM's.

Next we should consolidate several (possibly all) states from set B_d with the generating or generated states corresponding to them. If all of the states from set B_d cannot be consolidated with the generated or generating states corresponding to them, we must search for that consolidation for which $|X(A_d)| + \text{intlog}_2|A_d| = n - 1$. Fulfillment of this condition leads to a situation where all inputs of the corresponding PLM will be used. We should note that the number of elements in set B_d is usually small and the appropriate consolidation can be found easily by examining a few alternatives.

When a_i and a_{ij} are consolidated we must attempt to see that all elements of the set C_k corresponding to the particular consolidation can be included in set A_d and that the condition $|X(A_d)| + \text{intlog}_2|A_d| \leq n-1$ is fulfilled. If generating states and the states generated by them appear in block A_d, they must be consolidated within the block.

If $|X(A_d)| + \text{intlog}_2|A_d| < n - 1$, we must go to Step 5; if $|X(A_d)| + \text{intlog}_2|A_d| = n - 1$ we go to Step 4.

Step 7. Block A_d of decomposition π_A is formed. Then the value d is increased by one and we fix $A^d = \emptyset$, $A_d = \emptyset$.

Steps 1-7 of the algorithm must be carried out until all lines are crossed out from the transfer table of the MPA; in this case the decomposition π_A will be fully formed.

Let us consider an example of use of the algorithm to construct an MPA using a transfer table on PLM(4, m, q). We should note that the methodology of works [1, 2] does not provide a technique for constructing such an automaton.

1. We select state a_1 for which $|X(A_1)| = 1 = \max$.
2. $A^1 = \{a_1, a_5\}$.
3. $X(A^1) = \{x_1\}$, $|X(a^1)| = 1$, $|A^1| = 2$, $|X(A^1)| + \text{intlog}_2|A^1| = 2 < n - 1$, because $n = 4$; therefore, $A_1 = \{A_1, A_5\}$. We remove lines 1, 2, 16, and 17 from the transfer table of the MPA and go to Step 5.

5. The transfer $a_1 x_i a_{11}$ occurs, with the state a_1 as the generating state, state a_{11} as the generated state, and $a_1 \in A_1$; therefore $B_1 = \{a_1\}$. $B_1 \neq \emptyset$, therefore we go to Step 6.

6. $x_k = x_2$, $C_2 = \emptyset$, because in the table x_2 is encountered only on transfers from state a_{11}. States a_1 and a_{11} can be replaced by a single state and it can be included in block a_1 because $X(A_1) = \{x_1, x_2\}$ and $|X(A_1)|+\text{intlog}_2|A_1| = 3 = n - 1$. Only state a_1 remains of the two states a_1 and a_{11}. We also remove lines 3 and 4 from the table. Because $|X(A_1)|+\text{intlog}_2|A_1| = n - 1$, we go to Step 4.

4. Set A_1 cannot be expanded because the addition of even one state to A_1 results in a situation where $|X(A_1)|+\text{intlog}_2|A_1| = 4 > n - 1$. Therefore we go to Step 7.

7. $A_1 = \{A_1, A_5\}$, $X(A_1) = \{x_1, x_2\}$.

Continuing by analogy we obtain $A_2 = \{A_{12}, A_3\}$, $X(A_2) = \{x_3, x_4\}$ on the condition that states a_3 and a_{31} are consolidated; $A_3 = \{a_4, a_7\}$, $X(A_3) = \{x_7, x_8\}$ on the condition that a_4 and a_{41} are consolidated; $A_4 = \{a_6, a_9\}$, $X(A_4) = \{x_5, x_6\}$ on the condition that a_9 and a_{91} are consolidated; $A_5 = \{a_2, a_{71}, a_8\}$, $X(a_5) = \{x_9\}$. Then $\pi_a = \{A_1, \ldots, A_5\}$.

We encode the states in each block: a_1, a_{12}, a_4, a_6 — 0; a_5, a_7, a_3, a_9 — 1; a_2 — 1 ; a_{71} — 00; a_8 — 10.

We give block $A_1, \ldots, A_5$ the following code: A_1 — 000, A_2 — 001, A_3 — 010, A_4 — 011, and A_5 — 1 _ _, where the blanks indicate that the value (zero or 1) of the corresponding code complement does not matter.

The diagram of the MPA is constructed of six PLM's, and ROM, and a four-position register. The outputs of each position of the register, $\tau_1, \ldots, \tau_4$, are used as follows: $\tau_1, \ldots, \tau_3$ — to assign the code of the block; τ_4 — to assign the code of the state in the first four blocks; τ_3, τ_4 — to assign the code of the state in the last block.

Let us note the additional possibilities of simplifying the diagram given in Figure 1 above.

By selecting certain codes of line numbers of the transfer table it is possible for some $k \in \{1, \ldots, b\}$ and $l \in \{1, \ldots, R\}$ to fulfill the condition $Z_k =$ for all lines of the transfer table. This makes it possible to remove certain functions of direct excitation from the PLM output, reducing the number of ROM outputs while the number of PLM outputs remains as before.

If $m-b > 0$, then some of the excitation functions $\varphi_1, \ldots, \varphi_R$ can be taken from the three outputs of the PLM's, which reduces the number of ROM outputs.

Transfer Table of Mili MPA

№ п/п (1)	a_m	a_s	$X(a_m, a_s)$	$Y(a_m, a_s)$
1	a_1	a_{11}	x_1	—
2		a_5	$\bar{x}_1$	y_1
3	a_{11}	a_{12}	x_2	—
4		a_1	$\bar{x}_2$	—
5	a_{12}	a_6	$\bar{x}_3$	y_3, y_5
6		a_3	x_3	y_2, y_3, y_{13}
7	a_2	a_3	1	y_7, y_{16}
8	a_3	a_{31}	$\bar{x}_3$	—
9		a_3	x_3	y_4, y_{14}, y_{15}
10	a_{31}	a_9	x_4	y_4, y_6
11		a_8	$\bar{x}_4$	y_6
12	a_4	a_{41}	$\bar{x}_7$	—
13		a_6	x_7	y_{11}, y_{12}
14	a_{41}	a_4	x_8	y_{11}, y_{12}
15		a_8	$\bar{x}_8$	y_{17}
16	a_5	a_5	$\bar{x}_1$	y_1, y_2
17		a_6	x_1	y_3, y_5
18	a_6	a_1	x_5	y_4, y_{14}, y_{15}
19		a_1	$\bar{x}_5$	y_6
20	a_7	a_{71}	$\bar{x}_8$	—
21		a_3	x_8	y_7, y_{16}
22	a_{71}	a_2	x_9	y_7, y_8
23		a_4	$\bar{x}_9$	y_8
24	a_8	a_7	x_9	y_9
25		a_4	$\bar{x}_9$	$y_{10}y_{12}$
26	a_9	a_{91}	x_5	—
27		a_3	$\bar{x}_5$	y_2, y_5, y_{13}
28	a_{91}	a_2	$\bar{x}_6$	y_1, y_2
29		a_9	x_6	y_5

Key: (1) Ordinal Number.

After obtaining a transfer table in which the condition $|X(a_i)| \leq 1$ is met for any a_i, $i = 1, \ldots, E$, if necessary the PLM can be replaced by a multiplexor and the diagram shown in Figure 1 reduced to the structure of a control automaton with programmable logic [4].

The structure of the automaton (see Figure 1 above) changes only slightly for synthesizing a Moore MPA. The PLM based on the signals $x_1, \ldots, x_L, \tau_1, \ldots, \tau_R$ forms the excitation functions of the memory elements $\varphi_1, \ldots, \varphi_R$. The ROM based on signals $\tau_1, \ldots, \tau_R$, formed at the outputs of the memory elements, produces the output signals $Y(a_{im})$. The principle of construction of the MPA on several ROM's and PLM's remains as before.

FOOTNOTES

1. S. I. Baranov and V. N. Sinev, "Avtamoty i Programmiruyemyye Matritsy" [Automata and Programmable Matrixes], Minsk, "Vysheyshaya shkola", 1980, 136 pages.

2. S. I. Baranov and V. N. Sinev, "Synthesizing Automata on Programmable Logical Matrixes," USIM, 1979, No 2, pp 58-64.

3. V. A. Sklyarov, "Minimizing the Number of Microoperations and Logical Conditions in a Microprogram," AVTOMATIKA I TELEMEKHANIKA 1980, No 9, pp 157-154.

4. A. Grasseli and U. Montanari, "On the Minimization of Read-Only Memories in Microprogrammed Digital Computers," IEE TRANSACTION ON COMPUTER, 1970, No 11, pp 1111-1114.

5. V. A. Sklyarov, "Method of Encoding Microcommands," OBMEN OPYTOM V RADIOPROMYSHLENOSTI, 1980, No 2, pp 15-18.

6. V. M. Kirpichnikov and V. A. Sklyarov, "Methods of Describing and Synthesizing One Class of Microprogram Automata," IZVESTIYA AN SSSR. TEKHNICHESKAYA KIBERNETIKA, 1979, No 1, pp 127-137.

7. S. I. Baranov and Kh. Killinberg, "Decomposition of Microprogram Automata," AVTOMATIKA I VYCHISLITEL'NAYA TEKHNIKA, 1978, No 6, pp 5-11.

11,176
CSO: 1863/139

UDC 681.325.5-181.4:621.372.54.037.372

MICROPROCESSOR MEANS OF CONSTRUCTING SYSTEMS FOR DIGITAL SIGNAL FILTRATION

Kiev UPRAVLYAYUSHCHIYE SISTEMY I MASHINY in Russian No 1, Jan-Feb 82 (manuscript received 12 Feb 81) pp 73-76

[Article by Oleg Viktorovich Glushko, engineer, and Leonid Mikhaylovich Osinskiy, candidate of technical sciences, Kiev Military Radio Engineering School]

[Text] The theory and practice of using digital signal processing systems have shown the need to develop digital filters with adjustable structure [1, 2]. The digital filter is a device to whose input a sequence of K-bit binary numbers x(nT), n = 0, 1, 2, ... arrives at discrete moments in time. An output sequence of K-bit binary numbers y(nT), n = 0, 1, 2, ... is formed at the output of the filter in conformity with the algorithm contained in it.

When devising a filter with adjustable structure it must be considered that there has to be a possibility of flexible change in its working algorithm, the bit format of numbers being processed, and certain other parameters related to hardware expenditures and speed. These possibilities can be realized on a microprocessor module which includes a microprocessor, ROM and internal memory LSIC's, and internal and external interfaces. The module is tuned by initial loading of appropriate data in its internal memory. All possible variations of filter work are reflected in the form of programs stored in the read-only memory.

Therefore, in the elementary case building a programmable digital filter amounts to developing the program of its work depending on its initial condition and the composition of the microprocessor module. But in many practical situations this obvious solution does not permit the required speed. Therefore there arise the problems of breaking the computing process and digital filters into parallel paths and selecting the structure of the multimicroprocessor system.

Let us consider the idea of solving these problems with application to the digital filtration algorithms described by a difference equation of the type [1]:

$$y(n) = \sum_{i=0}^{N-1} a_i x(n-i) - \sum_{j=1}^{N-1} b_j y(n-j), \qquad (1)$$

where a_i and b_j are the coefficients of the filter; x(n) is the input signal at moment in time nT; y(n) is the signal formed at the output of the filter after

arrival of signal x(n); n = 0, 1, 2, ... is the number of the discrete time count t = 0, T, 2T, ..., nT, ...; T is the period of signal discretization; (N-1) is the order of the filter.

Computing process (1) can be broken into parallel on the basis of the following conversion. We will introduce D^i, a statement to delay the signal for i periods of discretization. Then expression (1) can be represented in the form

$$y(n) = \sum_{i=0}^{N-1} a_i D^i [x(n)] - \sum_{j=1}^{N-1} b_j D^j [y(n)].$$

Considering that
$$D^i [V(n)] \pm D^{i+1} [W(n)] = D^i \{V(n) \pm D^1 [W(n)]\}, \quad (2)$$
it is possible to obtain the following equation

$$y(n) = \{a_0 x(n) - b_1 D^1 [y(n)]\} + D^1 \{a_1 x(n) - b_2 D^1 [y(n)]\} +$$
$$+ D^2 \{a_2 x(n) - b_3 D^1 [y(n)]\} + \ldots + D^i \{a_i x(n) - b_{i+1} D^1 [y(n)]\} +$$
$$+ \ldots + D^{N-1} \{a_{N-1} x(n) - b_N D^1 [y(n)]\}, \; b_N = 0.$$

Using property (2) again, we obtain the final expression

$$y(n) = \{a_0 x(n) - b_1 D^1 [y(n)]\} + D^1 \{a_1 x(n) - b_2 D^1 [y(n)] +$$
$$+ D^1 \{a_2 x(n) - b_3 D^1 [y(n)] + \ldots + D^1 \{a_{N-2} x(n) -$$
$$- b_{N-1} D^1 [y(n)] + D^1 \{a_{N-1} x(n) - b_N D^1 [y(n)]\} \ldots\}\}. \quad (3)$$

Expression (3) makes it possible to reproduce the required computation with the one-dimensional iterative network shown in Figure 1 below. If each block of this

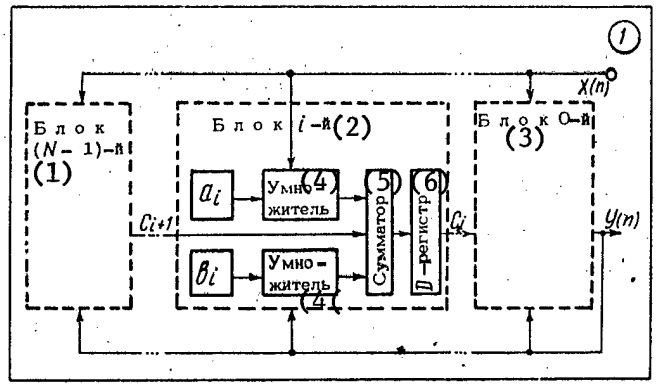

Figure 1.

Key: (1) Block (n-1); (4) Multiplier;
(2) Block i; (5) Adder;
(3) Block 0; (6) D-Register.

network is realized with a separate microprocessor module, we will obtain an adjustable N-microprocessor system of digital filtration with a high level of parallelism. The following operations must be performed in module i of this

system in the time interval from nT to (n=1)T:

- reception of input signal x(n) and output signal y(n-1);
- multiplication of x(n) by the filter coefficient a_i and multiplication of y(n-1) by the filter coefficient b_{i+1};
- reception of output data of module C_{i+1} (i+1) calculated in the time interval from (n-1)T to nT;
- formation of the algebraic sum
- entry of C_i in the output D-register of the module.

The speed of the N-microprocessor system is determined by time of performance of all these operations in one module. The module is tuned by entering the appropriate values of the coefficients a_1 and b_{i+1} in it.

When the system has m microprocessor modules (m < N), each module must realize computations corresponding to N/m blocks of the computation procedure shown in Figure 1 above. Let us suppose that N = lm. Then computation in conformity with expression (4) are carried out sequentially ℓ times in module j for i = $(j-1)\ell$, $(j-1)^{\ell}+1$, ..., $j\ell-1$. The value of c_i for i = $(j-1)\ell$ is transferred to module (j-1). In this case each module essentially realizes the working program of a digital filter of order (ℓ-1) (t = 1, 2, ... N).

Thus, the microprocessor module for digital filtration systems should reproduce the work of the (ℓ -1) order filter where the number ℓ and the coefficients of the filter are the variable quantities. The structure of the microprocessor module based on K580IK80 microprocessors [3] and the principle of construction of a digital filtration multiprocessor system and its use are shown in Figure 2 below. The module's read-only memory contains the working program of the (ℓ-1) order filter. The number ℓ and the filter coefficients are entered in internal memory with initial loading.

To expand the functional capabilities of the microprocessor module it is useful to execute this program in several variations which differ by a number of features, for example:

- by the bit configuration of the numbers being processed (8 or 16 bits);
- by type of filtration (filters with a finite impulse response or an infinite impulse response);
- by multiplication procedure (by program means, external multiplication units, or tabular multiplier).

All of these alternatives can be combined in a single program block with branching by appropriate signs. But to increase the speed of processing it is wise

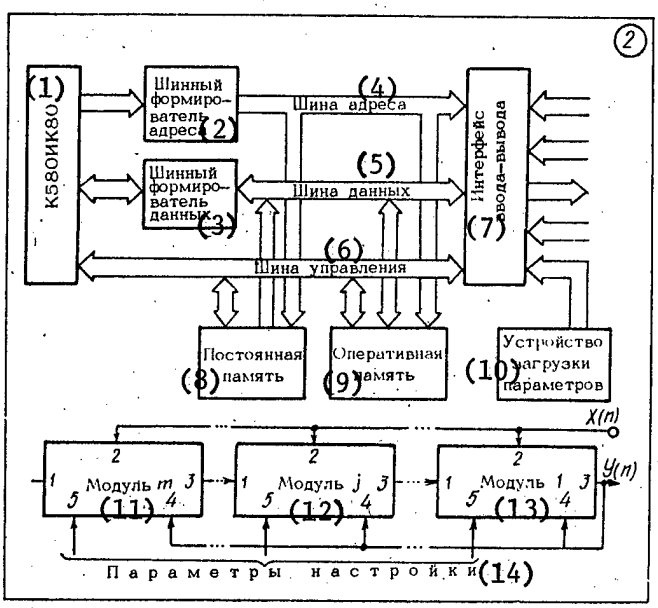

Figure 2.

Key:
(1) K580IK80;
(2) Address Line Shaper;
(3) Data Line Shaper;
(4) Address Line;
(5) Data Line;
(6) Control Line;
(7) Input-Ouput Interface;
(8) Read-Only Memory;
(9) Internal Memory;
(10) Parameter Loading Unit;
(11) Module m;
(12) Module j;
(13) Module ℓ;
(14) Adjustment (Tuning) Parameters.

---

not to use the module's time resources to test these signs. In practice, therefore, a separate program block must be created for each variation of digital filtration allocated by a certain set of signs and tuning for its execution must be done during the initial loading process.

Let us assess the speed of digital filtration in the system shown in Figure 2 above. All the modules in this system work concurrently. Therefore, the period T of discreteness of arrival of signals is determined by data processing time in one module. In turn, this time depends on the order of the filter realized in the module, that is, on the number ℓ and the program chosen for computation according to the set of signs indicated above. A concrete estimation of computation time was made with reference to the K580IK80 microprocessor for recursive and nonrecursive filtration algorithms for signals represented in eight-bit binary code.

An analysis of the programs showed that a large proportion of the time resources (about 80 percent) is used for realization of multiplication by program means. Therefore, it is proposed that the tabular method be used for multiplication in digital filtration microprocessor systems. Multiplication tables are placed

in internal memory and compiled according to given values of the filter coefficients in the period of initial tuning of the module. The results of estimation of time expenditures are represented in the table below in the form of the relationship between microcycles ($M \times 10^2$) and the value of the parameter ℓ.

(1) Количество вычислений	(2) Количество микротактов			
	(3) Рекурсивный алгоритм		(4) Алгоритм свертки	
	Программное умножение (5)	Табличное умножение (6)	Программное умножение (5)	Табличное умножение (6)
1	10,3	0,8	5,3	0,5
2	21,6	2,6	10,9	1,5
3	32,1	3,6	16,2	1,9
4	42,5	4,5	21,4	2,4
5	52,9	5,5	26,7	2,9
6	63,4	6,4	31,9	3,4
7	73,9	7,4	37,1	3,9
8	84,4	8,4	42,4	4,4

Table

Key: (1) Number of Computations; (4) Convolution Algorithm;
(2) Number of Microcycles; (5) Program Multiplication;
(3) Recursive Algorithm; (6) Tablar Multiplication.

The set of commands of the K580IK80 microprocessor is fairly standard. Therefore, assigning time expenditures in the form adopted in the table makes it possible to estimate the potential capabilities of other microprocessor means for setting up digital filtration systems or to formulate appropriate requirements for them.

Using the data in this table and knowing the length of execution of one microcycle in the chosen microprocessor, it is possible to determine the approximate value of period T (or frequency F = 1/T) of discretization of the signal for different values of ℓ. For example, the minimal length of the microcycle in the K580IK80 microprocessor is 0.5 microseconds. In the digital filtration microprocessor system based on it (see Figure 2 above), therefore, it is possible to process signals with a frequency of discretization of 25 kilohertz for the recursive algorithm and 40 kilohertz for the convolution algorithm. If a faster microprocessor, for example the K589IK02, is used [3], the frequency of discretization can be increased by an order.

The proposed microprocessor module (see Figure 2 above) can be used to set up a digital filtration system based on the cascade (stage) form of representation of the filter's work algorithm [1]. In this case the delay in formation of the output signal is increased, but the number of input-output channels in the module is reduced. Figure 3 below shows the structure of such a system. In this system signals are transferred sequentially from module to module. This makes it possible to introduce one input channel and one output channel in each module (not counting the input channel of the tuning parameters). As before, each module is tuned to realize the filter of order $(\ell-1)$ in conformity with the computation procedure shown in Figure 1 above.

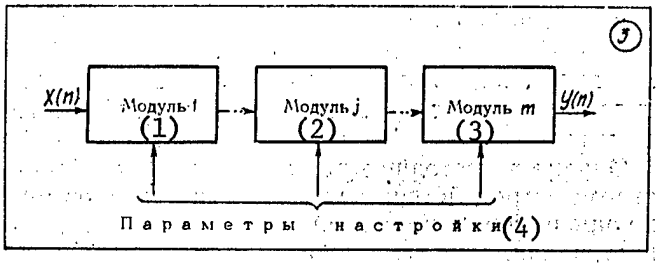

Figure 3.

Key: (1) Module ℓ;
(2) Module j;
(3) Module m;
(4) Tuning Parameters.

Cascade realization of the digital filtration algorithm by reducing the number of input-output operations in the module makes it possible to increase the frequency of arrival of signals. This gain (about 20 percent) is especially tangible when one computation block of the network shown in Figure 1 above is realized in each module. But the output reaction y(n) of the filter is formed with a delay of time mT relative to the moment of arrival of signal x(n). At the same time, in the system shown in Figure 2 above delay in forming the output signal does not exceed period of discretization T.

The digital filtration multimicroprocessor systems proposed above belong to the class of computing systems with parallel data processing that possess a single stream of commands and set of data flows [4]. In such systems the control unit (in the particular case the read-only command memory) can be removed from the modules and executed by a separate control module in one unit, which greatly simplifies the structure of the computation module itself. Sectioned (sliced) microprocessor means (for example series K589) are most suitable for realizing this approach to construction of a digital filtration system. An additional advantage of these means is that it is simple to build up the bit configuration of the data being processed. In microprocessor means realized on one type K580IK80 type chip, the bit configuration of numbers being processed is increased by program methods, and this naturally reduces speed.

Thus, contemporary microprocessor means make it possible to devise a general-purpose programmable module for digital filtration systems working in real time. The system based on this module has homogeneous structure and can be rearranged in conformity with the number of modules being used. An increase in the number of modules in the system with a simultaneous decrease in the order of the filter realized in one module increases the speed of the digital filter.

At the present time, however, the maximum estimates of this speed for series models of microprocessors are characterized by the frequency of discretization, which is measured in hundreds of kilohertz. Considering prospects for development of microprocessor means, we expect that frequencies of discretization of several dozen megahertz can be achieved in programmable digital filtration systems.

FOOTNOTES

1. L. Rabiner, and B. Gould, "Teoriya i Primeneniye Tsifrovoy Obrabotki Signalov" [Theory and Application of Digital Signal Processing], Moscow, "Mir", 1978, 848 pages.

2. R. D. Stinaff, "Microprocessor-Based Implementation for Programmable Digital Filters," PROCEEDINGS OF THE NATIONAL AEROSPACE AND ELECTRONIC CONFERENCE, Dayton, 1976, pp 463-470.

3. A. A. Vasenkov, V. A. Shakhnov, and B. M. Malashevich, "Microprocessor Integrated Circuits — the Foundation of Fourth-Generation Computers," MIKROELEKTRONIKA I POLUPROVODNIKOVYYE PRIBORY, 1978, Vyp 4, pp 3-17.

4. B. A. Golovkin, "Parallel Information Processing, Programming, Computing Methods, Computer Systems," IZVESTIYA AN SSSR. TEKHNICHSKAYA KIBERNETIKA, 1979, No 2, pp 116-151.

11,176
CSO: 1863/139

UDC 681.3.02

LINKING ELEKTRONIKA-T1000 VIDEO TERMINAL TO YeS COMPUTER

Kiev UPRAVLYAYUSHCHIYE SISTEMY I MASHINY in Russian No 1, Jan-Feb 82 (manuscript received 6 Aug 80, after revision 21 Oct 80) pp 119-121

[Article by Boris Vladimirovich Kardanovskiy, engineer, Pskov Electrical Machine Building Plant, and Viktor Aleksandrovich Kraynov, engineer, Leningrad]

[Excerpt] Statement of the Problem

Various types of video terminals which are included in the hardware of the Unified System [YeS] of Computers are now being used in remote data processing systems. Among these types of equipment are the YeS 7906 and YeS 7920 display complexes whose linkage with YeS computers has been standardized. But for organizing remote data processing systems with a few channels video terminals such as the Videoton-340 are also used. This unit is not included in the YeS hardware as a self-contained unit. It is small in dimensions, with a screen that has a large information capacity. The Videoton-340 can be used with a double-wire telephone line for remote control by computer over a significant distance [1].

The Elektronika-T1000 video terminal, which is similar to it, has similar characteristics. As a monitor it can use any remote receiver with remote control, which makes it possible to use this terminal as a group terminal when a Tauras-210 remote receiver is connected to it and as an individual receiver when the Yunost'-402 and Yunost'-R603 remote receivers are used. The Elektronika-T1000 is produced as an independent unit and is not included in the hardware of the Unified System. For this reason, it became necessary to develop the appropriate apparatus to organize communications between the Elektronika-T1000 video terminal and the YeS 1022 computer.

The control unit of the YeS 7077 typewriter was used as the linkage unit between the video terminal and the computer in order to speed up work and reduce expenditure of time and equipment for development. The YeS 7077 was modernized in a direction that insures use of the software of the base method of remote communications access (VTAM) by the hardware of the YeS 7906, in particular by the YeS 7566 unit that is included in this complex.

The following objectives were posed:

- refine the diagram of the unit so that it can execute commands issued by VTAM programs for the YeS 7906;

- solve the problem of converting DKOI codes issued from the computer into KOI-7 codes and vice versa because the code convertor available in the YeS 7077 does not perform this function;

- considering that the Elektronika-T1000 works with a two-wire communications line, issuing and receiving telegraph code in the form of a series of DC levels, provide for development of appropriate circuits that will make it possible to convert sequential code received from the video terminal into parallel code for transmission to the YeS 7077, and vice versa;

- insure coordination and synchronization of signals at the junction of the Elektronika-T1000 and the YeS7077.

The division of automated control systems at the Tskov Electrical Machine Building Plant has developed appropriate apparatus that insures this kind of linkage with the YeS 7906 in the VTAM regime using version 2.1 of the YeS disk operating system.

Designing the Linkage Circuit of the YeS 7077 and the Elektronika-T1000 Video Terminal

The fact that the video terminal has a bipolar interface predetermines the use of a signal converting unit, which made it possible to link a telegraph communications channel with the YeS 7077 by signal levels and exchange algorithms.

11,176
CSO: 1863/139

ELECTRON AND LITHOGRAPHY AS KEY PROBLEM IN DEVELOPMENT OF BASIC ELEMENTS OF COMPUTER EQUIPMENT

Kiev UPRAVLYAYUSHCHIYE SISTEMY I MASHINY in Russian No 1, Jan-Feb 82 pp 122-124

[Article by Vitaliy Pavlovich Derkach, doctor of technical sciences, Institute of Cybernetics of the Ukrainian SSR Academy of Sciences, Kiev, and Vladimir Moiseyevich Korsunskiy, candidate of technical sciences, Institute of Cybernetics of the Ukrainian SSR Academy of Sciences, Kiev]

> [Text] From the editors. In view of the importance of work on electron and ion lithography for development of the basic elements of computer equipment and control systems, the interdisciplinary nature of this new scientific-technical trend, the large role in it of research and development to devise appropriate control systems and software, and its close link with the problem of automating the design and manufacture of computers and systems, the editors plan to publish the principal materials from the 1st All-Union Coordination Meeting on Electron and Ion Lithography in one issue of the journal during 1982.

The 1st All-Union Coordination Meeting, devoted to consideration of the state of and prospects for development of scientific research on electron and ion lithography, was held at the Institute of Cybernetics of the Ukrainian SSR Academy of Sciences in Kiev on 24-26 June 1981. The meeting was conducted by the section of physical and physicochemical problems of microelectronics of the USSR Academy of Sciences' Scientific Council on Problems of Semiconductor Physics and Chemistry and the section on physicotechnological problems of cybernetics of the USSR Academy of Sciences' Scientific Council on Cybernetics. The meeting was attended by 96 scientists and engineers from 37 scientific research institutes, enterprises, and higher educational institutions in 12 cities of the country. Twenty-nine reports and communications were presented and discussed.

Development of the basic elements of computer equipment, which today are founded on LSIC's containing up to 10^4-10^5 elements, is largely determined by the capabilities of integrated technology. In recent years microcircuit technology has come fully up against the need to overcome the "micron barrier" in the dimensions of elements. Because of diffraction phenomena this "barrier," in principle, cannot be overcome using traditional photolithography processes which remain in the optical wavelength band. Therefore, the corresponding task has become one of the

key problems in further developments of the basic elements of computer equipment and microelectronics.

The way to overcoming the "micron barrier" is being provided by electron, ion, and X-ray lithography. High-resolution techniques of electron and ion optics are used in electron and ion lithography to form submicron structures. Reducing the dimensions of LSIC elements makes it possible to greatly increase their speed and degree of integration and reduce specific cost, in short, to improve all the techno-economic parameters of microcircuits and electronic equipment made from them. By the same token, this creates the prerequisite for the next qualitative leap forward in the development of computer equipment and its applications.

Almost all the industrially developed countries are working intensively on research and development in the field of electron and ion lithography. Soviet scientists and engineers have also made significant advances in this field. Research and development has been undertaken at academy and sectorial scientific research institutes, higher educational institutions, and leading industrial enterprises to develop the theoretical foundations of electron lithography, devise high-quality electron resists, determine the optimal conditions for their exposure and development, develop more sophisticated electron lithography equipment, means of automated program control of exposure, and so on.

In the field of the theoretical foundations of electron lithography attention has been concentrated on studying the physics of the passage of bundles of accelerated electrons through heterogeneous layered targets typical of electron lithography and to construction of mathematical models of this process. Both a rigorous analytical approach based on the use of cybernetic equations of electron transfer and the methods of direct statistical modeling of this random process by computer (the Monte Carlo methods) are being developed. Several simplified models have been proposed which cover the patterns of the real process only in the most general outline.

Unfortunately, little work has been done so far on investigation and theoretical description of the chain of processes leading from primary ionizations and excitations that occur in the resist when exposed to accelerated electrons to the final products of radiation-chemical conversion — destruction or transverse stitching of polymer molecules. In addition, there has not been enough theoretical work on the questions of developing images exposed by an electron beam. But it is precisely these processes that determine the efficiency of transforming the energy of the electron beam into the required radiation-chemical conversions that is, the sensitivity and resolution of the electron resists.

Even though scientists in our country have developed a number of positive and negative electron resists whose fundamental parameters correspond to the world level, further investigation and development of higher quality resists could be greatly accelerated by clarifying these theoretical issues.

The relationship between the sensitivity and resolution of existing electron resists and the energy of the electrons, the thickness of the resist layer, the material of the substrate, the composition of the developer, and the conditions of development are being intensively studied, as is the stability of the electron

resist masks being formulated, the various types of etching, including ion-plasma etching. This research, however, needs better coordination and should also cover questions of change in the characteristics of electron resists during the process of storage under different conditions.

A great deal has also been done in our country to develop good electron lithography equipment. Promising research has been carried out to devise highly efficient electron sources and to calculate and study the corresponding electron optical systems. A number of experimental models of single-poled, multi-beamed, and projection devices for electron beam exposure have been developed and tested. Unfortunately, however, series production of domestic electron lithography equipment still has not been set up.

A similar situation is observed in the area of means of automatic program control of electron beam exposure, even though good experimental models of the corresponding control systems were developed in our country as early as the first half of the 1970's and have been steadily improved since that time.

In recent years research has intensified concerning the different image distortions characteristic of electron lithography, in particular those related to dispersion of electrons in the resist and substrate. These distortions have received the name "proximity effects." Other distortions under intensive study are those caused by aberrations in electron optical systems, errors in electronic image generators, and the etching processes. With good electron resists and electron beam devices it has been found that the productivity of electron lithography systems depends significantly on the set of elementary figures from which the topological elements of microcircuits are synthesized, on the speed of generation of these elements, and on the technique for breaking topological designs into elementary figures.

All these considerations posed the challenge of developing special software to automate the very labor-intensive, complex, but essential work of correcting the topological drawings of LSIC's that contain hundreds of thousands of elements with due regard for technological and electron optical distortions of the designs and breaking them down (by the optimal or close to optimal technique) into separate working zones, subfields, and elementary figures, and preparing information for exposure in the language and formats envisioned by the appropriate control system. The corresponding software must be coordinated with the automatic LSIC design systems already existing in our country. To save time and expenditures it is important in the very early stages to work out standardized methods of data representation, to insure the possibility of exchanging appropriate programs, and to coordinate this work well.

The research and development done in our country in the field of electron lithography has already enabled us to successfully apply its results in the production of precision photo and X-ray templates, to shape semiconductor microstructures with submicron element dimensions, and to fabricate new types of microcircuits using TsMD [expansion unknown], PZS [expansion unknown], surface-active substances, and the Josephson effect.

The extent of research in the field of ion lithography is comparatively small at the present time. Nonetheless, exploratory studies that have been done in ion

lithography for clearance and using sharply focused beams have already demonstrated quite convincingly the promise of this technique. It provides even higher resolution and produces much less image distortion than electron lithography thanks to the much shorter path of the dispersed ion. It is true that the ion beams are much harder to control than electron beams, but it appears that these difficulties are fully compensated for by the technological advantages of ion beams.

All of these issues as well as the prospects for development of scientific research in electron and ion lithography were discussed at the coordination conference.

Having pointed out existing achievements and shortcomings, the conference singled out the following pressing problems in the field of electron lithography.

1. Clarification of the patterns of passage of electron beams through multilayered targets of complex composition; clarification of the mechanisms and factors on which the effectiveness of local physicochemical conversions in electron resists occur under the effect of exposure; identification of the patterns of development and formation of the profile of the topological elements being created in the resist; modeling the corresponding processes and on this basis searching for optimal conditions of exposure and development and new ways to synthesize highly efficient electron resists.

2. Quantitative theoretical and experimental study of "proximity effects" caused by different processes of delocalization of the radiation action of the electron beam; searching for effective methods of reducing and correcting these effects.

3. Development of highly productive domestic electron lithographic equipment using new sources of electrons with heightened brightness, high-quality electron optical systems, fast and high-precision coordinate tables, and highly productive control systems containing high-speed digital-analog convertors with enlarged bit configuration, more effective methods of generating designs, and large internal memory capacity; improving the stability and operating reliability of this equipment.

4. Development of systems for automated design of complex submicron LSIC structures and of systems of special software to prepare programs for electron beam exposure of such structures with correction of the "proximity effects" and other technological and hardware distortions of topology; development of standardized methods of data representation in such systems to insure the possibility of program exchange.

5. Synthesis of electron resists with heightened sensitivity, resolution, and contrast and synthesis of reagents to prepare and develop them; improvement of technical specifications for resists and corresponding reagents; study and publication of adequately complete summary reference data on the characteristics of resists and changes in them under varying conditions of storage, preparation, and development; development and incorporation of "dry" resist processing techniques.

6. Development of designs and methods of fabricating precision templates and of corresponding equipment for projection electron, ion, and X-ray lithography which transfers the entire topological design at one time.

7. Improvement of marker characters for electron lithography and methods of fabricating them, techniques of identifying signals from them, and techniques of rapid precision automatic integration of neighboring working fields and successive topological layers during electron beam exposure.

8. Development of automated methods of monitoring topological structures with submicron element dimensions created using electron, ion, and X-ray lithography.

9. Development of research toward comprehensive study of the interrelationship of all elements of electron lithography and their relationship with the entire technological process of fabricating LSIC's.

In the area of ion lithography the coordination conference considered it essential to concentrate attention on clarification of its potential in principle and the characteristics of practical use, development of highly productive ion lithography equipment which transfers the entire image at once and efficient probe-type ion optimal systems with the capability of design generation, clarification of the patterns of interaction of ion beams with resists and the substrate, and development of techniques for integrating working fields and topological layers during ion lithography.

The coordination conference recommended formulation of a special comprehensive program of research and development on electron lithography for institutions of the USSR Academy of Sciences and higher educational institutions in cooperation with the sectorial scientific research institutes and leading enterprises.

The need to organize broader training of specialists in electron and ion lithography was also noted. This applies above all to the higher educational institutions of Moscow, Leningrad, and Kiev, where there are highly qualified scientists in this area. The conference indicated further that we must bolster cooperation in this subject area with the socialist countries of the Council for Economic Mutual Assistance.

Rapid development of electron and ion lithography, which offers fundamentally new opportunities in technology and putting the results obtained in this area into practice as quickly as possible demands harmonious, coordinated work by production specialists, physicists, specialists in electron optics and computer technology, organic chemists, and programmer-mathemeticians. We must also have assistance from the State Committee for Science and Technology, interested ministries and departments, and the scientific community.

11,176
CSO: 1863/139

UDC 621.382.82

PRINCIPLES OF ORGANIZATION AND APPLICATION OF HIGH SPEED CASCADED CARRY MICROCIRCUITS

Kiev UPRAVLYAYUSHCHIYE SISTEMY I MASHINY in Russian No 2, Mar-Apr 82 (manuscript received 22 Apr 81, after revision 14 Jul 81) pp 16-21

[Article by Vladimir Borisovich Smolov, doctor of engineering science; Sergey Timofeyevich Khvoshch, candidate of engineering science; and Nikolay Gavrilovich Kuz'menko, junior scientific associate; all from LETI [Leningrad Institute of Electrical Engineering], Leningrad]

[Text] Thanks to high speed, broad logic capabilities and small unit power consumption, microcircuits for arithmetic and logic units (ALU) and LSI microprocessors with a sectioned structure are now widespread [1-5]. Using these microcircuits permits building micro and minicomputer processors of arbitrary word length by combining several IC's in shift and carry propagation circuits [2, 5]. However, such a build up leads to an increase in time for executing arithmetic operations in the ALU, which is due to the necessity of serial propagation of carry signals between the individual microcircuits combined in the processor. Shown in fig. 1a is the organization of a processor with word length (n x m) when n m-digit ALU's are used. Conversion of information occurs on operands $A=\{A_0, A_1, \ldots, A_{n-1}\}$ and $B=\{B_0, B_1, \ldots, B_{n-1}\}$, coming into inputs A and B of the ALUs. The result $S=\{S_0, S_1, \ldots, S_{n-1}\}$ and the output carry signal (C_{out}) are generated based on the control code Y, which adjusts the ALU's for the specific functional conversions and values of the input carry (C_{in}).

Assuming the time diagram of ALU operation can be represented by fig. 1b, in the first approximation, processor cycle time can be computed as

$$T_{cycle} \geqslant \max\{n \cdot t_C^C,\ (n-1) \cdot t_C^C + t_C^S\}, \qquad (1)$$

where $n \cdot t_C^C$ is the total delay in the carry propagation path, and $(n-1) \cdot t_C^C + t_C^S$ is the total delay in the result formation path in the last ALU based on the delay of the carry signal at its input.

In fig. 1b, signal propagation delay time is as follows: t_Y^C is from the inputs of control code Y to the output C_{out}, t_{AB}^C is from information inputs A and B to output C_{out}, t_C^C is from the carry input

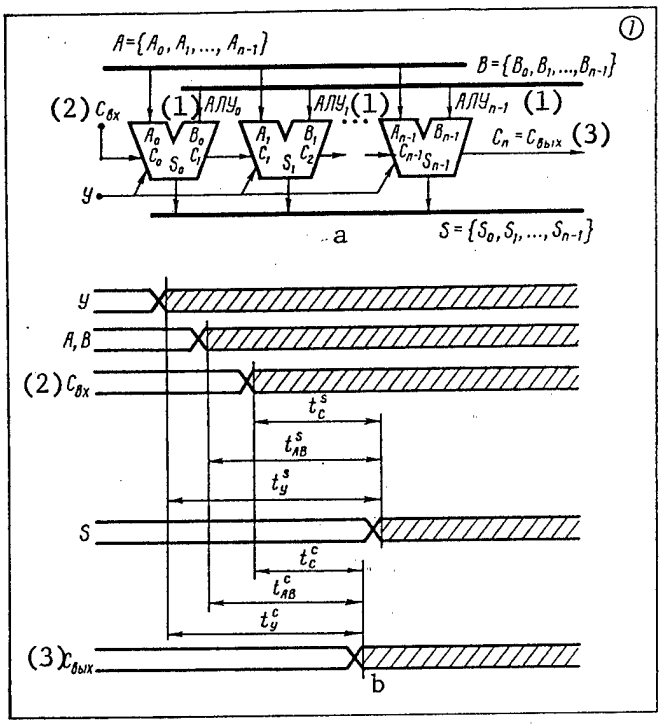

Fig. 1. Processor with word length n X m:
a--structural diagram,
b--time diagram of operation

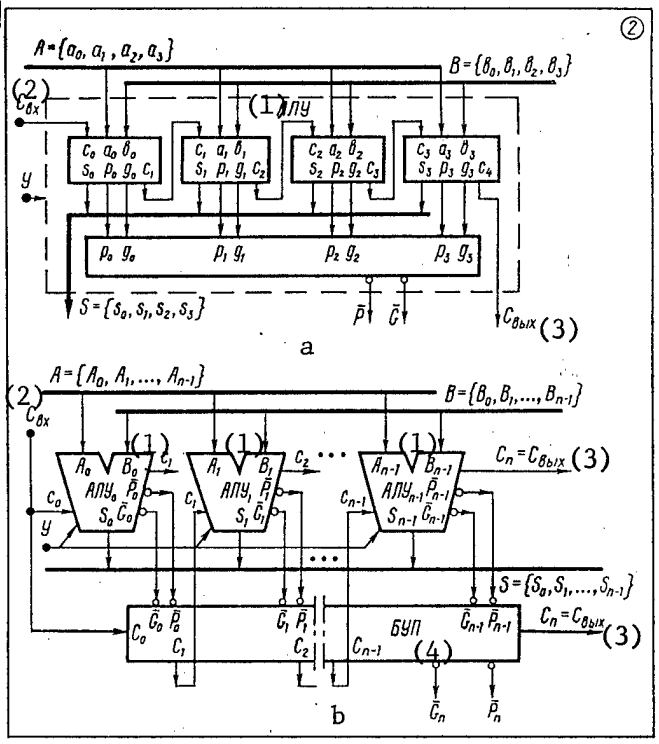

Fig. 2. Structural diagram:
a--four-digit ALU,
b--processor built on basis of n m-digit ALU's and hypothetical fast carry circuit

Key:
1. ALU's 0, 1, n-1
2. C_{in}
3. C_{out}
4. BUP [high-speed cascaded carry unit]

C_{in} to the output C_{out}, t_Y^S is from the inputs of control code Y to output S, t_{AB}^S is from information inputs A and B to output S, and t_C^S is from the carry input C_{in} to the output S.

To reduce processor cycle time, high-speed cascaded carry microcircuit units (BUP) have been incorporated in microprocessor sets of sectional LSI circuits (MPK) and in MSI IC series [2, 3, 6].

Principles of Organization of High-Speed Cascaded Carry Units. To explain the principle of organization of carry propagation signals in a processor, let us consider the hypothetical structure of the four-digit ALU shown in fig. 2a. The ALU contains four digit sections that perform conversions on the operands $A=\{a_0, a_1, a_2, a_3\}$ and $B=\{b_0, b_1, b_2, b_3\}$ based on the value of the input carry $C_{in}=c_0$ and which form the

operation result $S=\{s_0, s_1, s_2, s_3\}$ and the output carry $C_{out}= c_4$. In executing the operation of addition ($S=A+B+C_{in}$), the ALU operates by this rule:

$$s_i = a_i \oplus b_i \oplus c_i,\ c_{i+1} = a_i b_i + a_i c_i + b_i c_i,\ i = 0, 1, 2, 3. \quad (2)$$

To connect fast carry units to the ALU, it is necessary to provide within the LSI circuit for a special unit that generates the generation signal $\overline{G}$ and propagation $\overline{P}$ of the fast carry between the ALU units. For the addition operation, the signals $\overline{P}$ and $\overline{G}$ must be generated by the rule:

$$\left.\begin{aligned} P &= \overline{p_3 p_2 p_1 p_0}, \\ \overline{G} &= \overline{g_3 + g_2 p_3 + g_1 p_3 p_2 + g_0 p_3 p_2 p_1}, \\ p_i &= a_i \oplus b_i,\ g_i = a_i b_i,\ i = 0, 1, 2, 3. \end{aligned}\right\} \quad (3)$$

The output carry signal from the ALU ($C_{out}= c_4$) and the carry signals within the ALU (c_1, c_2, c_3) can be generated both sequentially (by the system of expressions (2)) and by using a circuit that generates a fast (ripple through) carry within the ALU (according to system of expressions (3)). In the process, the output carry signal must be generated by the rule:

$$C_{out}= G + p_3p_2p_1p_0C_{in} = c_4 = g_3+g_3p_3+g_1p_3p_2+g_0p_3p_2p_2p_1+p_3p_2p_1p_0c_0. \quad (4)$$

Expressions (2)-(4) are valid in using systems both with positive (true) logic (level of logical one is the high potential) and with negative (inverse) logic (level of logical one is the low potential). In the case of organization of subtraction operations in the ALU, it is sufficient to convert the subtrahend operand to two's complement code (when $S=A-B^*+C_{in}$, $B^*=\overline{B}+1$; when $S=B-A^*+C_{in}$, $A^*=\overline{A}+1$).

Shown in fig. 2b is the structure of an (n X m)-bit processor built on the basis of n m-digit ALU's and a hypothetical fast carry circuit; the time diagram for operation of this circuit is shown in fig. 3, where signal propagation delay times are designated as follows:

t^{PG}_{ABC} is from carry input C_{in} to ALU outputs $\overline{P}$ and $\overline{G}$,

t^{PG}_{CPG} is from fast carry inputs $\overline{P}_i$ and $\overline{G}_i$ to BUP [fast carry] outputs $\overline{P}$ and $\overline{G}$, and

t^{C}_{CPG} is from fast carry inputs $\overline{P}_i$ and $\overline{G}_i$ to BUP [fast carry] outputs C_i.

The fast carry unit receives the carry input signal ($C_o=C_{in}$) and the signals for generation and propagation of the fast carry in the ALU ($\overline{P}_i$ and $\overline{G}_i$) which come into the BUP [fast carry unit] inputs with the same designation. Since the signals $\overline{P}_i$ and $\overline{G}_i$, formed from expression (3), at the ALU inputs do not depend on the value of

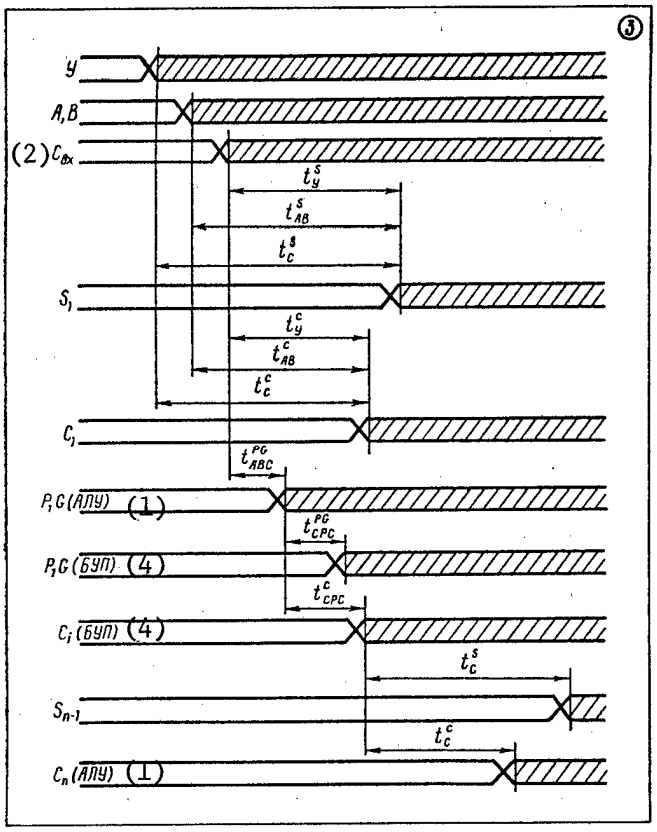

Fig. 3. Time diagram of operation of processor with word length n X m and hypothetical fast carry circuit

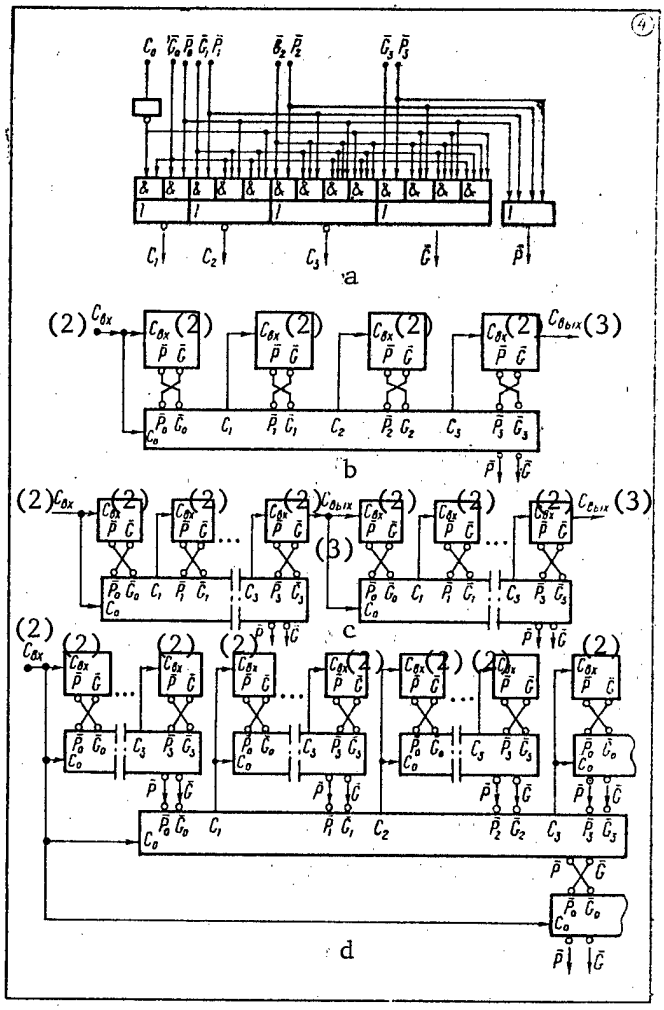

Fig. 4. Structural diagram:
a--K133IP4, K155IP4 and K134IP4 fast carry units,
b--16-bit processor with simultaneous carry,
c--32-bit processor with sequential carry between 16-bit units
d--multibit processor with three levels of intergrouped cascading of carries

Key:
1. ALU
2. C_{in}
3. C_{out}
4. BUP [high-speed cascaded carry unit]

the input carries for the sections, the total time for carry signal propagation to the input of the last ALU is

$$t_{C_{n-1}} = t^{PG}_{ABC} + t^{C}_{CPG}, \tag{5}$$

the delay in output of the operation result in the last LSI circuit is

$$t_{S_{n-1}} = t_{C_{n-1}} + t^{S}_{C}, \tag{6}$$

and at the output from the BUP [fast carry unit] or last ALU, the value of the output carry can be obtained with the following delays:

$$t^{1}_{C_n} = t^{PG}_{ABC} + t^{C}_{CPG}$$

or (7)

$$t^{2}_{C_n} = t^{1}_{C_n} + t^{C}_{C}.$$

In the process, in the BUP [fast carry unit], the grouped high-speed carry signals (C_i) and the signals for propagation and generation of the cascaded high-speed carry ($\overline{P}$ and $\overline{G}$) are formed by the rule:

$$\begin{aligned}
&C_1 = G_0 + P_0 C_0,\\
&C_2 = G_1 + P_1 C_1,\\
&\cdots\cdots\cdots\\
&C_n = G_{n-1} + P_{n-1} C_{n-1},\\
&\overline{G}_n = \overline{G}_{n-1}\overline{P}_{n-1} + \overline{G}_{n-2}\overline{G}_{n-1}\overline{P}_{n-2} + \ldots +\\
&+ \overline{G}_1\overline{G}_2 \ldots G_{n-1}P_1 + \overline{G}_0\overline{G}_1 \ldots \overline{G}_{n-1},\\
&\overline{P}_n = \overline{P}_0 + \overline{P}_1 + \overline{P}_2 + \overline{P}_3 + \ldots + \overline{P}_{n-1}.
\end{aligned} \tag{8}$$

Since the delay in forming the signals S in the ALU is always greater than the delay in froming the carry (C_{out}), cycle time for the processor with a BUP [fast carry unit] in the first approximation is defined as:

$$T^{*}_{cycle} \geq t^{PG}_{ABC} + t^{C}_{CPG} + t^{S}_{C}. \tag{9}$$

The gain in speed for the processor with the fast carry unit relative to serial connection of the ALU's can be computed from the expression

$$\Delta T^{abc}_{cycle} = (n-1)\cdot t^{C}_{C} - t^{PG}_{ABC} - t^{C}_{CPG}. \tag{10}$$

Use of the K155IP4, K133IP4 and K134IP4 Fast Carry Units. The K155IP4 and K133IP4 fast carry units are designed to be used with the K133IP3 and K155IP3 ALU's and as part of the K583 and K584 LSI microprocessor sets with the K583IK1 and K584VM1 microprocessors. The functional diagram of the fast carry unit is shown in fig. 4a. One microcircuit is designed to connect to four sectional LSI circuits and is placed in a 16-lead package. The fast carry unit generates three fast carry

Table 1. Rule for generation of signals output from K133IP4, K155IP4 and K134IP4 BUP's [fast carry units]

Inputs									Outputs				
C_0	$\overline{G}_0$	$\overline{P}_0$	$\overline{G}_1$	$\overline{P}_1$	$\overline{G}_2$	$\overline{P}_2$	$\overline{G}_3$	$\overline{P}_3$	C_1	C_2	C_3	$\overline{G}$	$\overline{P}$
X	B	B							H				
H	B	X							H				
X	H	X							B				
B	X	H							B				
X	X	X	B	B						H			
X	B	B	B	X						H			
H	B	X	B	X						H			
X	B	X	B	X						H			
X	H	X	H	H						B			
B	X	H	X	H						B			
X	X	X	X	B	B	B					H		
X	X	B	B	X	B	X					H		
X	B	B	B	X	B	X					H		
H	B	X	B	X	B	X					H		
X	X	X	H	H	H	H					B		
X	X	X	H	H	H	H					B		
B	X	H	X	H	X	H					B		
	X		X	B	B	B	B	B				B	
	X		X	B	B	B	B	X				B	
	X		B	X	B	B	B	X				B	
	B		B	X	B	B	H	X				H	
	X		X	X	H	H	H	X				H	
	X		X	X	H	H	H	X				H	
	H		H	H	X	X	X	X				H	
		B		X				B					B
		X		B				X					B
		X		X				X					B
		X		B				X					B
		H		H				H					H

B — high potential H — low potential X — arbitrary value

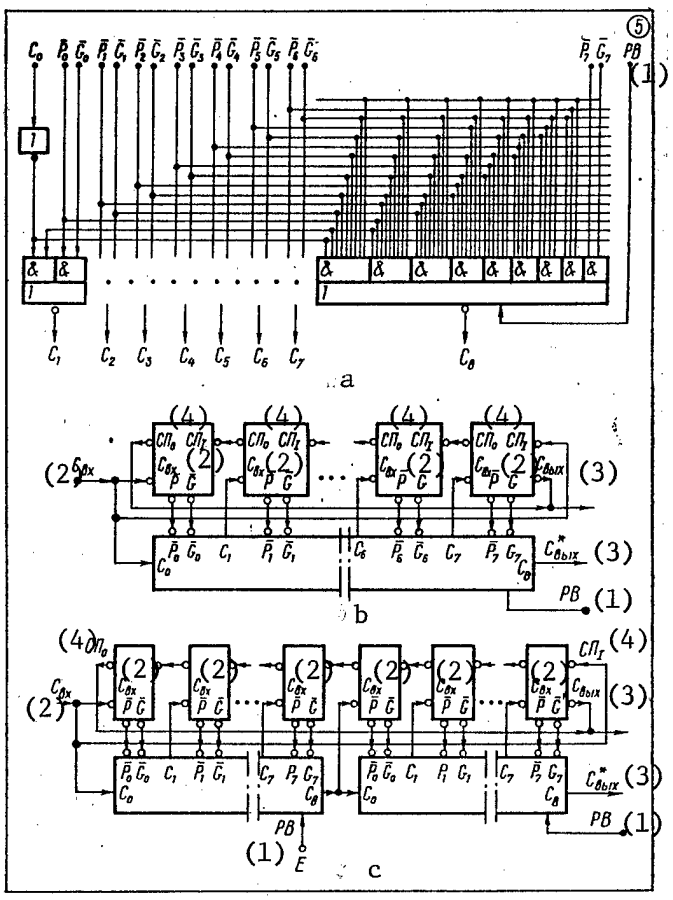

Fig. 5. Structural diagram:
a--K589IK03 fast carry unit
b--connection of eight K589IK02 LSI circuits to a K589IK03 fast carry unit
c--32-bit processor with serial-parallel carry between the 16-bit units

Key: 1. RV [output enable]; 2. C_{in}; 3. C_{out}; 4. SP [expansion unk]

signals (C_1, C_2 and C_3) and the cascade signals for generation and propagation of the grouped fast carry signals ($\overline{P}$ and $\overline{G}$). Table 1 shows the rule for generating values of output signals as a function of input variables. When microcircuits with active high potential of the input carry C_0 (K155IP3, K133IP3 and K584VM1) are connected to the fast carry unit, cross switching of outputs and inputs $\overline{P}$ and $\overline{G}$ should be effected at all odd levels of cascading, and straight at all even levels of cascading.

Fig. 4b illustrates organization of a 16-bit processor by using the K155IP3, K133IP3 or K584VM1 microcircuits. Table 2 gives the values of the main electrical and time parameters of the K155IP4 and K133IP4 fast carry units. When these

Table 2. Parameters of K133IP4 and K155IP4 fast carry units

Parameter	Parameter value		
	minimum	rated	maximum
Supply voltage (E_s), V	4.5	5.0	5.5
Consumption current (I_c), mA	59	65	72
Input current of logical zero for inputs (I^0_{in}), mA:			
C_0	--	--	3.2
$\overline{P}_i$, $\overline{G}_i$	4.8	--	16
Maximum time of delay, ns:			
when switched on (t^C_{CPG})	--	--	22
when switched off (t^{PG}_{CPG})	--	--	17

Measurement conditions: $T_{ambient}$ = -10 to +70°C, R_{load} = 200 Ohms, C_{load} = 30 pF, V^0_{in} = 0.8V, V^1_{in} = 2.0V, $V^0_{out} \leq 0.4$ V, $V^1_{out} \geq 2.4$ V

microcircuits are used, special attention should be paid to the load carrying capacities of the outputs connected to the inputs $\overline{P}_i$ and $\overline{G}_i$ which have an input current of logical zero to 16 mA ($\overline{G}_1$).

Using the fast carry unit with the ALU LSI circuit permits building 16-bit processors with about 50-ns arithmetic operation execution time, and 1 microsecond with the K584VM1 LSI circuit. Fig. 4c shows the structure of a 32-bit processor with sequential carry between the 16-bit cascades, and shown in fig. 4d is the principle of designing multibit processors with three levels of intergroup cascading of carries. It is characteristic that using intergroup cascading of carries yields a large gain in time in organizing processors with a large word length. Thus, for example, when sequential uarry is used between individual circuits in a 64-bit processor, organized on the basis of K133 or K155 series ALU's, operation execution time is about 200 ns, and about 4 microseconds when K584 series microprocessors are used. When two-level cascading is used, these times are 60 ns and 1 microsecond, respectively.

Cycle time of processors with an arbitrary word length when several levels of cascading (k) of BUP [fast carry units] are used can be computed as:

$$T_{cycle} \geq T^{MP}_{cycle} + \left\lceil \frac{n}{4^k} \right\rceil \cdot t^C_{CPG}, \qquad (11)$$

where T^{MP}_{cycle} is the cycle time of the microprocessor section, and n is the number of sections.

Table 3. Parameters of K134IP4 fast carry unit

Parameter	Parameter value minimum	rated	maximum
Supply voltage (E_s), V	4.5	5.0	5.5
Consumption current (I_c), mA	5	5.5	6
Input current of logical zero for inputs (I^0_{in}), mA:			
C_o	0.18	---	---
$\overline{P}_i$ $\overline{G}_i$	0.36	---	1.44
Maximum time of delay, ns:			
when switched on (t^C_{GPG})	---	---	200
when switched off (t^{PG}_{CPG})	---	---	150

Measurement conditions are same as in table 2 except for ambient temperature which in this case varied from -60°C to +70°C

Table 4. Parameters of K589IK03 fast carry unit

Parameter	Parameter value minimum	rated	maximum
Supply voltage (E_s), V	4.75	5.0	5.25
Consumption current (I_c), mA	---	95	130
Input current of logical zero (I^0_{in}) by inputs, mA:			
C_0, RP, $\overline{P}_6$, $\overline{P}_7$	---	---	- 0.25
$\overline{P}_0$ - $\overline{P}_5$, $\overline{G}_7$	---	---	- 0.5
$\overline{G}_0$ - $\overline{G}_6$	---	---	- 1.5
Dynamic parameters, ns: t^C_{PG}	---	10	---
t^C_C	---	18	---
t^C_{RP}	---	20	---

Measurement conditions are the same as in table 2

When one level of cascading of the fast carry unit is used (with sequential carries between cascades)

$$T_{cycle} \geq T^{MP}_{cycle} + \left]\frac{n}{4}\right[\cdot t^C_{CPG} + \left(\left]\frac{n}{4}\right[- 1\right) \cdot t^C_G, \quad (12)$$

where][indicates rounding to the next larger integer.

The K134IP4 fast carry unit has an architecture similar to that of the K133IP4 and K155IP4 IC's, but is made with the technology of low consuming TTL circuits. Its basic electrical and time parameters are given in table 3. Using this IC permits connecting to it microprocessors with smaller load capacity of outputs (the K582IK1, for example); in doing so, there is a power savings, but the time parameters of the processors are substantially degraded. It is also worthwhile to use the K134IP4 fast carry unit with the K584VM1 LSI circuit operating under the conditions of lowered injection current, which permits reducing power consumption of processors to the level of CMOS circuits with a proportional reduction in their performance.

Use of K589IK03 Fast Carry Units. The K589IK03 fast carry unit is designed for use with the K589IK02 microprocessor. Unit structure is shown in fig. 5a. The microcircuit is placed in a 28-lead package and provides for connecting eight LSI microprocessors. The availability of output C_8 provides the capability of sequential grouped carry between 16-bit processors. Values of the main electrical and time parameters for this fast carry unit are given in table 4. The availability in this unit of the RV (output enable) lead provides the capability of gating carry signal output at output C_8.

Shown in fig. 5b is scheme for connecting eight LSI K589IK02 circuits to one K589IK03 fast carry unit. Since the input carry to the K589IK02 LSI circuits is coded by active low level of voltage, a straight connection of microprocessor outputs ($\overline{P}$ and $\overline{G}$) and fast carry unit inputs ($\overline{P}_i$ and $\overline{G}_i$) is made. Operation execution time in a 16-bit processor with the fast carry unit is about 118 ns. Shown in fig. 5c is the structure of a 32-bit processor with sequential carry between the 16-bit units, which has a cycle time for executing microinstructions of about 136 ns.

The cycle time (ns) of processors with an arbitrary word size when the K589IK03 fast carry unit is used can be computed as:

$$T_{cycle} \geq T^{MP}_{cycle} + \left\lceil \frac{n}{8} \right\rceil \cdot t^{C}_{CPG} = 100 + \left\lceil \frac{n}{8} \right\rceil \cdot 18. \quad (13)$$

Conclusion. The examples given in this article far from exhaust the possibilities of using fast carry units in modern digital machine circuits. As an example of nontraditional use of a fast carry unit, the microcomputer interrupt system can be cited. Shown in fig. 6a is the structure of a 128-input circuit for processing interrupt requests, built on the basis of K589IK14 interrupt priority units (BPP), which has eight inputs for interrupt requests. Specification of intergroup priority between the interrupt priority units is made by sequential connection of inputs for enabling groups of interruption (RGP) of high-order LSI circuits with the inputs for enabling groups (RG) of low-order ones. In doing so, delay through the RG-RGP circuit for an LSI circuit is about 13 ns. For reliable system operation, cycle time (ns) must be selected from the following condition:

$$T_{cycle} \geq T^{BPP}_{cycle_{min}} + (n-1) \cdot t^{RGP}_{rg} = 80 + (n-1) \cdot 13, \quad (14)$$

where T^{BPP}_{cycle} is the interrupt priority unit cycle time, t^{RGP}_{rg} is the time for propagating the signal through the RG-RGP circuit, and n is the number of BPP units.

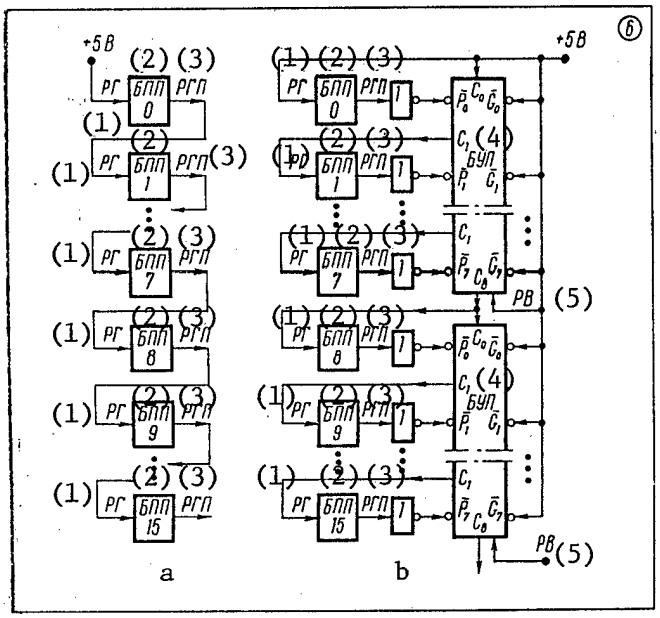

Fig. 6. Structural diagram of 128-input circuit for processing interruption requests:
a--without fast carry circuit,
b--with fast carry circuit

Key:
1. RG [group enable]
2. BPP [interruption priority unit]
3. RGP [interruption group enable]
4. BUP [fast carry unit]
5. RV [output enable]

The cycle time for the system shown [fig. 6a] is about 275 ns.

Shown in fig. 6a is the structure of this same system with two K589IK03 fast carry units for matching the signals RGP [interruption group enable] and $\overline{P}_i$, in which inverters are used. Cycle time of the system with fast carry units is

$$T^{*}_{cycle} \geqq T^{BPP}_{cycle_{min}} +] \frac{n}{8} [\cdot (t^{C}_{CPG} + t_{INV}). \qquad (15)$$

In the process, t_{INV} = 5 ns, n = 16 and t^{C}_{CPG} = 18 ns, $T^{*}_{cycle} \geqq$ 126 ns.

Expressions (11)-(15) can be used in practice in computing delays in systems built on the basis of fast carry units.

BIBLIOGRAPHY

1. Sharugin, I. I., "Tranzistorno-tranzistornyye logicheskiye skhemy" [Transistor-Transistor Logic Circuits], Moscow, Sov. radio, 1974, 124 pages.

2. Belous, A. I.; Boldyrev, V. P.; Marchenko, G. M.; and Savotin, Yu. I., "The K155IP3 and K155IP4 Semiconductor Integrated Circuits," ELEKTRONNAYA PROMYSHLENNOST', No 10, 1975, pp 60-64.

3. Berezenko, A. I.; Koryagin, L. N.; Nazaryan, A. R.; and Orlov, B. V., "The K589 Series LSI TTL Microprocessor Set with Shottky Diodes," ELEKTRONNAYA PROMYSHLENNOST', No 5, 1978, pp 20-21.

4. Khvoshch, S. T. and Klyashtornyy, M. Yu., "Architecture of K584 Series LSI Microprocessor Set," in "Kompleksnaya miniatyurizatsiya radioelektronnoy apparatury" [Complex Miniaturization of Radioelectronic Apparatus], Leningrad, 1980, pp 46-50.

5. Belous, A. I.; Klyashtornyy, M. Yu.; and Khvoshch, S. T., "Opyt razrabotki spetsializirovannykh mikroEVM na osnove mikroprotsessora K584MK1" [Experience in Developing Specialized Microcomputers Based on the K584MK1 Microprocessor], Leningrad, LDNTP [Leningrad House of Scientific and Technical Propaganda], 1980, 24 pages.

6. Belous, A. I.; Gorovoy, V. V.; Krasnitskiy, B. M. et al., "Micropower LSI Microprocessor Complex Based on Circuits with Injection Supply," ELEKTRONNAYA PROMYSHLENNOST', No 4, 1981, pp 26-29.

8545
CSO: 1863/179

HARDWARE DESIGN TO SUPPORT MICROPROGRAM DEVELOPMENT FOR BIT-SLICE PROCESSORS

East Berlin NACHRICHTENTECHNIK ELEKTRONIK in German
Vol 31, No 7, Jul 81 pp 294-297

[Article by R. Sandau, Berlin, Report from the Institute of Electronics of the AdW (Academy of Sciences) of the GDR]

[Excerpt] Descriptors: microprocessor, microcomputer technology, bit-slice processor, microprogramming

1. General Problem Area

Recently, a decisive factor in making industrial processes which bear on the national economy more intense and more efficient was to replace wire-programmed logic increasingly by memory-programmed logic. This trend was facilitated by the rapid development of microelectronics and the associated possibility of economically producing microcomputers, e.g. on the basis of the microprocessors U 808 D and U 880 D. Such freely programmable processors are capable of mastering a broad spectrum of tasks, functioning as cores of microprocessor systems. This capability is based on their architecture (width of address and data buses, number of control signals, instruction codes), and on their processing speed, which depends on their fabrication technology.

However, there are tasks which cannot be solved or which can be solved only unsatisfactorily by means of these processors. The reason for this lies in the properties of the MOS processors, such as

a fixed instruction code
width of the data bus is 8 bits
technologically related processing speed, depending on the architecture (cycle frequency 500 kHz for the U 808 D and 2.5 MHz for the U880 D; time-multiplex utilization of an 8-bit bus for addressing and data signals with the U 808 D).

As an example, one need mention here only the fast drive controls and regulators as well as the application for Fast Fourier Transforms with high accuracy and real time requirements. In such cases, however, microprogrammable microprocessors in bit-slice technology can frequently provide a remedy. Such microprocessors can e.g. be built up with switching circuits of the series K 589 (USSR) [1]. At Tesla, too, the equivalent switching circuit types are being developed and/or produced with the switching circuit series MH 3000 [2]. The switching circuits belonging to the switching circuit series are assembled in Table 1.

The processing speed of such systems naturally is also limited by the fabrication technology. They represent the Schottky-TTL technology and can be operated with a maximum cycle frequency of about 10 MHz. By suitable selection of the design architecture of the system (the two-bit wide processor wafers can be cascaded up to a maximum width of 320 bits, microinstruction call and processing can overlap...) and by specifying an instruction code that represents an optimum with respect to the problem being solved, it is possible to solve a series of problems, whose solution cannot be attacked with the processors mentioned in the introduction. Furthermore, bit-slice systems are frequently also used to simulate instruction codes of technically outmoded computer systems for the more effective processing of proven software packets.

8348
CSO: 1851/4

TTL SWITCHING CIRCUIT TYPES IN USSR

East Berlin RADIO FERNSEHEN ELEKTRONIK in German Vol 31, No 5, May 82 p 278

[Unattributed response to a reader inquiry]

[Text] What equivalent types of TTL switching circuits does the USSR have for Western types (Texas Instruments, etc.)?

L.K., Luckenwalde

According to Soviet documents, the following component series count as equivalent to the TTL series common in Western foreign countries:

USSR

K 130	54 H	(flat pack housing)
K 131	74 H	(DIL housing)
K 133	54	(flat pack housing)
K 134	54 L	(flat pack housing)
K 136	54 L	(flat pack housing)
K 155	74	(DIL housing)
K 158	74 L	(DIL housing)
K 530	54 S	(Flat pack housing)
K 531	74 H	(DIL housing)
K 555	74 LS	(DIL housing)

These series are still being supplemented, i.e. at this time they are not yet always complete. Other TTL series form the USSR, such as e.g. the series K 141, have been outmoded and are no longer being exported. Caution when purchasing without documentation: We generally do not find it possible to still dig up data for these ICs.

8348
CSO: 1851/4

INTERFACE FOR BIDIRECTIONAL DATA TRAFFIC BETWEEN MEASUREMENT ACQUISITION, TRANSMISSION SYSTEM, MICROCOMPUTER

East Berlin RADIO FERNSEHEN ELEKTRONIK in German Vol 31, No 5, May 82
pp 283-286

[Article by Bernd Schildwach, Institute for Space Research, GDR Academy of Sciences]

[Text] The use of components with different production technologies is often necessary for the effective application of microelectronic components, so as to increase the effectiveness of the resulting systems. The CMOS switching circuit K 564 IR can be coupled to the CPU U 880 D without technical problems and permits a large number of applications with very little extra power and software in specific applications.

Microelectronic components are used in the acquisition of measurement data in measurement acquisition systems and in the transmission of data to subsequent systems. In these applications, they offer the possibility of effectively organizing the acquisition of measurements and the processing of data, with very little expenditure for components, power, and volume. By data processing is understood the formation of data corresponding to the conditions necessary for their transmission. Into this category fall, for example, data organization corresponding to the fixed word length, the formation of test bits, etc. Individual bits can be emphasized or suppressed with a mask, through a logical operation in the microcomputer. Thus, the evaluation of the measurement data in the receiving station of the telemetry system is facilitated. On the basis of the measured results, the microcomputer can correct the measurement process, by switching over the measurement channels or, for example, changing the amplification factors. If the measurement acquisition system (see Figure 1) also requires analog measurement variables, the conversion result of the analog/digital converter can be corrected by the microcomputer, if the conversion function of the ADU is defective.

The interface described here has the aim of staticizing a parallel data stream with a word length of nine bits. The eight high bits are to be processed, checked and replaced by the microcomputer corresponding to the operating conditions prescribed by the user. After being supplemented with a test bit, the interface provides its output in a word length of ten bits. The power consumption should be kept low, and the power supply for the interface is derived from the operating voltage of 5 V.

Bidirectional Interface

In implementing the interface module, the CMOS switching circuit of the Soviet series K 564 IR 6 was used for coupling to the CUP U 880 D. This is equivalent to the types CD 4034 (RCA) and MC 14034 (Motorola); it is often regarded as a universal bus register. Consisting of a cascade with eight flip-flops, the transmission unit with eight transmission gates (transfer gates), and the control, this switching circuit can be used for many tasks. Thus, in this configuration, it can be used as an eight-bit memory, as a shift register (series-parallel converter) or as a buffer memory for a bus coupling (Figure 2).

The transmission unit makes possible data exchange between the data bus A and the data bus B from and to the memory cascade. The transmission unit of the bus A also makes it possible to switch off the data lines (high-ohm). Bus B has a constant connection to the memory cascade, and a control unit specifies the direction of data exchange. The following control possibilities result:

The data line is switched off to the bus A:

If level 1 is set at the input AE, the data lines from bus A are switched on. With level 0, the data lines are switched off in high-ohm fashion.

Data exchange between bus A and bus B is controlled:

Input AB:
0 ≙ data exchange from bus B into the memory cascade, the output to the memory cascade is at bus A, depending on the control instruction AE
1 ≙ data input in dependence on control instruction AE into the memory cascade, the outputs to the memory cascade are in bus B

Control of the serial or parallel data input

Input PS:
0 ≙ data are inputted serially through the input into the memory cascade
1 ≙ data input occurs in parallel over bus A or bus B over the memory cascade in dependence on the control input AB

Control of the cycle dependency of the parallel data input

Input AS:
0 ≙ transmission of data takes place with the 0-1 cycle edge
1 ≙ transmission of data takes place independent of the cycle.

Table 1 gives a comprehensive survey of the various operating modes of the switching circuit. Table 2 and Figure 3 list all the operating conditions and delay times. As a supplement of the technical documentation from the component manufacturers, the time t_v was determined because, according to [1], with a 1-0 transmission of the control instruction PS, the control AS should be zero, to prevent an undesirable shift of the data in the memory cascade. Investigations on several switching circuits confirm the correctness. The time t_e was likewise determined by outselves, since it can be important for some applications, if data have already been written in the memory cascade and are to be outputted in a short time through the AE instruction. This case occurs, for example, when the switching circuit is used as an INT-vector port if, in the INT-acknowledge cycle, the low part of the INT vector is read from the port.

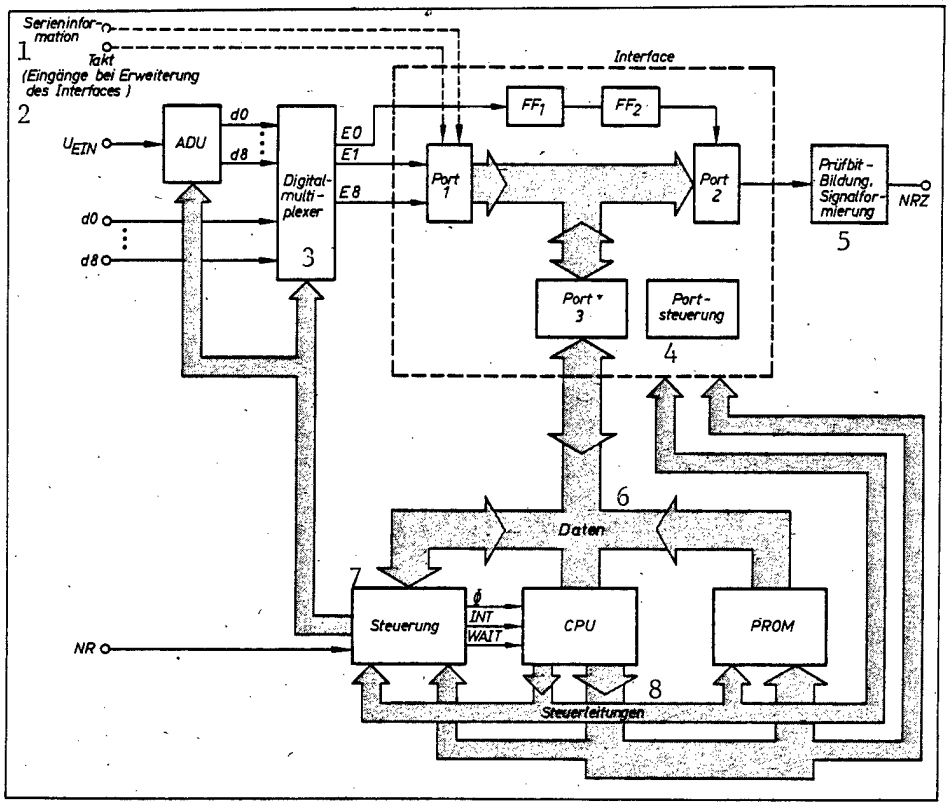

Figure 1: Block circuit diagram of the measurement acquisition and transmission system

1 serial information
2 cycle (inputs for expansion of the interface)
3 digital multiplexer
4 port control
5 test bit formation, signal shaping
6 data
7 control
8 control lines

Staticizing the Data in Port 1

The offered data are staticized and stored at the time t0 (see Figure 4, Figure 5), while the bits E1 through E8 are staticized in port 1, the bit E0 is staticized in the first D-flip-flop of the IC K 564 TM 2.

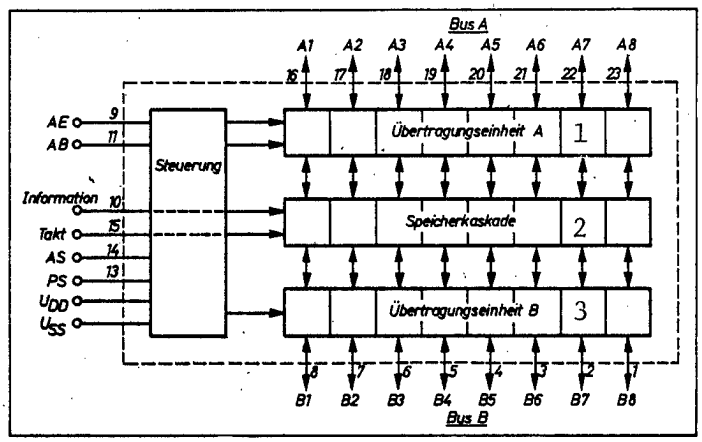

Figure 2: Block circuit diagram of the K 564 HP 6

1 transmission unit A
2 memory cascade
3 transmission unit B

Level 1 at the AS input causes staticization in port 1 (asynchronous staticization), while, outside of this time, a shift of the data in the memory cascade is not possible, since the cycle input has the level 0.

Staticizing the Data in Port 2

Port 2 has the function of an output port:

Staticizing the data being outputted
Parallel-serial conversion.

The parallel data are staticized at the time T9. At the same time, the bit E0 is staticized in the second D-flip-flop and subsequently is situated at the serial input of port 2. Since the signal T9 is derived from the 0-1 position of the bit cycle T_B, the following 0-1 transition at the end of T9 for the content of the memory cascade in port 2 is not considered because of gate running times. After PS_2 and become zero, the information is right shifted and the bit E0 is staticized in port 2. At the cycle time T8, this bit E0 appears at the output 8, in the case of parallel-serial conversion.

Data Traffic Between the Microcomputer in Port 3

From the previous description, it is apparent that the parallel information is staticized in ports 1 and 2 at different times. This difference can be used in data traffic with port 3, such that the microcomputer can perform operations for input and output and for data processing.

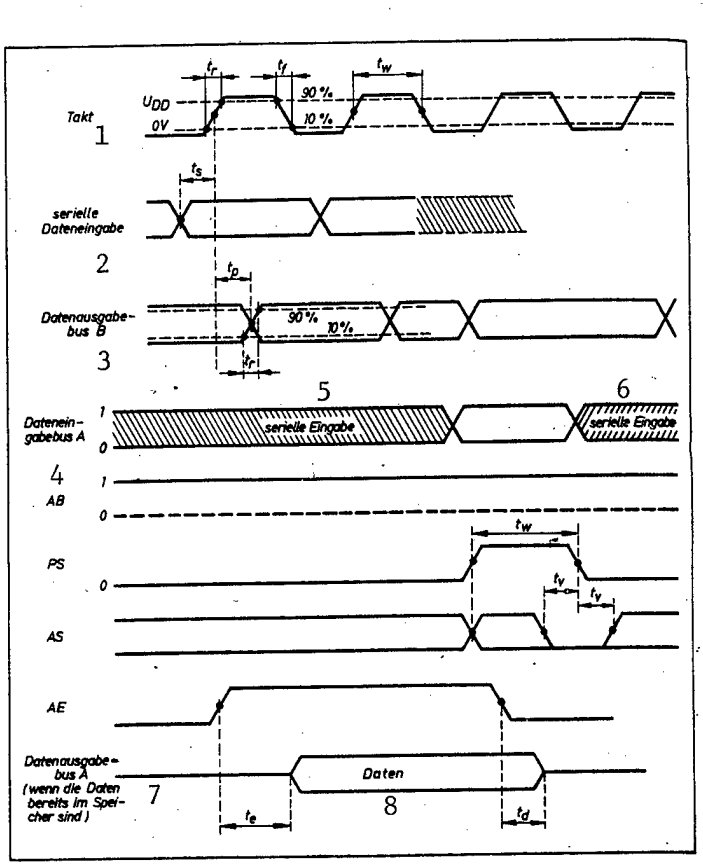

Figure 3. Switching times of the K 564 HP 6

1 cycle
2 serial data input
3 data output bus B
4 data input bus A
5 serial input
6 serial input
7 data output bus A (if the data are already in memory)
8 data

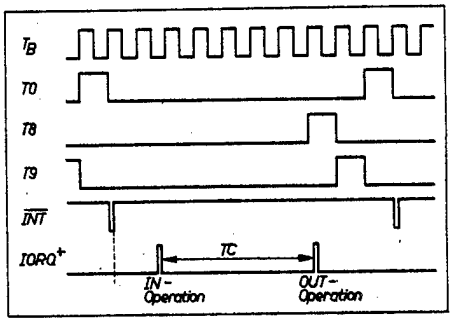

Figure 4. Pulse diagram of the interface

Table 1. Logical functions of the IC K 654 IR 6

AE	PS	AB	AS	Function
0	0	0	X	Serial data input, bus A is switched off, bus B is ineffective to the "data input" mode
0	0	1	X	Serial data input, bus A is switched off, bus B in the "data output" mode
0	1	0	0	Parallel data input, cycle-synchronous, bus A switched off, bus B in the "data input"mode
0	1	0	1	Parallel data input, asynchronous to the cycle, bus A is switched off, bus B in the "data input" mode
0	1	1	0	No change of the data in the cascade, bus A switched off, bus B in the "data output" mode
0	1	1	1	No change of the data in the cascade, bus A switched off, bus B in the "data output" mode
1	0	0	X	Serial data input, bus A in the "data output" mode, bus B is ineffective through the "data input" mode
1	0	1	X	Serial data input, bus A is ineffective through the "data input" mode, bus B in the "data output" mode
1	1	0	0	Parallel data input, cycle-synchronous, bus A in the "data output" mode, bus B in the "data input" mode
1	1	0	1	Parallel data input, asynchronous to the cycle, bus A in the "data output" mode, bus B in the "data input" mode
1	1	1	0	Parallel data input, cycle-synchronous, bus A in the "data input" mode, bus B in the "data output" mode
1	1	1	1	Parallel data input, asynchronous to the cycle, bus A in the "data input" mode, bus B in the "data output" mode

Table 2: Operating conditions and delay times of the IC K 564 ИР 6

	U_{DD} in V	min.	typ.	max.
Fortpflanzungszeit t_p in ns 1	5	—	600	1 200
	10	—	240	480
Übertragungszeit t_T in ns 2	5	—	250	750
	10	—	100	300
min. Taktimpulsbreite t_w in ns 3	5	—	200	400
	10	—	100	175
AE-, PS-, AS-Impulsbreite t_w in ns 4	5	—	240	480
	10	—	85	195
Anstiegszeit t_r und Abfallzeit t_f des Taktes in µs 5	5	—	—	15
	10	—	—	15
Vorsetzzeit der Daten t_s in ns 6	5	—	250	500
	10	—	100	200
maximale Taktfrequenz f_{cl} in MHz 7	5	1,5	2,5	—
	10	3,0	5	—
Vorstellzeiten 8				
t_v in ns	5	200	—	—
	10	100	—	—
t_v in ns (50 pF)	5	200	400	600
t_d in ns	5	150	200	—

1 propagation time
2 transmission time
3 minimum cycle pulse width
4 AE-, PS-, AS-pulse width
5 rise time t_r and decay time t_r, of the cycle in µs
6 lead time of the data
7 maximum cycle frequency
8 lead times

The following control functions result for port 3:

$$AE_3 = \overline{T9} \cdot IORQ^+ \cdot \overline{NR} \qquad (1)$$

$$IORQ^+ = \overline{IORQ} \cdot (\text{port address}) \qquad (2)$$

Thus, bus A of port 3 is capable of control only during input or output operations outside the staticizing time in port 2 and in the case of a non-commanded emergency regime (NR).

$$PS_3 = AS_3 = 1 \qquad (3)$$

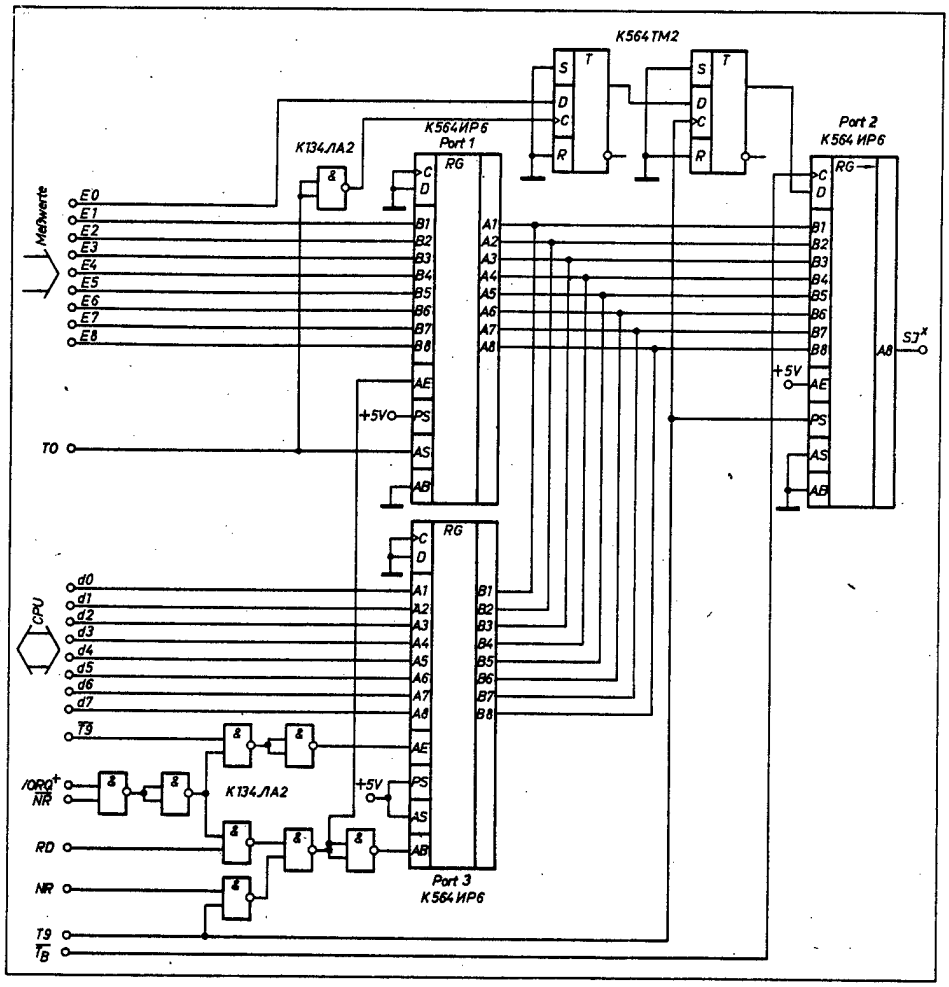

Figure 5 Block diagram of the interface

1 measured values

Thus a constant cycle-independent mode of operation as well as parallel staticizing of the data into the port is possible.

$$AB_{;} = \overline{\overline{IORQ^{+}} \cdot NR \cdot RD} \cdot \overline{NR \cdot T9} \qquad (4)$$
$$= \overline{AE_{1}}$$

From this function can be seen that the use of the internal bus between the individual ports can be utilized only by one port, port 1 or port 3. If the emergency regime NR is commanded, the data in port 2 are always staticized from port 1 at time T9, and the B bus of port 3 is switched to input. The emergency regime

represents a possibility which guarantees the operational mode of the data transmission system if a component of the microcomputer should fail.

If the data from port 1 are to be transmitted to port 2, the must first be staticized in port 3 by a read instruction of the microcomputer. There they can remain until their staticization in port 2. They are analyzed, reformed, or exchanged by the microcomputer. It is thus indispensible for each data word to initiate an interrupt instruction to the microcomputer (to the system control). It is directly possible to connect port 3, i.e. the data bus B of the switching circuit K 564 IR 6, to the data bus of the microcomputer.

Time Conditions

Table 2 specifies the delay times also as a function of the operating voltages. With a 5 V operating voltage, as compared to operation at 10 V, there are much larger pulse widths and delay times. An operating voltage of 5 V was specified for the interface. The CMOS ICs of series A 564, to implement the logical functions (1) through (4), have delay times that are too long (about 400 ns/gate) so that the operating possibilities of the microcomputer are impaired because of the necessary WAIT cycles. Since the low power consumption is a criterion in designing the interface, the low powered TTL switching circuits (K 135, K 136) were used, whose power consumption is about 1 MW/gate, and whose delay times are less than 35 ns.

With an input and output operation, the duration of $\overline{\text{IORG}}$ is about 950 ns, if the cpu cycle frequency amounts to $\emptyset$ = 2.5 MHz. According to Table 2, the duration t_p may be 1.2 µs, and thus the time for staticizing the information from port 1 through port 3 into the microcomuptuer is not sufficient without an additional WAIT cycle. If this WAIT cycle is derived from the $\overline{\text{IORQ}}$ signal of the CPU, it will also be valid in staticizing the INT vector and the IC K 564 IR 6 can, under these conditions, also be utilized as the INT-vector port.

Program Possibilities

In setting up data processing programs, the starting point must be the time conditions of the data acquisition and transmission system. For example, if the transmission cycle of the system t_0 = 10 kHz, the following time results between staticization in ports 1 and 2, which is available to the microcomputer as processing time:

$$TC_{max} = 8\,TZ - 4\,TM - t_v \qquad (5)$$

$TZ = 1/f_C = 100\ \mu s$
$TM = 1/\Phi = 400\ ns$
t_v = processing time for port 2.

This results in a time TC_{max} = 7.27 µs. This time is sufficient to control the data selection of the measurement acquisition system, for outputting status words to analyze the internal status of the system, and for the coarse analysis and correction of the transmitted data. If the data transmission frequency is increased, a second microcomputer should be provided for larger processing programs.

The correction of the data by the microcomputer will be demonstrated here using as an example the acquisition of analog data. The processing of these data transverses

a sensible process in the analog/digital converter. Errors can occur here, which are described in their totality in [2]. The zero point of the ADU can be sensed rather easily and can be represented as a digital quantity. This acquisition takes place with a conversion of the zero-point level, by occupying the input of the ADU or respectively a channel of the analog multiplexer with the zero-point level. The correction of the microcomputer can limit the digital storage method cited in [3] to a signed addition of the analog word (or its digital equivalent) with the correction word. It is thus possible to circumvent storage of the analog correction quantity in a sample and hold circuit.

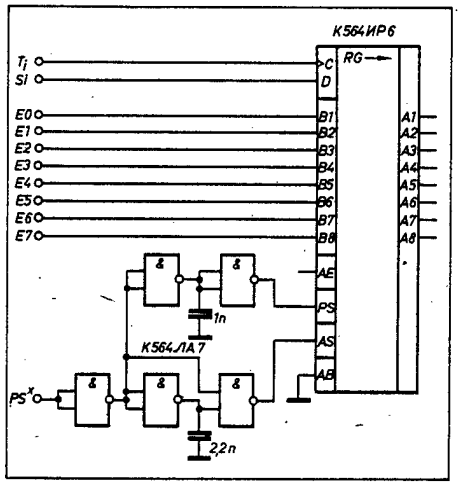

Figure 6: Circuit example for the control of inputs PS and AS

The program must take into account that, when full scale (FF_H) has been reached, the further addition may yield only the result FF_H. This also makes apparent the disadvantage of this correction method as compared to the one presented in [3]. In [3], the full quantization level can be maintained through an integration method, while, by means of the present method, the quantization steps can be specified only in terms of

$$Q = 2^n - KW \tag{6a}$$

$$KW = d_{n-1} \cdot 2^{-1} + \ldots + d_1 \cdot 2^{-(n-1)} + d_0 \cdot 2^{-n} \tag{6b}$$

n = number of bits of the conversion results

Experimental Possibilities of the Interface

The switching circuit K 564 IR 6 affords the possibility of serial-parallel conversion. This property can be used to input serial data if, for example, stored sensors have already digitalized their measurements through their own analog/digital converter. This input line is shown in Figure 1 as a dashed line. Since one must distinguish between parallel and serial input, this results in changes in the control of the system. For a control function that must then be implemented, one obtains the following:

$$AE_1 = T9 \cdot \overline{MW} \quad (7)$$
$$PS_2 = T9 \quad (8)$$
$$AE_3 = \overline{T9} \cdot IORQ^+ \quad (9)$$
$$AB_3 = T9 \cdot MW \vee \overline{T9} \cdot RD \quad (10)$$

Equations (7) and (10) also show a different situation with respect to the acquisition of a measurement word (serial or parallel) and with respect to the data transmission status word (MW). This distinction is made by the overall control, which is not described in the present paper. This status word provides information concerning the internal status of the system, but it can also present an analysis of the measured status, e.g. the formation of average values. In implementing the circuit, it is no longer possible to exchange data through the microcomputer during the transmission of the data, since the shifting in of the data into the serial input of port 1 requires a finite time which additionally, in asynchronous operation, is defined only within the time limit. This time limit is formed by the moment T9, at which the data from port 2 must be staticized. Corresponding to the necessary time conditions for the switching circuit K 564 ИР 6, Figure 6 shows a circuit proposal for driving the PS and AS control input.

Signal Shaping, Generation of the Test Bit

Before the information from the interface is conducted further for data transmission, it must be coded in accord with the system conditions. By signal shaping is here understood a synchronization of the serial information with the cycle frequancy. This is done by staticizing the data in a flip-flop with the 0-1 edge of the bit cycle. Before signal shaping, the serial word, which consists of nine weighted bits, is supplemented with a test bit. This supplementation occurs upon an odd number of all 10 bits of the resulting word. The test bits are generated statically or dynamically. The test bits are generated statically in parallel to the digital word being tested. The implemeted circuit represents an antivalence of all the bits being tested. However, if a serial data stream is present, dynamical formation of test bits is suitable.

This principle is based on counting out the number of bits with level 1 in one data word. It is here ncessary that the beginning and end of the word are identified by the system control. The signal T9 is used for the beginning of one word and the signal T8 for its termination.

Within the cycle time T9, the flip-flop with its feedback at a modulo-2 counter is reset (Figure 7). Since the serial information from port 2 is shifted by half a cycle with respect to the counting cycle, each bit is tested in the middle for its logical level. With a logical 1, the counter advances. At the time T8, the results of the counting process are placed at the input of the shaping flip-flop and are thus inserted as the tenth bit into the serial information.

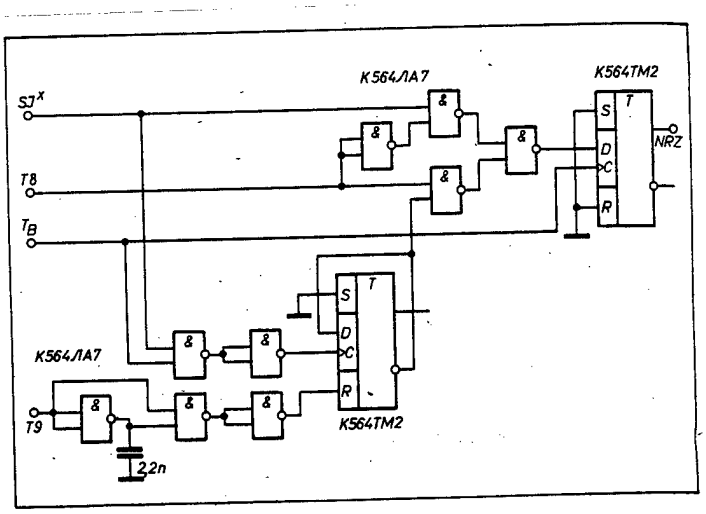

Figure 7: Illustration of the test bits and signal formation

For the sake of completeness, the formation of the test bit is also supplemented by the microcomputer. However, in the present case, this solution does not appear advantageous, since the hierarchy of the emergency regime can then no longer be implemented, and testing the bit D0 would require greater hardware expenditure.

BIBLIOGRAPHY

1 McMOS Manual: Products - Properties - Applications. Third Edition, Motorola 1975.

2 Schildwach, B.; Stroetzel, K.H.: Balancing and testing of an analog-digital and digital-analog converters, radio fernsehen elektronik [Radio, Television, Electronics] 27 (1978) No. 7, pp. 425-427.

3 Schildwach, B.: Automatic zero-point correction in an analog-digital converter, radio fernsehen elektronik [Radio, Television, Electronics] 25 (1976) No. 13, pp. 435-438.

8348
CSO: 1851/4

UDC 681.3.06.003

CALCULATION AND ANALYSIS OF TECHNICAL-ECONOMIC INDICATORS OF DEVELOPMENT OF SOFTWARE

Kiev UPRAVLYAYUSHCHIYE SISTEMY I MASHINY in Russian No 1, Jan-Feb 82 (manuscript received 30 Jun 80) pp 3-7

[Article by Vladimir Vasil'yevich Lipayev, doctor of technical sciences, Moscow, Viktor Ivanovich Makarov, engineer, Moscow, and Aleksey Ivanovich Potapov, candidate of technical sciences, Moscow]

[Excerpts] Existing technological processes of software development can be classified by the level of automation of the primary stages and elements of software development as follows:

Zero level — programming completely in machine codes without automated equipment;

1st level — elementary translator from autocode with subsequent debugging in machine codes and partially automated production of documents; the volume of technological programs does not exceed 1,000 commands, and their total cost is 1,000-2,000 rubles;

2nd level — automated programming equipment includes a modern translator from autocode, elementary data base, text editor, program to shape certain documents, and possibly elementary means for self-contained debugging; total volume of technological programs is about 10,000 commands and their cost is in the range of 10,000-20,000 rubles;

3rd level — corresponds to fairly modern programming technology in which automated equipment has a volume of about 100,000 commands and a cost reaching 200,000-500,000 rubles; it includes translators from high-level languages, means for self-contained and integrated debugging, and various other types of automation equipment;

4th level — the most sophisticated, in contemporary terms, systems for automating program design, including sets of translators from various languages, means of monitoring and verifying programs, elaborate means for planning, monitoring, debugging and testing, and so on; these systems have means of automating practically all elements of the technology of designing complex programs. The total program volume is on the order of 1 million commands, and the cost of such systems is 3-5 million rubles.

When general-purpose computers, which are widely employed in the national economy, are used for software development the cost of one hour of machine time can be determined by the price list [4], which establishes the following rates for one machine hour of computer use:

M22, Minsk-32, BESM-4	35 rubles
YeS 1022	85 rubles
YeS 1033	110 rubles
BESM-6	100 rubles

A project was carried out to collect primary statistical data on the characteristics of developed software in order to determine the technical-economic indicators of certain software development projects underway in the sector. To do this a table was compiled corresponding to the first stage of collection of statistical data, by the retrospective method, on the general characteristics of development of software for computers included in various types of automated systems. Tables to be filled out were distributed to the technical directors of software development for the following systems:

- automated data processing and control systems working in real time (ASU-1 and ASU-2);
- systems to automate the design of radioelectronic equipment (SAPR REA RAPIRA-3);
- systems to automate software development for specialized computers (SAPO YaUZA-6, versions A, V, and D) [5];
- systems to automate software development for microprocessors (SAPO TEMP) [6].

While filling the tables out on an experimental basis, difficulties were encountered answering the question about the length of development, especially concerning the actual beginning of software development. The reasons for this were the following:

- transition during the development process to another type of computer (for example, a transition from the M220 to a YeS computer for the RAPIRA-3 system);
- change in software development technology (switch from a high-level language to the YaUZA autocode for system ASU-1);
- overlapping of stages of developing of versions of the system and its practical use (SAPO YaUZA-6, versions A, B, and D).

The answers to the questions about the number of development workers and their average monthly pay were complicated by the fact that, in accordance with official instructions, data are kept in divisions of labor and wages and production planning divisions for just the last three years, but the development of sets of programs sometimes takes longer. The psychological factors noted in [3] also had a strong impact on how well the tables were filled in and how long it took.

Difficulties occurred with determining the volume of the software developed in cases where ready-to-use programs were borrowed for refinement in order to solve additional problems with them. This was most typical where the SAPO YaUZ-6 was modified to solve the problems of the SAPO TEMP. In this situation the refined programs have many new qualities, and the volume of software developed must be characterized by two quantities. The first quantity is the volume of the newly developed programs (the supplementary work), which is determined by the total number of commands modified and added to the refined program. The second quantity characterizes the full volume of the programs of the refined subsystems because the programmer in fact studies and analyzes the subsystem programs in their full volume to determine where modifications and supplements are necessary in the program being refined. The stage of self-contained debugging is partially repeated and the stage of integrated debugging is almost entirely repeated for the refined program, and documentation is amended or even redone. The determining factor for the values of labor expenditures in refining programs is the existence and completeness of design documentation. Thus, in the case of programs that are modified in this way development will be characterized by two series of values of technical-economic indicators because of the dual character of measurement of the volume of software development.

The technical-economic indicators of software development for a number of specific systems were calculated according to formulas (1)-(4) for the initial statistical data of the data collection table in the first stage. Table 2 (next page) gives a graphic representation of these calculations. The first thing that should be noticed is the great difference in technical-economic indicators depending on the purpose and description of the software and conditions of its development. For example, the labor productivity achieved during development of software for the RAPIRA-3 system in which virtually no requirements were imposed for degree of use of computer resources relative to memory and productivity and the documentation factor was low (0.002) is 5.1 commands per person per day. In development of software for the real-time ASU-1 and ASU-2 systems in which requirements for degree of use of memory and productivity resources are close to one, the documentation factor is high (0.133), and a much greater volume of testing was done, labor productivity is 1.1 commands per person per day.

The range of variation in labor productivity for program development for the 24 subsystems of the SAPO YaUZA-6D is very broad, from 0.5 to 14 commands per person per day. Given uniform technological software development means for all development workers this large spread in labor productivity values for different groups of development workers is explained by different levels of qualifications, the differing algorithmic complexity of the subsystems, and different amounts of scientific work already done on the subject matter of the subsystems. In our investigation the term qualifications was taken to mean the level of knowledge and experience of the programmer in the field of the assigned task and with technological means of software development. The average productivity of one programmer from a collective of development workers, including technical supervisors, managers, and auxiliary personnel, was 3.4 commands per person per day for the YaUZA-6A, but rose to 4.6 commands per person per day for the YaUZA-6D. This should be explained by the higher level of qualifications of development workers and by the greater backlog of scientific work for a number of subsystems.

We should emphasize the great differences in the values of the individual labor productivity of programmers for software modules and subsystems taken separately

Table 2. Technical-Economic Indicators of Software Development

Software	Development Time, years	Number of Commands	Average Number of Employees Participating in Development	Labor Productivity, Commands per person per day	Cost of Design of One Command, Including Overhead Costs, rubles Per Command	Cost of Machine Time per Command rubles per Command	Cost of Development of One Command, rubles	Volume of Design and Operating Documentation, Fll Sheets	Number of Fll Sheets of Documentation per Command
SAPR RAPIRA-3	4	125,000	25	5.1	3	1	4	250	0.002
SAPO TEMP									
Newly Developed Programs	2.3	17,000	16	1.9	6	1	7	300	0.05
Refined programs	2.3	88,000	16	9.6	1	0.2	1.2	300	0.05
SAPO YaUZA-6A	2	55,000	32	3.4	4	2	6	-	-
SAPO YaUZA-6V	4	238,000	47	5.0	3	2	5	-	-
SAPO YaUZA-6D	6	342,000	49	4.6	3	2	5	880	0.002
ASU-1	6	75,000	47	1.1	15	33	48	10,000	0.133
ASU-2	9	47,000	20	1.1	12	43	55	1,500	0.03

and the average productivity per person from a collective of development workers as a whole taking acount of the full cycle of development from systems analysis to turning the software over for testing. For example, the individual labor productivity of specialists doing the actual programming in ASU-1 is 3.2 commands a day, while the average productivity per person in the collective of development workers is 1.1 commands a day.

Analysis of the processing results showed that comparison of the efficiency of software development for different automated systems, even of one class, should be done by a group of technical-economic indicators, not for any one indicator. For example, with ASU-1 and ASU-2 identical values were obtained for labor productivity, but the documentation factor for ASU-1 was almost five times as high as ASU-2, which will make it possible later to conserve capital when accompanying the software of the ASU-1 in actual use and updating it.

The evaluation of the efficiency of developing software for the SAPO TEMP deserves special notice. Table 2 gives two columns of values for its technical-economic indicators. If we look only at the first column of indicators for newly developed programs, we could conclude that the efficiency of software development for the SAPO TEMP is only one-half to one-third of the efficiency of development of similar systems for the SAPO YaUZA-6 or SAPR REA RAPIR-3. But this conclusion would be one-sided, because by using the method of developing the SAPO TEMP primarily by refining the programs of the SAPO YaUZA-6 a system with a new quality was obtained which allowed automated development of microprocessor programs with a range of soluble problems equal to that of the SAPO YaUZA-6. Therefore, estimates of software development of systems which borrow ready-to-use programs for refinement should be supplemented by a second column of technical-economic indicators which are calculated based on the total volume of all refined programs.

Conclusions. Our project of collecting statistical data by the retrospective method concerning particular projects to develop software and analysis of processing results confirms the possibility of evaluating the technical-economic indicators of software development by the proposed technique. Analysis of processing results made it possible to evaluate in some degree the factors that affect technical-economic indicators. The values of technical-economic indicators given for the software of actual systems may be used as guidelines for estimating the time and labor expenditures necessary to develop similar software for new automated systems.

11,176
CSO: 1863/139

UDC 681.14.2

SOME POSSIBILITIES OF STRUCTURAL PROGRAMMING OF AVTOKOD LANGUAGE OF EL'BRUS-1 MULTIPROCESSOR COMPUTING COMPLEX

Kiev UPRAVLYAYUSHCHIYE SISTEMY I MASHINY in Russian No 1, Jan-Feb 82 (manuscript received 7 Apr 81, revised 24 Jun 81) pp 7-11

[Article by Khachadur Dzadurovich Dzhenibalayev, candidate of technical sciences, Computing Center of Rostov State University, Rostov-na-Donu]

[Text] Introduction. This article considers means for structural programming of AVTOKOD language [1] of the El'brus-1 multiprocessor computing complex (MCC). The El'brus-1 MCC is the first domestically produced fourth-generation computer [2] whose main architectural elements are a hardware stack, tagged memory, context data security, and operations on arrays. The AVTOKOD language of this complex is a recursive, procedural machine-oriented high-level programming language. In terms of its capabilities it significantly surpasses ALGOL-60 and is comparable to ALGOL-68; it has the means to construct well-structured programs.

Considering the chief prerequisite of structured programming to be making it easier for the human being to perceive the program, we will take up only a few of the requirements mentioned in [3] [4] which structured programs must meet. Specifically, we will consider the principal control structures, the technology of work with multilayered data structures consisting of sequences of syllables, and certain other characteristics of AVTOKOD which are of interest from the standpoint of structured programming.

The material presented is the result of an experiment with the use of the AVTOKOD language to write a compiler from the language SIMULA-67 [5] for the El'brus-1 MCC.

The use of control structures. Work [6] expressed the idea that there should be a simple correspondence between the text of the program and the procedure for executing it in order to make it easy to follow the logic of the program. But this is made much more complicated by disorderly use of the transfer (go-to) statement where control is transferred either forward or backward. This article analyzes AVTOKOD means which make it possible to minimize the use of the transfer statement or, possibly, get by without it completely in the form in which it is found in most known algorithmic languages for high-level programming. We will consider the principal control means of structured programming and compare them with the corresponding AVTOKOD means.

There are five known constructions for structured programming: sequence; if-then otherwise; selection; do as long as; and, do until. The first three

constructions are found in AVTOKOD with syntactical representation that makes it easy to read the program and is similar to that used, for example, in the languages ALGOL-68 and YaRMO. The if-then otherwise construction in AVTOKOD has the appearance IF-THEN OTHERWISE-ALL, and the choice construction has the appearance CHOICE-ALLCHOICE.

The constructions of AVTOKOD which differ fundamentally from the above-mentioned five constructions of structured programming are the structured sentence and, related to it, the structured transfer. The structured sentence has a title which gives the identifier of the situation which may occur in the closed sentence directly following the title. A closed sentence is one which has opening and closing parentheses (BEGINNING-END, IF-ALL, CHOICE-ALLCHOICE, CYCLE-REPEAT, and so forth). Within the closed sentence the programmer monitors the occurrence of the situation mentioned in the title of the structured sentence and makes the structured transfer, according to which execution of the closed sentence stops and control is transferred to the statement following it.

The additional capabilities related to use of the structured sentence are, in the first place, that the structured transfer can be parametered and, in the second place, that there can be several situation identifiers. In the special construction that completes the structured sentence and is called the action prefix it is possible to program actions that are specific to each particular situation or to several of them. In the third place, there are identifiers of static and dynamic situations. Static situations are those which may occur only within the closed sentence. Dynamic situations may also occur in constructions that are not located within the closed sentence but are dynamically included in it.

An important feature here is that regardless of whether the structural transfer occurred according to dynamic or static situation X, control is always transferred to the statement that follows the closed sentence which was preceded by the title of the structured sentence with the identifier of situation X. The structured sentence and structured transfer essentially overlap the capabilities of the constructions of do-as-long-as and do-until as well as the LEAVE construction of the BLISS language [7], which allows leaving a cycle and procedure ahead of schedule. Just as control, branching according to an arbitrary or chosen sentence, ultimately leads to a point that marks their conclusion so the structured transfer leads not to an arbitrary point in the program, but always downward, towards the end of the corresponding closed sentence. This is the fundamental difference between the structured transfer and the conventional transfer by tag.

But the most significant advantage of the structured sentence is that the situation identifiers enumerated in its title precede entry into the closed sentence, informing the program reader in advance of how execution of the closed sentence may end and what its content designation is. This is a very important property of the structured sentence from the standpoint of program recognizability.

The above-described organization of the structured sentence requires definite programming discipline, involving the fact that it encourages thinking through program situations (in content terms, not algorithmic terms) whose occurrence may lead to the end of the closed sentence, and designating these situations with appropriate identifiers. Neither external commentary nor marking the closed sentence with an appropriate mnemononic tag, which incidentally suggests the

the thought of possible transfer of control to this point of the program, can be called an adequate means compared to the corresponding means of the structured sentence of the AVTOKOD language.

Considering the purpose of structured programming as convenient program reading, we will note here that the cycle statement has a very good mnemonic notation in the form of cycle-repeat parentheses which remind the program reader of the cyclical segment at both its beginning and its end and, moreover, adequately reflect the meaning of the repeating program segment. This cannot be said, for example, of ALGOL-68 (DO-OD), and even less of ALGOL-60 in which the pair of symbols BEGIN-END is already quite overloaded.

The technology of work with data structures. A shortcoming of AVTOKOD as a system programming language is that it does not have entries analogous to those, for example, found in the PASKAL' language [8]. As long as we are referring to an entry (structure) that occupies one word of the El'brus MCC, AVTOKOD means are adequate and efficient. There is a possibility of breaking the word into syllables (fields), linking them to appropriate identifiers, and then using them without thinking about the specific position of the field within the word. But quite often information about objects cannot be contained in one word with a 64-bit format.

The first alternative for achieving the goal is to have each use of a certain field which is not located in the first word of a multiword structure coupled with an indication of how many words after the beginning of the structure the word containing the necessary field is located.

The other alternative, which is considered in greater detail below, is to use the "indirect variable" construction with a trimmer and a corresponding conversion of the format to bit format. Finally, the third possibility is to use bit arrays to store these structures. But this is hardly acceptable because it significantly increases the number of memory bits necessary to arrange the structure indicators.

Fields are generally used very often in system programs and are spread throughout the entire program text. During the debugging process and when modifying execution of a certain meaningful function it may become necessary to correct the set of fields in the structure, their mutual arrangement, and the number of bits allocated to the field. These changes may demand very substantial modification of the program text.

To simplify this modification we propose using a combination of the capabilities of AVTOKOD macromeans (definitions and text substitutions) and means of access to word fields. The first alternative of determining the position of the field in the structure is more convenient for reading and more efficient in realization. Information on displacement of the word relative to the beginning of the structure and the position of the field within the word is fixed once in the field definition, which is a macrodefinition. With subsequent change in the position of the field within the structure or of the entire structure as a whole, only the defining inclusions of the corresponding fields are modified and the many used field inclusions, which are macrocalls, are not touched. This makes it possible when using any field of the structure to indicate only the beginning of the structure (index of the first word of the structure in an array that contains

a sequence of arbitrary structures) and the identifier of the field within the structure, which adequately reflects the necessity of representing the structure as an integral and unitary element.

This alternative of representing structures has a greater degree of flexibility than the entries in the language PASKAL' mentioned above. Greater flexibility is achieved because in this case the programmer himself controls the number of bits allocated to the field, whereas in PASKAL' either the complete word without breaking into component syllables or part of the word is allocated to the field, but in this situation the arrangement is determined by the realization process, not by the user.

For example, information on a simple variable contained in the description table of the compiler, in which information on all the descriptions of the SIMULA program is concentrated, is represented by the following structure:

PVID (6)	PTIP (4)	PFOR (2)	PUR (5)	PBAZA (10)	PIMYA (16)	PTsEP (21)
PATR (2)	PKOORD (15)		PADRES (14)	PKVAL (16)	PKSORDOSh (17)	

The structure consists of two 64-bit words and contains the enumerated fields. The corresponding field definitions for the first alternative may have the following appearance, for example:

```
Text
PVID (TAB, UKAZ) = TAB [UKAZ].  [63:6],
PTIP (TAB, UKAZ) = TAB [UKAZ].  [57:4],
. . . . . .
PATR (TAB, UKAZ) = TAB [UKAZ+1]. [63:2],
. . . . . .
PKOORDOSh (TAB, UKAZ) = TAB [UKAZ+1]  [16:17];
```

In this case TAB is the identifier of a certain unidimensional array that contains 64-format elements and is designed for storage of structures, while UKAZ is the indicator of the beginning of the structure. The text definition is parametered so that the field identifiers are not aligned with the tables. It follows from this that some tables may contain different structures which have, for example the field PVID and the field position PVID the same in all structures, namely the six left-hand bits of the first word.

For the second alternative the definition of these fields should have the following appearance:

```
Text
PVID (TAB, UKAZ) TAB [UKAZ:][F1 0:6])↑,
PTIP (TAB, UKAZ) = (TAB[UKAZ:].  (F1 6:4])↑,
. . . . . .
PATR (TAB, UKAZ) = (TAB [UKAZ:][F1 64:2])↑,
. . . . . .
PKOORDOSh (TAB, UKAZ) = (TAB [UKAZ:].  [F1 110:17])↑;
```

Let us look in greater detail at the semantics of the above-written constructions of the second alternative, assuming that the first one is obvious. The construction TAB [UKAZ:], which has a trimmer in the index brackets, generates a descriptor of the subarray of array TAB beginning with the index UKAZ and going to the end of the TAB array (the descriptor is a special type of data in which information on the array is concentrated: address of the first element, status, number of elements, format of the elements, and so on). The descriptor created, just like the array descriptor TAB, describes format 64 elements. Its further transformation, done for example by the construction [Φ 1 64:2] involve first converting the format to bit format and then transforming the descriptor so that it begins to describe two bits in the TAB array, the 64th and 65th. Finally, on the character ↑ the descriptor obtained is converted into a name, which allows it to appear as the left part of an assignment statement or as the operand of an expression with the renaming necessary in this case.

This kind of conversion is possible, on the one hand, because the machine language contains descriptors and, on the other hand, because it can have several descriptors of different formats in the same physical memory.

In both versions of field definition the entry of information, for example in the PTIP field of the table of descriptions for a certain index UKTOP, looks as follows:

```
PTIP (TOP, UKTOP) : = integer;
```

Selection of a value, for example of the field PATDRES and entering in a certain variable ADR, has the following form:

```
ADR: = PADRES (TOP, UKTOP);
```

We will note the following with respect to the efficiency of these two alternatives of determining the position of a field in the structure. Selecting the field value and entering the information in the field by the second alternative requires three times as much code and on the TEMP interpretation complex [9] takes 2-2.5 times longer to execute than the corresponding selection and entry by means of the first version. The reason for this appears to be the multiple transformations of descriptors. But the advantage of the second alternative is that the field location is not limited by the physical structure of memory. This means that the beginning of a field may be placed in one word and its continuation in another field adjacent to the first.

Some general remarks on AVTOKOD. While a detailed and precise description of AVTOKOD is not our purpose, nonetheless we will consider certain of its other characteristics that are interesting from the standpoint of structural programming.

Work [3] observed: "The struggle against global data is becoming as important as the struggle against the transfer statement, and it is beginning to receive attention in the professional literature." Global data complicate perception of the program, in the first place because their use is usually far removed from the appropriate descriptions and, in the second place, because the value computed at the particular point in the program and assigned to the global quantity is

practically accessible at any point of the program, which does not always correctly reflect the essence of the algorithm. On the other hand, program experience shows that programmers seldom formulate special blocks for such "small" reasons as setting up a small amount of local work data; they prefer to use global data. In this respect it would be convenient for AVTOKOD to include a broad list of syntactical constructions within which it is possible to make declarations with fairly natural fields of localization. Among such constructions, in particular, are the closed sentences mentioned above.

The sequential sentence is another example of a construction that permits the introduction of declarattions. In conformity with the description of the language, the sequential sentence is used to introduce a new context of identifiers by means of declarations and define a sequence of actions performed one after the other by means of statements. In AVTOKOD the sequential sentences are, in particular, the alternatives of the conditional and selected sentences and the lower and upper boundaries of the step cycle. In addition, as in ALGOL-68, when the sequential sentence becomes a part of a closed one, it may be the operand of an expression. Let us look at the conditional sentence of AVTOKOD in more detail:

```
<conditional sentence> : : = IF <internal part of conditional
   sentence> ALL
<internal part of conditional> : : = [<sequence of descrip-
   tions>] <condition> <alternative   then> [<alternative
   otherwise>]
<condition> : : = [<sequence of statements>]  <conclusion>
<alternative   then> : : = OTHERWISE <sequential sentence>|
   INES [expansion unknown] <internal part of conditional>
<sequential sentence> : : = [<sequence of descriptions>]
   [<sequence of statements>] <conclusion>
<conclusion> : : = <sentence>
```

In this entry the parts of language constructions which could be absent are put in brackets. The area of action of the sequence of description from the internal part of the conditional sentence is the entire conditional sentence, while the area of action of the sequence of descriptions from the sequential sentence is the alternative THEN (OTHERWISE) and only this alternative.

The fact that AVTOKOD contains operations to test array affiliation and hardware means for group operations on lines (recopying, retrieval, transfer, editing, packing, and the like) makes it possible to avoid cyclical constructions, which to some degree linearizes controlling the program and improves its cognitive qualities.

For example, after detection of the letter the operation of singling an identifier out from a certain input text may be written as follows:

```
IDENT < : = VKhT poka sredi BUKVITsIFR,
```

where IDENT is a byte array for storage of the identifier and BUKVITSIFR is a bit scale in which the positions of letters and numbers correspond to 1 and the positions of other characters correspond to zero.

The AVTOKOD macros make a certain contribution to structuring program texts. Because they do not have means of macrogeneration, in conformity with the Chisem classification given in work [10], AVTOKOD macros are text macros with positional, not key parameters.

It is characteristic of the AVTOKOD macros that syntactically the body of the macro definition is a sentence, that is, a fully comprehensible and complete linguistic concept, not an arbitrary sequence of symbols. This makes it easier to read programs and frees the reader from having to construe texts at a very low level (symbolic and lexical). In addition, the body of the macro definition is executed in the context that existed at the moment of definition of the macro, while the factual parameters are executed in the context of the macrocall, that is, analogously to the procedural mechanism. This permits a reasonable compromise between time of program execution and memory space occupied by it, because it is easy to convert a procedure definition into a macro definition and vice versa.

In conclusion we will note that this article does not claim to be a complete analysis of the structural programming means of the AVTOKOD language. But experience has shown that the AVTOKOD means we have discussed together with a certain programming style and appropriate formulation of the program listing refelcting the hierarchical inclusion of certain constructions in others are effective means of structuring program texts and promote clear programmer thinking and an improvement in the cognitive qualities of programs.

FOOTNOTES

1. V. M. Pentovskiy, "Fundamental Characteristics of the El'brus MCC AVTOKOD," Moscow, 1977, 28 pages (ITM i VT AN SSSR, Preprint No 6).

2. B. A. Babayan, and Yu. Kh. Sakhin, "The El'brus System," PROGRAMMIROVANIYE, 1980, No 6, pp 72-86.

3. G. Mayers, "Nadezhnost' Programmnogo Obespecheniya" [Reliability of Software], Moscow, "Mir", 1980, 360 pages.

4. E. Yodan, "Strukturnoye Proyektirovaniye i Konstruirovaniye Programm" [Structural Program Design and Construction], Moscow, "Mir", 1979, 416 pages.

5. U. Dal, B. Myurkhaug, and K. Nyugord, "SIMULA-67 -- Universal'nyy Yazyk Programmirovaniya" [SIMULA-67 -- a Universal Programming Language], Moscow, "Mir", 1969, 100 pages.

6. U. Dal, E. Deykstra, and K. Khoor, "Strukturnoye Programmirovaniye" [Structural Programming], Moscow, "Mir", 1975, 246 pages.

7. W. A. Wulf, D. B. Russel, and A. N. Haberman, "BLISS: a Language for Systems Programming," COMMUNS ACM, 1971, Vol 14, No 12, pp 780-790.

8. N. Virt, "The Paskal' Programming Language," ALGORITMY I ORGANIZATSIYA RESHENIYA EKONOMICHESKIKH ZADACH, 1977, Vyp 9, pp 52-86.

9. I. F. Lesovaya, and V. N. Polivanov, "Translator from El'brus-1 MCC AVTOKOD. Model and Realization," in "Tr. Vsesoyuz. Konf. po Avtomatizatsii Proizvodstva Paketov Prikladnykh Programm" [Proceedings of the All-Union Conference on Automation of Production of Packages of Applied Programs], Tallinn, 1980, pp 76-78.

10. P. Braun, "Obzor Makroprotsessorov" [Survey of Macroprocessors], Moscow, "Statistika", 1975, 78 pages.

11,176
CSO: 1863/139

UDC 681.3.053

USE OF R-LANGUAGE FOR DESIGNING ASU PROGRAMS

Kiev UPRAVLYAYUSHCHIYE SISTEMY I MASHINY in Russian No 1, Jan-Feb 82 (manuscript received 26 May 1981) pp 11-15

[Article by Stanislav Vital'yevich Mikoni, candidate of technical sciences, VNII Elektronstandart, Leningrad]

[Excerpts] The steadily growing complexity of ASU [automated control system] problems performed on computers is accompanied by an increase in the volume of specifications and requirements for them for development of the appropriate software. In this case, along with the condition of unambiguous interpretation of specifications on the part of problem suppliers and programmers, the condition of compact notation of the specifications is becoming timely.

The generally accepted method of representing requirements for a program is to reduce the general data processing algorithm in application to technical specifications for programming (GOST [State All-Union Standard] 19.201-78). The description of such an algorithm includes not only the sequence of operations on data, but also the types of information media used (manual documents, punched cards, magnetic tape, and disks) and identifiers of the data structures placed on these media.

According to GOST 19.003-80 the diagrams of algorithms and programs are described in flowchart language, which graphically reflects the structure of simple algorithms. The principal drawback of this language is its unwieldly system of symbols.

The graphic R-language does not have the shortcomings noted in flowchart language [1]. From the standpoint of describing complex algorithms the greatest advantages of R-language are the compactness and brevity of its expressive means. Compared to flowchart language the arcs in R-language carry a double load: description of the structure of the algorithm and description of processes. The succession among processes is insured by the proximity of the corresponding arcs of the R-graph, while simplicity of assimilation is insured by the simplicity of the corresponding conceptions of R-language.

It should be noted that representing algorithms by means of the graphic R-language is close to Mili's universal theoretical model of a finite automaton, which is applicable for objects of any nature [2]. Unlike the finite automatan model graphic R-language does not explicitly fix the states of data at the output

of the process, which creates some inconvenience for the one who states the problems. When designing the general algorithm of a problem it is also convenient for the one formulating it to fix the structures of intermediate data and their states for a check example. Description of the composition and structures at the input and output of processes makes it possible to avoid excessive detail in algorithms. The process of converting the data structure and composition is assigned implicitly by means of such a description. The only additional thing necessary is to indicate conversions performed on quantities of data.

The present article proposes to elaborate R-language to a language of specifications for describing the general algorithms of ASU problems.

Experience with the use of the graphic R-language in the stage of designing algorithms for ASU problems demonstrated its high efficiency. The compactness of description offers great advantages. The diagram of an algorithm described in R-language has roughly the same relationship to the flowchart of an algorithm as a mathematical formula has to its textual interpretation. R-language increased the graphic quality of complex algorithms and made it possible to allocate them in a limited space, which makes it possible to trace the interaction of different modules and distribute processes in an optimal fashion when running the program on multiprocessor complexes.

The analogy between the graphic R-language, where arcs correspond to processes, and critical path schedules, where lines correspond to jobs, greatly simplified the planning and monitoring of program development. Program segments that have been written and debugged are appropriately recorded directly on the diagram of the algorithm.

There was an improvement in mutual understanding between the one who formulated the problem and the program developers, as well as better interaction among the developers during development of the system. With a good picture of the problem, the programmer at the same time has adequate freedom to realize different alternatives of optimal programs.

The volume of documentation in the program design stage was reduced significantly. The graphic quality of the general algorithm of the problem written in R-language made it possible to identify significant design errors in the early stage of design development.

The above allows us to conclude that the use of R-language is promising not only for its basic purpose, but also as a language of specifications in the design of complex ASU programs.

11,176
CSO: 1863/139

UDC 681.3.06.069

ORGANIZATION OF MULTILEVEL RECURSION IN FORTRAN PROGRAMS

Kiev UPRAVLYAYUSHCHIYE SISTEMY I MASHINY in Russian No 1, Jan-Feb 82 (manuscript received 2 Jul 80, after revision 29 Dec 80) pp 15-18

[Article by Aleksandr Borisovich Livshits, senior research associate, Zaporozh'ye Pedagogical Institute, and Viktor Vladimirovich Gorbachev, engineer, Zaporozh'skaya Oblast Clinical Hospital]

[Excerpt] Problems with recursive structure are often encountered when solving various engineering, mathematical, and optimization problems. The area of practical application of recursion is very broad. It includes problems of numerical analysis, translation algorithms, and list processing which are essential apparatus for the development of modern ASU's [automated control systems].

Recursive programming methods, which reflect a characteristic feature of abstract thinking, are practically irreplaceable in a number of cases. The apparatus of recursion is contemplated in most programming languages which have appeared since ALGOL. But of course, the algorithmic language FORTRAN and a number of languages similar to it that are widely used in scientific-technical calculations, do not have their own means of organizing recursions. This often imposes undesirable limitations on the structure of the program complexes being designed.

Work [1] gives a fairly complete presentation of the recursive methods used in programming as well as the means of realizing recursion. But the methodology for realization of recursion in the FORTRAN/FAT system as described in [1] is based on the features of this system's translator, which differs from the YeS computer FORTRAN and, therefore, cannot be used in practice. The fundamental principles of organization of recursion in programs, not oriented to the specific features of languages, are presented in [2]. Works [1, 3] consider chiefly the formal relationship between recursion and iteration. Work [3] analyzes constructions and gives examples typical for the language ALGOL-60. At the present time there is no methodology for constructing recursive programs in FORTRAN oriented to the YeS FORTRAN system. The apparatus for realization of recursion in FORTRAN without using interfacing with programs in assembler language, as presented in [4], has limited application capabilities because of the rules of FORTRAN.

This article will propose a methodology for organizing multilevel recursion for programs realized in the YeS FORTRAN language. The methodology includes two

different approaches. The first is based on interfacing recursive subroutines and subsidiary subroutines in the assembler. The second method is based on the possibility of creating iterative programs directly on the basis of recursively described statements of problems without converting them first. The proposed methods are also applicable in principle to other programming languages which do not have their own means for realization of recursion, and they can be used for programming with other operating systems that have the necessary capabilities.

11,176
CSO: 1863/139

UDC 681.3.06.069

REALIZATION OF RECURSION IN ALGORITHMIC LANGUAGE FORTRAN

Kiev UPRAVLYAYUSHCHIYE SISTEMY I MASHINY in Russian No 1, Jan-Feb 82 (manuscript received 9 Jan 81) pp 18-19

[Article by Nikolay Aleksandrovich Zinchuk, engineer, Kiev, and Vasiliy Vasil'yevich Khristosenko, engineer, Kiev]

[Text] Many problems which are described recursively occur in practice [1]. The use of recursion in programming produces tangible results when solving this type of problems on computers. Despite the fact that recursion is an efficient method of programming, by no means all algorithmic languages (and this includes FORTRAN) have means for the realization of recursion. Therefore, to realize recursion in FORTRAN it is necessary to supplement it with certain subroutines written in another language (for example in assembler language).

Investigators have already done some work in this direction. A subroutine structure in FORTRAN has been found which makes it possible to refer to oneself recursively [2]. The structure of the subroutine is such that it can only be used once. After it finishes its work the recursive procedure transfers control not to the program which calls it for the first time, but rather to the operating system which stops execution of the particular program. The recursive procedure is almost always a part of some program which can call on it more than once [1, 3]. It is also possible that the particular recursive procedure calls up other recursive procedures (for example, realization of syntactic analysis can be translated by the method of recursive discharge and finding the largest common divisor of polynomials) [3, 4].

It is impossible to realize this type of recursion in FORTRAN without additional means. When accessing a subroutine to oneself in FORTRAN its connection with the calling program is lost; to be more accurate, the address of the domain where the calling program is stored is lost. This address is stored in the second word of the storage domain of the subroutine being called, but with recursive access the address of the storage domain of the called subroutine is placed in this word. In order for it to be possible to return to the calling problem, its address must also be stored upon entry into the recursive subroutine and its value restored upon exit from the subroutine. This procedure is accomplished by the subroutines SAVE and RETURN. The SAVE subroutine stores the second word of the storage domain of the recursive subroutine (the address of the storage domain of the calling program) in the SAVE stack. The RETURN subroutine copies the top element of the

stack in the second word of the storage domain of the recursive subroutine. We will write out the structure of the recursive subroutine:

```
      COMMON/STEK11/I,STEK(200)
      M=0
      I=0
      .
      .
      .
      CALL A(M,A1,A2,...,AN,A)
      .
      .
      .
      STOP
      END
```

```
      SUBROUTINE A(M,A1,A2,...,AN,B)
      COMMON/STEK11/I,STEK(200)
      IF(M.EQ.I)CALL SAVE

C     ИЗМЕНЕНИЕ I
C     НЕОБХОДИМЫЕ ВЫЧИСЛЕНИЯ
C     ЗАГРУЗКА СТЕКА ПАРАМЕТРАМИ

      CALL B(M,A1,A2,...,AN,B)

C     ВОССТАНОВЛЕНИЕ I
C     ВОССТАНОВЛЕНИЕ ПАРАМЕТРОВ ПО СОДЕРЖИ-
C     МОМУ СТЕКА

      CALL RETURN
      RETURN
      END
```

The same array is used as the stack in all recursive subroutines. This array is assigned by the general domain STEKII. M indicates the beginning of that part of the stack which is used by the given recursive subroutine; I is the stack indicator; A_1, A_2, ..., AN are the parameters transmitted; B is the parameter which receives the value of the initial address of subroutine A when control is transferred to recursive subroutine A (it is necessary for the subroutine to realize access to itself). This method of realizing recursion is achieved in the YeS [Unified System] disk operating system on the YeS 1033 computer using the FORTRAN-IV translator, version 2.2.

Unlike the method of realizing recursion considered in work [2], this method makes it possible to call recursive subroutines many times and also allows recursive subroutines to include one another, in other words a recursive subroutine can call another recursive subroutine.

11,176
CSO: 1863/139

UDC 681.3.43

DATA CONTROL IN OS-NTs-1 SYSTEM

Kiev UPRAVLYAYUSHCHIYE SISTEMY I MASHINY in Russian No 1, Jan-Feb 82 (manuscript received 20 Nov 79, after revision 28 Apr 80) pp 53-57

[Article by Yuriy Abramovich Basin, engineer, Special Design Bureau of Computer Technology of the Pskov Radio Equipment Plant]

[Text] Introduction

Interest in the development of methods of structural organization of data arises chiefly because of the significant efforts expended by the developers of applied systems to create data bases in information and information-control systems. The construction of iterative systems and artificial intellect systems also involves access to information in an on-line regime. Moreover, the requirement of dynamic change in data links naturally arises in such systems [1].

Works [2, 3] describe the MULTICS system, one of the first in which a significant share of these questions are resolved.

The present article describes a hierarchical structure of files that insures flexible use of the OS-NTs-1 [4] in composing complex data structures. The principal conclusions in the article define the logical organization of files, do not depend on machinery and external memory units, and do not involve physical arrangement of data on information media.

In designing the data control system the following requirements that the system must meet were formulated:

- the files must not be a separate information addressing mechanism, but must define the logical organization of the segmented space of the virtual memory of the system;
- the structural organization of the files should be assigned to the operating system;
- the hierarchical structure of the files should be parametrically adjusted to the required number of levels and permit arbitrary variation (reduction or increase) in the number of levels during the process of developing data bases;

- within the framework of the hierarchical structure of files the system should insure the organization of random links among levels, authorizing the grid structures of files (not necessarily hierarchical);

- the user should address files by their symbolic names. The physical addressing of files should be done by the system depending on the actual position of the files; it should be concealed from the user;

- when there is a right of access, the file should be easily accessible to different users.

The Structure of the Files

This structure is the basic branching (tree-like), hierarchical structure on which additional links can be imposed to insure simple access to the files from any place [2]. Works [5, 6] give a definition of the branching structure and related concepts.

In this article we are interested in the concepts of "depth of search on the tree" and "branching of the nodes of the tree" because representing them in the form of parameters taken together with the recursive nature of trees makes it possible to construct random hierarchical structures.

In fact, if the depth of the search on the tree is defined as the number of levels (n) from the root of the tree to the assigned level and the recursive definition of the tree is used, then this parameter acts as a delimiter of recursion of passage by levels of the tree structure, in other words, when the value of the parameter n is given it becomes possible to operate with trees of the required search depth. In the present version of the system the maximum value of the parameter n is 255, although in principle the parameter has no limits.

A node of the tree is a special file which is supported by the file control system and contains a list of file descriptors. Each such descriptor can address a node at a lower level of the tree. If the branching of tree nodes is defined as the number of lower-level (on the tree hierarchy) nodes generated, then the corresponding parameter will be determined by the capacity of the file which describes the node of the tree. But because the file is a set of a random number of segments, it becomes possible to construct trees with unlimited branching.

The file in the system is identified by its own name. The name of the file is a list of subnames that reflect the position of the file in the tree structure relative to the root of the tree. Each subname identifies the corresponding node of the tree and is called by the node name. The list of node names in the name of the file is called the file of the path and determines the direction of movement on the structure of the tree from the root to the node that addresses the given file. The list is quantitatively linked with the value of the parameter of depth of search on the tree, that is, it includes as many subnames as there are nodes from the root of the tree to the node under consideration. The name

of the path should be unique for the entire tree structure. The condition of uniqueness must be met for node names only in the particular direction on the tree.

The file descriptor contains the name of the node and two types of indicators: the node indicator of a lower level on the hierarchy of the tree, and the indicator of the information file of the particular level. Thus, the node name identifies both its own file and the segment of the path passing through the particular node. This is convenient, for example, in data bases that organize a hierarchy of concepts when the same name must be used to identify both concrete information that describes the given concept itself and abstract concepts defined by means of the given one. If the node name is a terminal subname in the name of the path, then the node name, like the path name, identifies a file. If this is not the case, it identifies the node of the next level on the tree hierarchy, that is, it determines direction of movement on the tree. The fact that the descriptor contains two types of indicators assures independent addressing of both files and nodes.

It follows from this that the descriptor at any moment may contain either both indicators or each of them separately. In this case for the descriptor to exist it is necessary that it define at least one indicator. The set of file descriptors of one level n with one preceding node of the (n-1) level forms the catalogue of files of level n corresponding to the node of the n level which is realized by a special file. The appearance of a new branch in the node of the n level involves including a descriptor of the file of level (n+1) in the corresponding file catalogue of level n. When the current segment is overfilled a new segment is added to the file that realizes the node.

Segments within a file form a linear structure and are additionally identified by their own number. As a result, access to individual segments of the file is index-sequential access.

The presence of just one node indicator in the file descriptor insures the structure branching which follows directly from the definition of the tree.

In the system under consideration the grid structure [5] is insured by imposing additional links among nodes on the base tree structure. With the imposed links the tree structure becomes a directed graph.

The links make it possible to address files from any place in the hierarchical structure, giving the illusion that each link actually exists. Hierarchical subordination is not mandatory for the links. The links make it possible to avoid duplication of files shared by several directions.

The file indicator of the descriptor, not the node indicator, is used to realize additional links because the node indicator insures movement only along the structural links of the tree. But the file indicator addresses only terminal elements of the tree (leaves) which by themselves do not insure any movement. When links are organized such a file contains the name of the path along the tree and the coordinates of the required file, which make it possible for the system to choose a new direction of movement.

All actions related to change in direction of movement on the structure of the tree are performed automatically by the system if during movement in the assigned direction a file is encountered that indicates the need to change direction. We will call the file that insures linkage of different directions on the tree the linking file.

From the individual elements placed in different directions of the branching structure a chain of links can be compiled which is possible when the file addressed by the linking file is itself a linking file. The path along the chain of links will be followed by the system without intervention by the user. The chain can be entered from any of the directions linked to it.

Closed chains and loops, which in the current redaction of the system lead to cycling, are a limitation on the construction of chains of links. This limitation may be removed if the process of movement along a closed chain or loop is accompanied by analysis of the exit condition from the cycle, that is, entering the user processing process. Access to a file on the chain of links can be accomplished either directly by the coordinates of the file or by its path name on the tree. If the file with which a link is established is defined, access to it can be accomplished by coordinates. When the file is not defined, only its path name on the tree by which access is accomplished is recorded in the linking file.

Instructions

Whereas the primary links of the hierarchical branch and structure are formed by the file control system, the supplementary links that provide the grid structure are established by the user using the instruction LINK [SVYaZ'].

In the system under consideration the files are essentially open. This means, in the first place, that the files are equally accessible to all users, and in the second place that any user is authorized for any set of operations on files in the on-line regime with the corresponding right of access. As a result, there is no need for such standard instructions as opening and closing the file.

The set of instructions given to the user will work only on files. Actions on nodes and the structure are performed by the system as accompanying actions to performance of actions on the files, and they are not accessible to the user. For example, on the instruction WRITE [ZAPIS'] a segment or batch of data is written in the file of the given level. In this case the system forms all nodes that are absent in the prestructure and establishes hierarchical links to insure the direction of movement on the tree given in the instruction. On the instruction CANCEL [ANNULIROVANIYe] individual segments of files, files, and subtrees of files may be cancelled, depending on the modification. The system traces the state of the nodes of the tree, corrects links, and where necessary cancels files that realize nodes.

According to the definition in work [6], the tree is a finite set T consisting of one or more nodes. In this case one specially designated node is called the root of the particular tree, and the other nodes (except the root) are contained in $m > 0$ paired nonintersecting sets $T'_1, \ldots, T'_m (r = \overline{1, n})$, each of which is, in its turn, a tree. The trees $T'_1, \ldots, T'_m$ are called subtrees of the particular root.

The condition of paired nonintersection of the subset $T^r_1, \ldots, T^r_m$ leads, when they are consolidated, to the existence of m branches in the root of the tree at level (r-1), that is, if for any

$$T^r_i, T^r_j (i, j \in [1, m], i \neq j), T^r_i \cap T^r_j = 0,$$

then

$$\bigcup_{i=1}^{m} T^r_i = T^{r-1}(m).$$

For subtrees of different levels l, k, and $l \neq k$, the condition of nonintersection is not mandatory. Therefore, consolidating such sets does not always lead to the addition of new branches to the corresponding roots, that is, if among the pairs

$$T^l_i, T^k_j (i \in [1, m_l], j \in [1, m_k])$$

some are found so that

$$T^l_i \cap T^k_j \neq 0,$$

then

$$T^l \cup T^k = T^l(S), S < m_l + m_k.$$

Based on these properties of the tree and applying the condition of coincidence of node names to test the criterion, the following instructions can be introduced to manipulate the structure of the tree:

- graft subtrees to one another;
- prune subtrees from one another;
- change a node name with a corresponding correction of the structure.

The use of such instructions makes it possible to automate the accumulation of knowledge independently received in the system and to control the branching of nodes of the tree and the depth of search in order to optimize retrieval time.

The set of operating system programs that insures the file organization considered above forms the OS-NTs-1 file control system. This system is developed on virtual memory, and realization of its principal functions is insured by the virtual memory of the system.

The Organization of Virtual Memory and Files

The dimensions of the memory given two processes by the segmented space of virtual memory make it possible not to use files as a separate mechanism for addressing information. For this reason there is no need to make a distinction between files and segments, in other words a file is a segment and all segments are filed. The concept of the "file" is transformed and loses its meaning as specific information in external memory units [2, 7].

In the OS-NTs-1 the file is a set of linear arrays in the form of segments in virtual memory. Access to them is accomplished explicitly by name using

read-write operations or implicitly by assigning standard procedures for access to data. The latter is conditioned on the features of organization of access to the system's virtual memory. The concept of homogeneity of memory includes two aspects: homogeneity of access and homogeneity of control.

The traditionally used method of organizing access to virtual memory is based on representing the mathematical address in processor instructions. By means of special equipment this address is converted into the coordinates of the physical location of the corresponding segment or the name of the segment, which identifies the position of the segment in the file structure. The latter is the key to solving the problem of homogeneous system memory, including files [2, 7], because the segment and the file are given in mutually unique correspondence both from the standpoint of organization of access and from the control standpoint.

When controlling files it is not necessary to use the method of buffering entries, subordinating file behavior in the system to the laws of controlling segments in virtual memory. The capability of automatic replacement of data, which defines the extrapolation feature of control, makes it possible to leave in the primary memory only those segments which are most frequently needed. The segments which are not regularly retrieved can be temporarily replaced by those which are required more often. As a result there is no need for all segments of a program or data to be in primary memory at the same time, and the programmer does not have to decide which segment should be in memory and which should not, and which segment should be replaced when the need for a new segment arises [9].

As a result of the homogeneous representation of memory, users can share files without duplicating information and access can be monitored not only for information in external memory but also in primary memory [2]. On the other hand, the condition of correspondence of files and segments means that the segments must, for their part, reflect the structure of the files. Corresponding to each segment in virtual memory is a descriptor, which is nothing else but a descriptor of a file served by the system. The segment is identified by its name, which is equivalent to the file name; in other words, the name reflects the position of the segment in the hierarchical structure.

Establishing a correspondence between the attributes of file and segment control makes it possible to speak of the homogeneity of memory with respect to control regardless of the organization of access. The condition of homogeneity of memory with respect to organization of access should be considered, based on the designation of the system; in the present version the system is oriented to solving information support problems.

The above-described method of access to data in virtual memory is an address method and requires special measures to organize non-address (associative) access, which is standard for information support problems. Work [10] describes an associative-type virtual memory based on the dynamic method of organizing information arrays [11] and considers the mathematical address as the key to access. This system achieves homogeneity of memory with respect to organization of access. It is possible here to speak of homogeneity of memory with respect to access within a single file [12].

Non-address access offers a corresponding organization of data which forms the internal structure of a segment.

Generally speaking, the internal structure of a segment differs from the organization of segments in virtual memory (which in the particular case represents a hierarchical structure). Therefore, the condition of homogeneity of memory with respect to axis should follow from coordinating both levels of access: external (by the structure of segments in virtual memory) and internal (by the structure of the data of the segment).

Procedures for External and Internal Access

The condition of coordinating access by the external and internal structure of the segments necessitates procedures for external and internal access respectively. Each of the procedures accomplishes non-address (associative) access at its own level of identification of objects. The external access procedure identifies data with a precision to the segment level, that is, objects that are known and defined by the system in virtual memory. The internal access procedure operates with data which are structurally defined by the user.

Both types of procedures are realized with due regard for optimal distribution of available means. This refers to the means of the operating system and the linguistic means offered to the user.

The problem of optimal distribution of either operational or linguistic means of realization must be solved in all stages of system design. The criteria of optimality may be, for example, relative overhead expenditures for the effect achieved and user convenience. In the particular case, it seems advisable to impose the internal access procedure on linguistic means by introducing a set of subroutines or expanding the composition of language statements. The external access procedure should be assigned to the operating system. To organize interaction among both types of procedures an interface is introduced. It is simple both from the standpoint of understanding and use, and from the standpoint of its realization.

The interface between procedures is constructed beginning with the idea that the physical file often represents not one segment, but a set of segments corresponding to the physical limitations of the modules loaded in primary memory. Thus, it is necessary to switch the user process among file segments, creating the illusion that the file is a single segment.

The external access procedure defines the necessity of representing the particular segment, interacting with the internal access procedure of the user process. It is constructed on the basis of the property of periodicity or rate of repetition of processing particular batches of file data. Periodicity of data processing means that the user process within the limits of a segment operates with particular parts of it or the segment as a whole in an identical manner from the standpoint of access control; in this case, the procedure performed by the user process is repeated in the file from segment to segment. If the process as a whole does not meet this requirement, its functional decomposition in the class of problems under consideration is found, the case where each

component of the process represented in the form of a separate process will meet the given requirements.

The possibility of the existence of a decomposition follows from the fact that any process can always be broken into parts, each of which either reads data, writes data, or combines these operations in solving information support problems.

Thus, the user process can be started again from the beginning in working with each successive segment and may be considered as a subprocess of the external access procedure because it develops on a known data base and is initiated by the external access procedure for representing data [4].

The organization of access to data by a system procedure that is external to the user process and reducing work with files to work with segments insures homogeneity of memory with respect to organization of access and control with a precision to the segment level, and in combination with internal access procedures provides full homogeneity of memory.

The fact that the external access procedure is external in relation to the user process makes it possible to organize interaction of arrays within the limits of a single file and the limits of arrays belonging to different files. In this case access to arrays is accomplished by the external access procedure and their interaction and processing are accomplished by a user process.

The many different forms of access to data and their interactions lead to the organization of a standard set of external access procedures organized at the level of the operating system. In the OS-NTs-1 data are correlated with the procedure for access to them on the level of initiation processes [4].

Thus, introducing external access procedures in the operating system makes it possible to decompose applied information and information-control systems into separate components, correlating each of the components with an existing set of external access procedures. This simplifies system design because the available set of procedures makes it possible to identify the components of the system related to data processing and data access more clearly and precisely, reduces the time required, and simplifies debugging of software because the functions of access to data and organization of their interaction are performed by the system. The user concentrates his efforts mainly on solving the problems themselves.

Conclusion

From the user standpoint the system described above represents practically unlimited virtual memory structurally organized in the form of a tree with the possibility of adding additional links that construct grid structures with automatic change in the direction of movement. The organization of access to segments insures the use of associative means of access to particular elements, for example, those described in work [10]. Access to data, structural organization of data, and the organization of data interaction are accomplished by the system without user intervention.

From the standpoint of the operating system the memory organization considered here provides standardization and homogeneity of access and control of segments and the entire system memory. The proposed organization of memory does not require additional hardware. At the same time, it can easily be adapted without any substantial additional work if an associative memory unit of adequate capacity is included in the system hardware. This greatly improves the time and functional characteristics of the external and internal access procedures and makes it possible to insure efficient realization of internal access procedures with the means of the operating system.

FOOTNOTES

1. V. M. Glushkov, "The Information Processing Industry," UPRAVLYAYUSHCHIYE SISTEMY I MASHINY, 1977, No 5, pp 3-6.

2. A. Bensoussan, C. P. Clinden, and R. C. Daley, "The Multics Virtual Memory: Concepts and Design," COMMUNS ACM, 1972, vol 15, No 5, pp 308-318.

3. R. C. Daley, and P. G. Neuman, "General-Purpose File Systems for Secondary Storage," PROC. AFIPS, 1965, vol 27, pp 213-219.

4. Yu. A. Basin, "The Architecture of the OS-NPs-1 System," UPRAVLYAYUSHCHIYE SISTEMY I MASHINY, 1979, No 3, pp 55-57.

5. J. Martin, "Organizatsiya Baz Dannykh v Vychislitel'nykh Sistemakh" [The Organization of Data Bases in Computer Systems], Moscow, "Mir", 1978, 616 pages.

6. D. Knut, "Iskusstvo Programmirovaniya dlya EVM" [The Art of Programming for the Computer], Moscow, "Mir", 1976, Vol 1, 380 pages.

7. R. C. Daley, and J. B. Dennis, "Virtual Memory, Processes and Sharing in MULTICS," COMMUNS ACM, 1968, Vol 11, No 5, pp 306-312.

8. J. Donovan, "Sistemnoye Programmirovaniye" [System Programming], Moscow, "Mir", 1975, 540 pages.

9. T. D. Atkinson, U. O. Galgliardi, G. Raviola, and C. S. Schwenk, "The Architecture of Contemporary Central Processors," TIIER, 1977, No 6, pp 37-46.

10. S. Ya. Berkovich, Yu. Ya. Kochin, and Yu. N. Khrebtov, "Principles of Organization of Associative-Type Virtual Memory," PROGRAMMIROVANIYE, 1976, No 6, pp 51-60.

11. S. Ya. Berkovich, "Machine Organization of a Growing Access Tree," DOKL. AN SSSR, 1972, Vol 202, No 2, pp 115-119.

12. U. O. Galgliardi, "Trends in the Development of Computer System Architecture," TIIER, 1977, No 6, pp 31-36.

11,176
CSO: 1863/139

UDC 681.3.06:621.317.799

PROGRAMMING SYSTEM WITH INTERPRETER FOR DIGITAL INSTRUMENTS WITH BUILT-IN MICROPROCESSORS (MICROCOMPUTERS)

Kiev UPRAVLYAYUSHCHIYE SISTEMY I MASHINY in Russian No 1, Jan-Feb 82 (manuscript received 16 Dec 80, after revision 11 May 81) pp 76-78

[Article by Valeriy Viktorovich Barashenkov, doctor of technical sciences, LETI (Leningrad Electrotechnical Institute imeni V. I. Ul'yanov), Tat'yana Gennad'yevna Bakhareva, graduate student, LETI, Nadezhda Tsanzhiyevna Bil'gayeva, graduate student, LETI, Aleksandr Filippovich Kazak, candidate of technical sciences, LETI, Mikhail Naumovich Kuperman, engineer, Leningrad, Aleksandr Orestovich Timofeyev, candidate of technical sciences, LETI, and Aleksandr Timofeyevich Trifonov, engineer, Leningrad]

[Excerpts] The programming system developed by the authors while designing software for voltage pulse generators and word generators with built-in microprocessors is described below [2]. The programming system takes account of the basic feature of this class of instruments, which is the fact that the built-in microcomputer does not participate in the generation process but rather executes user commands arriving from the keyboard or general-use channel, monitors the correctness of actions by the user, shapes data for display, adjusts particular units of the instrument, monitors their work, and controls data exchange with the general-use channel.

Up to one second is allocated to process user commands. The low speed requirements permit internal interpretation of a high-level language in the instrument. This language "compresses" the program and the control memory, which stores the program in the high-level language and the interpreter, has a smaller volume than the case where it stores the program in machine codes (of course, if the volume of the program in machine codes exceeds a certain threshold value).

The problem orientation of the programming system was determined in the process of analyzing algorithms and the first programs for voltage pulse generators and for word generators. The analysis involved identifying the operations encountered and determined the frequency of their appearance. A list of high-level language instructions for the SPRINT (System of PRogramming with INTerpreter) was formed as the result of the analysis [3]. Because the work algorithms of the generators have a clearly expressed control character, the SPRINT language is distinguished by elaborate instructions for controlling transfers in the program. Another characteristic of the SPRINT language is the possibility of describing

procedures for processing binary words of variable length. This is closely linked to the heterogeneity of memory structure in digital instruments.

The software of the programming system that has been described was written in the codes of the Elektronika S5-01 microcomputer, which makes it possible to use this system to develop software for measurement instruments with built-in Elektronika-S5 microcomputers [4]. The volume of the system is 17,000 bytes. Translation speed averages 220 input-language instructions a minute.

Let us give an example of the use efficiency of this programming system. Development of a program in machine codes for one generator with an Elektronika-S5 microcomputer required six worker-months, and the program took about 6,000 bytes. The equivalent program in SPRINT language was written and debugged in two worker-months and with its INTERPRETER proram has a volume of about 4,000 bytes.

11,176
CSO: 1863/139

UDC 681.3.06

SIMPLE GRAPHIC PROGRAMMING IN GRAS SYSTEM

Kiev UPRAVLYAYUSHCHIYE SISTEMY I MASHINY in Russian No 1, Jan-Feb 82 (manuscript received 19 Jun 81) pp 110-113

[Article by Mikhail Aleksandrovich Kurilov, engineer, Institute of Cybernetics of the Ukrainian SSR Academy of Sciences, Kiev]

[Excerpt] One of the fundamental problems in machine graphics today is standardization of the representation of graphic information and functions performed at different levels by graphic packages. If machine graphics are intended for use in a computer network, problems arise related to the form (protocol) for transferring graphic information and to optimal distribution of functions between the terminal graphic processor (for example the automated work position of a designer) and a large processing computer. The purpose of standardization is to insure that graphic programs can be transferred and to greatly simplify graphic programming, that is, to bring us closer to the time when the ultimate user will be able to express his or her graphic intention in programs in fairly simple and natural form.

The Institute of Cybernetics of the Ukrainian SSR Academy of Sciences, working on writing and elaborating software for the system of the Ukrainian SSR Academy of Sciences' Computing Center, has developed the GRAS standard graphic package (graphic standard) [1]. It satisfies international recommendations for standardization of machine graphics programs ACM/SIGGRAPH [2, 3]. The GRAS functions within the YeS operating system and provides access from programs written in high-level languages (FORTRAN, PL-1) to the following graphic units:

- YeS 7051, YeS 7052, YeS 7053, and YeS 7054 graph plotters;
- YeS 7064 graphic display;
- YeS 7066 alphanumeric display.

11,176
CSO: 1863/139

PRINCIPAL DIRECTIONS OF DEVELOPMENT OF SOFTWARE FOR COMPUTERS AND COMPUTER COMPLEXES AND NETWORKS

Kiev UPRAVLYAYUSHCHIYE SISTEMY I MASHINY in Russian No 1, Jan-Feb 82 p 126

[Article by Viktor Nikolayevich Tsoy, scientific advisor, State Committee for Science and Technology, Moscow]

[Text] On 13-15 October 1981 in Sevastopol' the USSR State Committee for Science and Technology and the Sevastopol' Branch of the RDENTP [expansion unknown] of the Ukrainian SSR Znaniye Society conducted an all-Union conference under the title "Principal Directions of Development of Software for Computers and Computer Complexes and Networks."

The purpose of the conference was to summarize practices, development, and application of software for computers and computer complexes and networks and define the directions of its development.

It was noted at the conference that definite advances have been made in our country in building and using computer equipment. In conformity with the resolutions of the 25th and 26th CPSU congresses and the Comprehensive Program for Development, Production, and Application of Computer Equipment in the National Economy until 1990, 3,000 automated control systems and 2,000 computing centers, including seven collective-use computing centers, have been set up and are operating successfully and experimental research is underway on the network of interacting computing centers which joins computers located in Moscow, Riga, and Kiev.

Scientists and specialists have achieved good results in formulating and developing the scientific and practical foundations of programming and in development of software. During the 10th Five-Year Plan many questions of building and using the country's computing potential were resolved. The national economy received more sophisticated hardware and software: the YeS [Unified System] Ryad-2 series of computers, the SM computers, Elektronika class microcomputers, and others. Work was begun to formulate the software of computing complexes and experimental networks of computers.

The State Committee for Science and Technology, the USSR Academy of Sciences, the State Committee for Standards, and other interested organizations are working to formulate uniform technical policy in many areas of the development of software. A number of directive and normative-technical documents on software have been

prepared and published. Specifically, in 1974-1975 the State Committee for Science and Technology published decrees on the standard composition of computer software. This decree envisioned mandatory coordination with the State Committee for Science and Technology of technical specifications for computers and their software. In 1977-1978 standards were ratified for the algorithmic languages ALGAMS, FORTRAN, and COBOL. In 1979 the statute on GosFAP [expansion unknown] was ratified and serves as the organizational and legal foundation of the functioning of this body. In 1979 the State Committee for Science and Technology, jointly with the State Committee for Standards, published a decree according to which all technical specifications for development of software are coordinated with the State Committee for Science and Technology. This procedure is mandatory for all ministries and departments. In 1979-1981 the State Committee for Science and Technology published the following decrees:

- on acceleration of the introduction of technological programming complexes in the national economy (the PRIZ, RTK, and TKP systems were recommended for use on YeS computers);

- on accelerated introduction in the national economy of the software means of the data bank control systems OKA, SIOD-3OS, BANK-OS, SETOR, SEDAN, and DIAMS, as well as the information retrieval systems POISK-1, POISK-4, and ASPID-3;

- on programming languages for minicomputers and microcomputers (FORTRAN, COBOL, BASIC, and PASKAL' were declared mandatory for minicomputers and microcomputers).

More than 30 State All-Union Standards have been worked out and introduced in the software area.

Substantial attention was devoted to discussing the problems of developing collective-access systems, interactive information systems, software for subscriber stations, terminal processors, methods of organizing local distributed data bases, and problems of improving control of the computing process.

Subjects related to work on development of software for collective-use computing centers occupied a significant place. A number of reports were devoted to the development of operating systems for series-produced YeS and SM computers, improving their efficiency, development of software for putting together complexes of YeS and SM computers, and so on.

The conference took note of original domestic developments in the field of software with no analogs in foreign practice and the development of operating systems with messages in the Russian language as positive phenomena.

The resolution adopted by participants at the conference defined the principal directions of development of software for computers and computer complexes and networks:

- development of software for new models of computers (the YeS Ryad-3, the second-phase SM, the El'brus microcomputer,

Elektronika class minicomputers and microcomputers, the recursive computer, the modular asynchronous computing system Mars, and others), including development of new operating systems, translators, software for remote network processing and special processors, and interactive means of data processing with remote and local access;

- development of software for distributed multimachine complexes for collective-use computing centers and networks of computers based on the YeS Ryad-2 and Ryad-3 machines, and development of means for forming complexes of YeS models with SM models;

- increasing the productivity of collectives of programmers, putting together new technological programming complexes and introducing those which have already been developed, development of programming means, development of programming systems for new programming languages, and improving the organization of programming;

- development of work on control systems for data banks and setting up data banks and information retrieval systems for specific fields of knowledge, science, and technology;

- formulating packages of general-purpose applied programs for YeS and SM computers, microcomputers, and computing complexes; producing, introducing, and escorting these packages as industrially produced articles;

- development of standardization of computer equipment generally and of software specifically;

- pure research and scientific investigation in the following areas: the architecture of programs and data; microprogramming; quality and reliability; automated development and debugging; program correctness; parallel computation; and so on;

- organizational-technical, economic, and legal questions of improving the activities of GosFAP;

- improving the methods and organizational forms for raising the qualifications of programmers and training them.

11,176
CSO: 1863/139

UDC 681.322

STRUCTURAL METHODS OF REDUCING EFFECTIVE CYCLE OF TWO-LEVEL MAIN STORAGE IN MULTIPROCESSOR SYSTEMS

Kiev UPRAVLYAYUSHCHIYE SISTEMY I MASHINY in Russian No 2, Mar-Apr 82 (manuscript received 28 Oct 80, after revision 25 Mar 81) pp 22-25

[Article by Anatoliy Il'ich Slutskin, engineer; Yuriy Petrovich Tsukanov, engineer; and Nikolay Mikhaylovich Sharunenko, candidate of engineering science; all from NITsEVT [Scientific Research Center for Electronic Computer Engineering], Moscow]

[Excerpts] To surmount the existing gap (approximately by an order of magnitude) in long cycles of processor operation and of the main storage (OP) access cycle, a cache type high-speed buffer, the speed of which is commensurate with the processor operation rate, is usually made part of modern high-performance computers.

All the given approaches to the discipline of high-speed buffer--main storage access fall short when used in multiprocessor computers. We propose a new algorithm called the "fully asynchronous flagged swap."

The essence of this algorithm is as follows. When the information requested by a processor is in high-speed buffer storage, then irrespective of the operation type (read or write), main storage is not accessed, and the data are read or written only in high-speed buffer storage [HSBS]. When the information is not in HSBS, the processor request information--main storage address, operation code, information to be written (for a write operation), mathematical number of a processor and other identifying information--is buffered in a special functional unit (let us call it an adapter for communication with processors (ACP)). The ACP organizes a queue of requests from processors to main storage for reading. After reading an information block from main storage, the ACP polls the priority scheme for the HSBS, after first writing (for a write operation) the information from a processor to the information block read from main storage. An inactive information block from HSBS (if its change bit equals one) is first moved from HSBS to special registers making up a data rewrite unit (BPD), and then the information block from the ACP is put into the high-speed buffer storage. The data rewrite unit organizes a queue of information blocks in main storage to be replaced. Fig. 3 [not reproduced] shows the flowchart for the "fully asynchronous flagged swap" algorithm.

In contrast to the algorithms for WT [write through], SS [simple swap], FS [flagged swap] and FRS [flagged registered swap], the suggested algorithm permits asynchronous operation of the high-speed buffer and the main storage since during operations to move a new information block from main to high-speed buffer storage and to rewrite an information block being replaced to main storage, the high-speed buffer is free and can service requests from other processors.

So, even though some additional hardware is needed to implement it, the "fully asynchronous flagged swap" algorithm is not inferior to FRS, the most efficient of the known algorithms for single-processor systems, and eliminates the shortcomings of the algorithms listed (from the viewpoint of minimizing T_{ef} [effective access cycle time]) in building multiprocessor systems.

8545
CSO: 1863/179

UDC 681.323:518.5

APPROACH TO DESIGNING MULTIMICROPROCESSOR SYSTEMS TO SOLVE ORDINARY DIFFERENTIAL EQUATIONS

Kiev UPRAVLYAYUSHCHIYE SISTEMY I MASHINY in Russian No 2, Mar-Apr 82 (manuscript received 13 Feb 81, after revision 31 Jul 81) pp 42-45

[Article by Vitaliy Petrovich Boyun, candidate of engineering science; Leonid Grigor'yevich Kozlov, candidate of engineering science; and Valeriy Ivanovich Tereshchenko, post-graduate student; all from IK AN USSR [Institute of Cybernetics, Ukrainian SSR Academy of Sciences], Kiev]

[Text] In controlling rapidly flowing processes in real time, simulating such processes and in a number of applied fields of science and technology, there occurs the problem of rapidly solving systems of ordinary differential equations, i.e. high-speed specialized processors are needed to solve these systems of equations [1]. The performance of these processors can be raised by algorithmic methods: special methods of problem solving are employed to reduce the volume of computations and problem solving algorithms are processed concurrently, i.e. special multimicroprocessor systems are built.

Considered in this work is the application of incremental methods that permit reducing the volume of computations in solving differential equations based on known methods of numerical solution of the Cauchy problem; also discussed are some questions of designing multimicroprocessor systems to implement these methods. The application of incremental methods for solving systems of linear algebraic equations is well known [2].

Incremental methods are based on making use of small position (one-position, most often) increments in computations. Computation volume is reduced largely through simplification of the multiply operation which is reduced to a single-cycle operation of shifting the multiplicand by some number of positions instead of r cycles of addition of partial products.

The numerical solution to the Cauchy problem for systems of ordinary, linear, homogeneous differential equations with constant coefficients

$$Y' = AY \tag{1}$$

is found by the methods of Adams, Milno, Nystrem and others [3], which can be represented as

$$Y_{k+1} = Y_k + h \sum_{l=}^{p} b_l Y'_{k-l+1}. \tag{2}$$

The main volume of computations in formula (2) consists in computing the sum of paired products of constant coefficients b_1, defined by the integration formula, and values of derivatives of unknown functions at p points for p step methods of integration.

For a number of methods of numerical solution to the Cauchy problem, for example, for the methods of Adams to the fifth order inclusively, the coefficients from formula (2) can be represented in the form of the product of the power of the pair 2^{m_1} and some number b_0, not a multiple of the pair; then formula (2) assumes the form

$$Y_{k+1} = Y_k + hb_0 \sum_{l=1}^{p} 2^{m_l} Y'_{k-l+1}. \qquad (3)$$

which permits simplifying computation of the indicated sum of paired products by replacing multiply operations by single-cycle shifts.

To design incremental microprocessor systems to solve systems of ordinary differential equations, integration formulas in which increments of derivatives of unknown functions are used are of interest.

To reduce the volume of computations, it is advisable to make use of one-position increments ∇U, equal to the values of the high-order significant positions of the increments of derivatives of unknown functions ΔU. Let us denote the operation of singling out the high-order significant position by $\nabla U = \text{lncr}\ (\Delta U)$. Then from formula (2), let us derive the incremental procedure of integrating differential equations

$$Y_{k+1} = Y_k + h\left(C_0 \cdot Y'_k + \sum_{l=1}^{p} C_l \nabla Y'_{k-l+1}\right), \qquad (4)$$

in which the sum of paired products is replaced by the sum of constant coefficients C_1, shifted by the number of positions defined by the increments of the increments of derivatives of unknown functions at p points.

We have discussed the first part of tht algorithm for the numerical solution to a system of differential equations (1), integration. The second part of the algorithm is computing new values of derivatives of unknown functions and for the incremental algorithm new values of the increments of the derivatives of these functions (computation of the right part of system (1)).

Thus, at the (k+1)-th point of integration, the value of the right part will equal

$$Y'_{k+1} = AY_{k+1}. \qquad (5)$$

If we represent the value of the vector of unknown functions Y_{k+1} by the value of this vector at the k-th point and the vector of increment of unknown functions at the (k+1)-th point, then formula (5) can be transformed for incremental algorithms to the form

$$Y'_{k+1} = Y'_k + \nabla Y'_{k+1}, \quad \Delta Y'_{k+1} = I_{\text{ксч}}\ (A \cdot \Delta Y_k). \qquad (6)$$

Based on the formulas given above, let us now construct several algorithms for the numerical solution to the Cauchy problem for the system of differential equations (1). The flowchart for the first and second (full-position) algorithms is shown in fig. 1, and for the third (incremental) algorithm, in fig. 2.

Let us assess the efficiency of these algorithms. To evaluate efficiency, let us use the number of additions in one cycle of the algorithm (in a step of integration).

Given below is the number of additions in executing the individual statements of the algorithms given above and these algorithms as a whole with the dimension of the matrix of coefficients A equal to n X n = n^2, and the number of positions of the numbers equal to r:

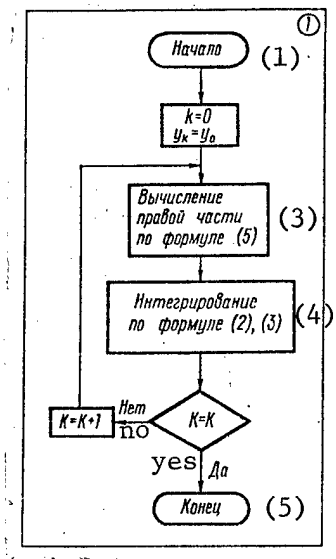

Fig. 1.

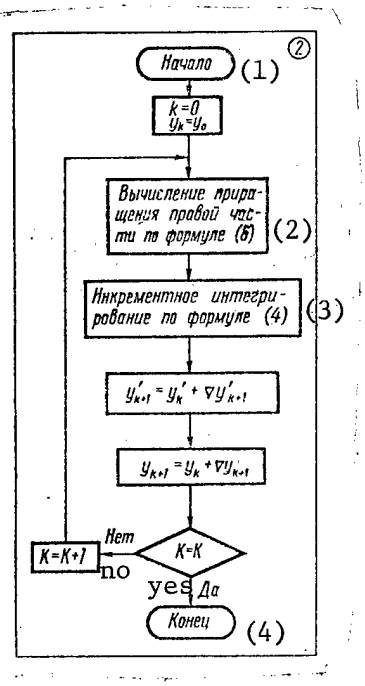

Fig. 2.

Statement	Number of additions
Computation of the right part by formula (5)	$q_1 = n^2(r+1)$
Integration by formula (2)	$q_2 = (p+1)(r+1)$
Integration by formula (3)	$q_3 = 2r+p+1$
Incremental computation of right part by formula (6)	$q_4 = n^2$
Incremental integration by formula (4)	$q_5 = 2r+p(r+1)+1$
First (full-position) algorithm (fig. 1) (integration formula (2))	$Q = q_1 + q_2 = (r+1)(n^2+p+1)$
Second (full-position) algorithm (fig. 1) (integration formula (3))	$Q_2 = q_1+q_3 = n^2(r+1)+2r+p+1$
Third (incremental) algorithm (fig. 2)	$Q_3 = q_4+q_5+2 = n^2+2r+p(r+1)+3$

Let us determine the factor of advantage in computations of the third algorithm relative to the first and second, respectively:

$$L_{31} = \frac{Q_1}{Q_3} = \frac{(r+1)(n^3+p+1)}{n^2+2r+p(r+1)+3},$$

$$L_{32} = \frac{Q_1}{Q_3} = \frac{n^2(r+1)+2r+p+1}{n^2+2r+p(n+1)+3}.$$

Shown in fig. 3 are the plots of the relationship between the advantage factor in computations and the number of positions of the numbers to be processed and the order of the system (1) to be solved. From this it follows that the efficiency of the incremental algorithm is higher than the full-position algorithms and increases as the number of positions in the numbers to be processed r and the order of the system (1) n increase.

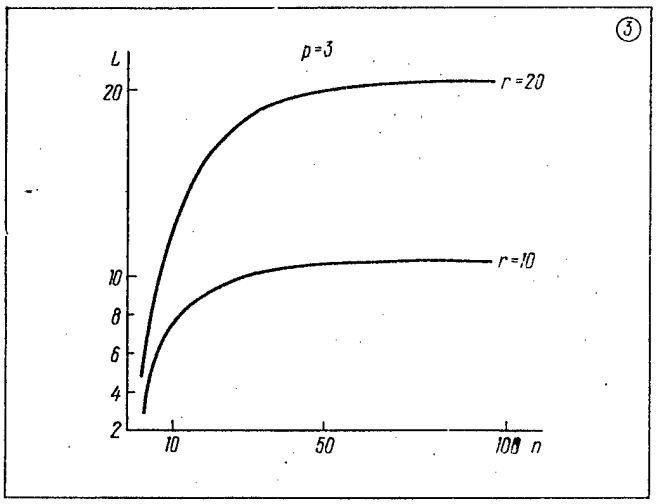

Fig. 3.

Based on existing methods for solving system (1), incremental algorithms for solving this system of differential equations have been constructed and their efficiency demonstrated. Let us design a processor to implement the incremental algorithms for solving the system with general-purpose microprocessors. Let us define the additional requirements that arise in designing this processor. By way of example, let us use the K589 series microprocessor set, with which rather flexible and general-purpose computer structures can be built. But in evaluating the incremental algorithm for solving the system, it should be noted that in designing a processor to solve this system, a unit to identify the high order significant digit of a number and a unit for fast shift by an arbitrary number of positions must be included in it. Incorporating these units will permit performing the operations of identifying the high order significant digit and of a fast shift within one cycle and thereby ensure the high efficiency of the inuremental algorithm.

Another direction for raising the performance of processors is parallel processing of the problem solving algorithm. Let us consider the possibility of parallel processing of the incremental algorithm and designing a multimicroprocessor system.

Fig. 4b shows the structure of the data to be processed by the incremental algorithm. Analyzing the algorithm, let us note that within an algorithm cycle, we can process the data, after dividing it line-by-line into groups (bold lines in fig. 4b) each independently of the other.

Thus, let us divide each file into α groups of β rows in each group. Then in each group, before executing the cycle, the vector of increments of unknown functions at the k-th point ∇Y_k must be of the full dimension, n rows; after execution of a cycle, the dimension of the computed vector of increments of unknown functions at the (k+1)-th point ∇Y_{k+1} in each group will be β rows.

Let the computations be performed by α processors, and let us call them point microprocessors, each of which is a general-purpose or specialized microprocessor or processor built on the base of microprocessor sets. Some requirements for this point microprocessor have already been formulated above.

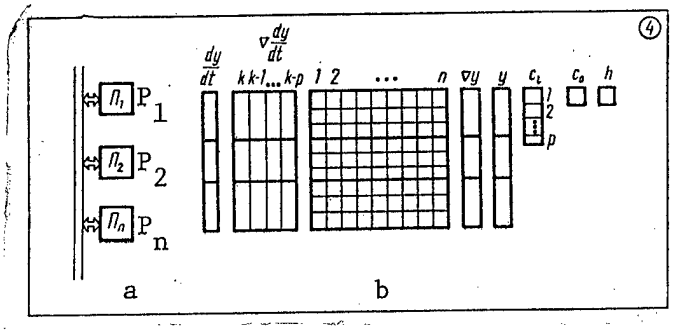

Fig. 4.

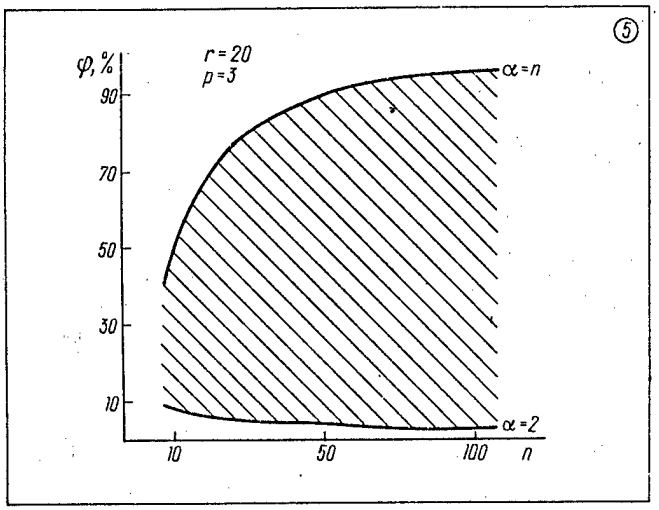

Fig. 5.

Placed in each point microprocessor are the β rows, to be processed by them, of all the arrays, and the vector ∇Y_k is placed in each microprocessor completely. After execution of the algorithm cycle, each α microprocessor contains just β each of the corresponding rows of the vector ∇Y_{k+1}, which in the next cycle must be present in each microprocessor completely. In other words, after each algorithm cycle, it is necessary to transfer from each microprocessor its group of rows of the vector ∇Y_{k+1} to all the other microprocessors. Thus, in each algorithm cycle in a multimicroprocessor system consisting of α microprocessors, an exchange of data is effected between the microprocessors which consists in transferring n rows of vector ∇Y_{k+1}. Since each row of vector ∇Y_{k+1} consists of one element, n words are transferred in each cycle over the buses for exchange in the multimicroprocessor system. Let us note that the number of words n transferred in the system does not depend on the number of point microprocessors α.

The algorithm for exchange between the microprocessors, when each in turn sends information to all the others at once, permits, without reducing the performance of the multimicroprocessor system, connecting the microprocessors to each other by the common bus for exchange. The structural diagram of this multimicroprocessor system is shown in fig. 4a.

Since the number of words sent between the microprocessors does not depend on the number of microprocessors, the speed of this system increases in direct proportion to the increase in number of point microprocessors. The word size of the exchange bus in connection with the transfer between the microprocessors of incremental increments is reduced and equals $r = \log_2 \nabla Y_{max}$, where ∇Y_{max} is the maximum allowed increment of an unknown function. It was shown above that the number of words sent between the microprocessors is constant and equals n, which makes up a small part of all the data to be processed. The total amount of data to be processed, the structure of which is shown in fig. 4b, is $D = n^2 + n(p + 3) + p + 2$ words; then the amount of data sent between the

microprocessors of the total amount of data will be

$$z = \frac{n}{n^2 + n(p + 3) + p + 2} \cdot 100\%.$$

The relation between the time of computations (useful time) and time for exchanges is also of interest. Let us assess this relation. Let the addition of two numbers be performed within some timeslice t, a step, and the transfer of one number from one microprocessor to all others also be performed within one step. Then, after assuming with some error that algorithm cycle time is defined by the sum of additions Q_3, let us derive the algorithm cycle time (time of computations) which equals $T_b = Q_3 t$, and the time of exchanges, $T_0' = nt$. For the incremental multimicroprocessor system in question, as the number of point microprocessors increases, computation time is reduced in inverse proportion to the number of them and will be defined as $T_{bm} = T_b/\alpha$. As it follows from what has been discussed earlier, exchange time does not depend on the number of processors, is constant and equals T_0. Then the relation between computation and exchange time is

$$\varphi = \frac{T_0}{T_{bm}} = \frac{n \cdot \alpha}{n^2 + 2r + p(r + 1) + 3} \cdot 100\%.$$

If $\alpha = 1$, then $T_0 = 0$ (no exchanges) and $\phi = 0$ too. The graph in fig. 5 illustrates what part of computation time in the multimicroprocessor system (T_{bm}) is accounted for by exchange time (T_0). Shown in the graph are characteristics for two extreme versions of multimicroprocessor systems, when the system has just two microprocessors ($\alpha = 2$) or when the system has the maximum number of microprocessors, equal to the order of the system of differential equations to be solved ($\alpha = n$). All other versions of the multimicroprocessor systems in question are described by the interval between the curves $\alpha = 2$ and $\alpha = n$.

It follows from the graph that when the number of microprocessors is small, computation time increases faster than exchange time as the order of the system to be solved grows; and conversely, when there is high concurrency in running the job, computation time increases more slowly than exchange time as the order of the system to be solved grows, and in the process, computation and exchange time tend to equality. Naturally, one can conclude that as the number of microprocessors α increases, exchange time, being a constant quantity for some system order n, reduces the percentage of useful multimicroprocessor system operating time.

Let us now turn to a feature of solving differential equations on the multimicroprocessor system in question. In the majority of real problems, matrix A in system (1) is a sparse matrix (containing zero factors). In solving system (1), the zero factors are not needed; thereforef matrix A is stored in the point microprocessors in compressed form (without the zero factors). This allows eliminating unproductive processing of the zero factors in matrix A and solving a system of differential equations of a higher order with some finite amount of storage.

Algorithm cycle time in the multimicroprocessor system is defined by the maximum operating time in one algorithm cycle of one of the microprocessors, which in turn

is in direct proportion to the number of nonzero factors of matrix A in the group of rows to be processed. Exchange in the system can be begun only after completion of an algorithm cycle in each microprocessor; therefore, idle time of some of the microprocessors in each cycle is possible in the system. To prevtnt it, in preparing the problem, matrix A has to be divided into groups of rows so that the number of nonzero factors is about equal in each group. This allows more uniform loading of each microprocessor and consequently raises the performance of the entire system.

We have discussed incremental algorithms for solving the Cauchy problem for system (1). The cycle of this algorithm is executed about r-fold faster than in full-position algorithms based on known methods. In the process, to achieve the indicated efficiency of incremental algorithms, the specialized processor built on the base of general-purpose microprocessor sets must incorporate units to identify the high-order significant digit of the number and units to shift a number by some arbitrary number of positions. The incremental multimicroprocessor system also allows raising the speed of solving system (1) linearly from the number of point microprocessors. In the process, the amount of data to be exchanged in the system is small, as is the word size of the exchange bus.

The high performance of incremental multimicroprocessor systems allows using them in systems for controlling rapidly flowing processes in real time.

BIBLIOGRAPHY

1. Boyun, V. P. and Kozlov, L. G., "Printsipy postroyeniya problemno-orientirovannykh protsessorov dlya resheniya nauchno-tekhnicheskikh zadach" [Principles of Designing Problem-Oriented Processors to Solve Scientific and Technical Problems], Kiev, RDENTP [expansion unknown], 1978, 22 pages.

2. Malinovskiy, B. N.; Boyun, V. P.; and Kozlov, L. G., "Algorithms for Solving Systems of Linear Algebraic Equations, Oriented to Structural Implementation," USiM, No 5, 1977, pp 79-83.

3. (Kollats, L.), "Numerical Methods of Solving Differential Equations," Moscow, IL [Foreign Literature Publishing House], 1953, 364 pages.

8545
CSO: 1863/179

UDC 680.3

TECHNIQUE AND SOFTWARE PACKAGE FOR INVESTIGATING COMPUTING PROCESS IN UNIFIED SYSTEM OPERATING SYSTEM

Kiev UPRAVLYAYUSHCHIYE SISTEMY I MASHINY in Russian No 2, Mar-Apr 82 (manuscript received 24 Feb 81, after revision 26 Aug 81) pp 67-71

[Article by Vladimir Apollonovich Korotkevich, senior scientific associate; and Ivan Vasil'yevich Maksimey, candidate of engineering sciences; both from GGU [Gomel' State University], Gomel']

[Excerpts] Problems of organizing the computing process (VP) in VTs [computer centers] are continually focused on by researchers and developers of computer software. Interesting results of analysis of organization of work with computers are presented in [1-8]. Proposed in this work is an approach to investigating the computing process with Unified System computers, based on measuring statistics of information streams in solving problems.

A number of authors investigating information streams (IP) in Unified System computers started from, as a rule, standard facilities for recording statistics, that are available in the Unified System operating system [4-7]. Taken as the basis were statistics compiled by the system monitor program (SMP), which were used to generate a set of statistics of interest to the investogator, from which an analysis of the appropriate aspects of organization of the computing process with Unified System computers was made. There is no doubt that the statistics compiled by the SMP are rather abundant, and they can and must be used in analyzing the functioning of YeS OS and in studying the composition and structure of jobs being run. But, statistics on information streams are essentially integral. In a number of cases, especially in studying the properties of YeS OS for special applications, the investigator is interested in the time diagram for execution of job steps both in the single and multi- program modes of YeS OS operation, as well as the time diagram for YeS OS operation. In these cases, developers of special-purpose software for the Unified System computers are in a bind, for there are no measuring facilities to obtain differentiated statistics for execution of job steps. Described in this work is the procedure and software for organizing measurement of information streams in Unified System computers that allows recording the differentiated statistics for job step execution and the structure of the tasks solved with Unified System computers.

Also, a number of researchers of the computing process (VP) in YeS OS get the capability of making use of actual statistics on the computing process to design task generators in planning special-purpose computer complexes [8, 10].

Implementation of the Procedure for Studying the Computer Process

The suggested procedure for studying the computer process has been implemented in the form of a package of interrelated programs for measuring and processing; the programs include both the statistics recorded by the SMP and the results of operation of programs for recording tht statistics of YeS OS functioning at the TsP [central processor = CPU] interrupt level. Fig. 3 shows the makeup and flowchart for the software package interaction. The measuring and processing software package consists of the SMP STATSMF statistical processing program, the STATOS program package for measuring the parameters of the computing process at the level of CPU interruptions, and the SPRINT package of procedures for statistical processing of observation results.

Key:
1. jobs
2. Unified System OS
3. results
4. SMP [system monitor program]
5. SMP statistics
6. job 1 for processing
7. results of analysis
8. statistics on CPU interruptions
9. job 2 for processing
10. information for task generators

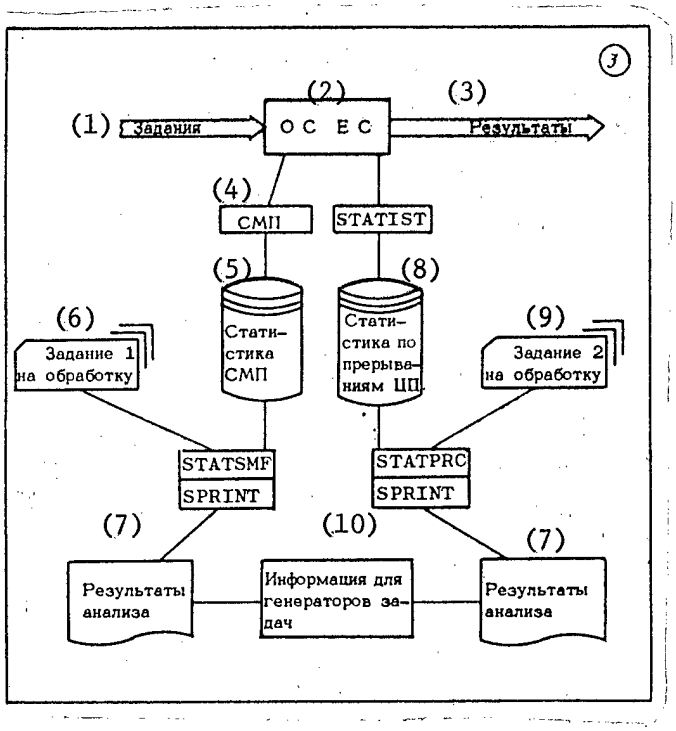

Fig. 3. Interaction of measuring software package

Approbation of Procedures and Introduction of Packages

Experience in operating these software packages has shown that with regular acquisition of statistics, YeS OS performance losses do not exceed five percent in the measuring process.

The software packages were installed in these scientific research organizations: the VTs SO AN SSSR [Computer Center, Siberian Branch of USSR Academy of Sciences] in Novosibirsk, and the NII EVM i ITK AN BSSR [Scientific Research Institute of Computers and the Institute of Engineering Cybernetics, Belorussian SSR Academy of Sciences] in Minsk.

BIBLIOGRAPHY

1. Sergiyenko, I. V., "Concerning the Design of Software for the Computing Process on Computers," KIBERNETIKA, No 2, 1970, pp 108-116.

2. Sergiyenko, I. V., "Metody organizatsii vychislitel'nogo protsessa na vychislitel'nykh mashinakh" [Methods of Organizing the Computing Process on Computers], Kiev, IK AN USSR [Institute of Cybernetics, Ukrainian SSR Academy of Sciences], 1971, 164 pages.

3. Sergiyenko, I. V. and Skopetskiy, V. V., "Problems of Studying Data Processing Systems and Raising the Efficiency of Them," KIBERNETIKA, No 6, 1977, pp 96-100.

4. Pan'shin, B. N.; Il'ves, A. R.; and Snegireva, L. K., "Software Package for Computer Load Accounting," USiM, No 1, 1979, pp 14-17.

5. Vinnichenko, A. I.; Gurin, N. N.; Kogan, Ya. A.; and Lepicheva, N. G., "Experience in Measuring and Adjusting the YeS OS Operating System," PROGRAMMIROVANIYE, No 1, 1979, pp 61-66.

6. Usov, S. A., "Issledovaniye vychislitel'nykh sistem" [Study of Computer Systems], Moscow, IPM AN SSSR [Institute of Applied Mathematics, USSR Academy of Sciences]m 1972, 110 pages.

7. Koshman, Ye. A., "Synthesis of a Statistical Model of the Load in Studying Computing Systems," VOPROSY RADIOELEKTRONIKI. ELEKTRONNO-VYCHISLITEL'NAYA TEKHNIKA, No 7, 1978, pp 95-99.

8. Maksimey, I. V., "Funktsionirovaniye vychislitel'nykh sistem" [Functioning of Computer Systems], Moscow, Sov. radio, 1979, 270 pages.

9. Korotkevich, V. A. and Maksimey, I. V., "On the Formal Representation of the Computing Process in Unified System Computers," in "Vychislitel'nyye sistemy kollektivnogo pol'zovaniya" [Computer Systems for Shared Use], Novosibirsk, 1976, pp 115-121.

10. Maksimey, I. V., "Parametrizatsiya, izmereniye i postroyeniye modeley informatsionnykh potokov v vychislitel'nykh sistemakh: Avtoref. dis. ... kand. tekhn. nauk" [Parametrization, Measurement and Design of Models of Information Streams in Computer Systems: Author's Abstract of Dissertation ... Candidate of Engineering Sciences], Novosibirsk, VTs SO AN SSSR [Computer Center, Siberian Branch of the USSR Academy of Sciences], 1973, 20 pages.

8545
CSO: 1863/179

UDC 681.3.51./6.42

INTERACTIVE GRAPHIC DESIGN SYSTEM IN MULTIACCESS MEASURING-COMPUTER SYSTEM

Kiev UPRAVLYAYUSHCHIYE SISTEMY I MASHINY in Russian No 2, Mar-Apr 82 (manuscript received 1 Oct 80, after revision 28 Apr 81) pp 79-81

[Article by Yuriy Vladimirovich Obukhov, junior scientific associate, and Sergey Aleksandrovich Platonov, junior scientific associate; both from IRE AN SSSR [Institute of Radio Engineering and Electronics, USSR Academy of Sciences], Moscow]

[Excerpts] Described in this work are the structure and functional capabilities of an interactive system for design and editing of graphs that has been developed at the Institute of Radio Engineering and Electronics, USSR Academy of Sciences, for a multiaccess measuring-computer system (IVSKP) based on the NORD-10/S computer network. The system is written in FORTRAN-77 and operates with the SINTRAN-III/VS DOS file system and the PLOT-10 graphic subroutine package.

This system is superimposed on a basic package of graphic subroutines and is an interface between the user input langauge and the graphic subroutine package. It is an interactive system, i.e. all information on the plot, instructions for its design and any changes are specified in the interactive mode.

The graphic design system described became part of the systems software for the multiaccess measuring-computer system at the IRE AN SSSR [Institute of Radio Engineering and Electronics, USSR Academy of Sciences] in 1980; it is operated in the real-time mode and in the mode of designing from data files. The system allows constructing graphs on three types of units:
--oscillographs with a storage tube and X-Y self-recorders controlled by CAMAC modules;
--Tektronix-4662 type laboratory plotters and the Tektronix-4006, 4012 and 4025 graphic displays; and the
--Tektronix-4663 systems plotter.

Operating the system has especially shown the conveniences of operating in the real-time mode since there is no need for preliminary operations in designing graphics (construction of axes, correlating user units to screen coordinates, drawing letters, etc.). Using the graphic system has reduced writing time for programs and simplified their debugging and has reduced design time. The latter circumstance is due to all information on graphic parameters being specified before the start of user program operation, and axis construction and lettering being done before the start of the operation of the time-critical part of the real-time program responsible for acquisition and preprocessing of experimental data. One can state that

with the initiation of the graphic system described, outputting information in graphic form has become for the user no more complex than outputting information in alphanumeric form.

The authors wish to thank V. V. Romanovtsev, A. Ya. Shul'man and A. V. Moshkov for their help in all stages of this work.

8545
CSO: 1863/179

UDC 681.3:62-52

TEMP INSTRUMENTAL COMPLEX FOR SIMULATING EL'BRUS MULTIPROCESSOR COMPUTER COMPLEX ON THE BESM-6

Kiev UPRAVLYAYUSHCHIYE SISTEMY I MASHINY in Russian No 2, Mar-Apr 82 (manuscript received 18 Sep 81) pp 95-97

[Article by Aleksandr Abramovich Gutman, engineer, and Viktor Aleksandrovich Markov, candidate of physicomathematical sciences; both from the Novosibirsk branch of the ITM i VT AN SSSR [Institute of Precision Mechanics and Computer Engineering, USSR Academy of Sciences], Novosibirsk]

[Text] Introduction. TEMP is a complex of system programs on the BESM-6 that simulate the "El'brus-1" MVK [multiprocessor computer complex] developed at the ITM i VT AN SSSR [Insititute of Precision Mechanics and Computer Engineering, USSR Academy of Sciences] [1]. The TEMP complex is intended for designing, programming, debugging and transferring to the MVK the various BESM-6 program products: software, software packages, documentation, etc.

Using the TEMP complex enables compatible development of the hardware and software parts for the "El'brus-1" MVK, including general program support, training personnel to program in the input languages used on the "El'brus-1" MVK and to work with its software; and production programming for the "El'brus-1" MVK by means of the BESM-6, which is especially important in the initial period when there are few MVK installations. In the process, the TEMP complex is capable of meeting the needs of users deprived of regular access to the actual "El'brus-1" MVK installations.

The TEMP complex was developed in the 1977-1979 period at the Novosibirsk branch of the ITM i VT AN SSSR [Institute of Precision Mechanics and Computer Engineering, USSR Academy of Sciences] and is now operated in more than 20 organizations in the country. The total components of the program complex exceeds 100,000 lines in the high-level YaRMO language [2] or about 400,000 BESM-6 instructions. Applied in developing the complex was a number of instrumental facilities as well as the technology, based on them, for design, programming and debugging, which allowed designing and implementing the TEMP complex in a brief period and also solving the problem of compatibility under the conditions of changing specifications.

Capabilities of the TEMP Complex. On the whole, the general capabilities are quite traditional [3]; they offer the user convenient and full-fledged operation with the main components of the complex simulating the "El'brus-1" MVK.

These capabilities can be mentioned:
--assignment of work to the complex (job control language and the monitor that interprets it);
--storage of various files (an archive that allows rather conveniently controlling the storage of information, i.e. access, protection, etc.);
--convenient generation of tasks (task generation system with its own archive and editor);
--convenient method of text input/output and text editing facilities (text editor that supports conventional and context editing, as well as operation with editing sessions);
--facilities for adequate visualization of texts (editor-supported facilities for reflecting structure of programs in a listing, and a documentor that enables qualitative structuring and output of technical documents on a printer or "Optima" typewriter).

The monitor and all other TEMP complex components operate in both the batch and interactive modes.

The TEMP complex contains a translator from "El'brus" autocoder [4], the basic MVK programming language, into MVK codes. The autocoder translator enables rapid translation, qualitative diagnostics of errors, and at the user's option, output of various information on the translated program and its correspondence to the source text (for example, tables of correspondence of lines to the code).

The TEMP complex contains an "El'brus-1" MVK interpreter that executes files of object code for the MVK, obtained in the complex. The interpreter also simulates the basic functions of the MVK operating system that are necessary for program execution. In a dialog, the interpreter can operate in the debugging mode, in which it is possible to halt at specific points in the program and visualize in accordance with the user job the status of computations at these points.

The complex also has translators for ALGOL-60 and standard FORTRAN, which are part of the standard software for the "El'brus-1" MVK. They were written in autocoder and were themselves debugged with the instrumental TEMP complex. Since they are executed in the TEMP complex through the interpreter, their rate on the BESM-6 is low.

Service programs as a whole enable efficient, practical programming in the TEMP complex through facilitating routine work that accompanies programming and debugging. They include the following programs:
--object code file editor; editing is performed in coordinates natural for MVK code (segment number, byte number); the editor enables visualization of code in hexadecimal and in mnemonics;
--autostructurer, which creates a structured listing and performs automatic editing of a program in autocoder, after which clearly identified in the listing are the main constructions of the language, nesting level of constructions, etc.;
--facilities for communication between the BESM-6 and MVK; transfer of texts to the MVK and vice versa is now supported by several methods: through punched cards, disk for communication with a special machine with subsequent transfer to a disk compatible with the MVK, and Unified System magnetic tape, connected to the BESM-6 the standard way.

We should also mention the following service routines that have been incorporated into the various general systems components and those simulating the MVK [multiprocessor computer complex]:
--archive service and archive services [servis arkhiva i sluzhby arkhiva];
--service for developers of translators;
--service for various queries on the situation and changes in it that facilitates interactive operation with the complex.

Problems of Operation. The TEMP complex is an autonomous system and needs only the DISPAK OS operating system for operation on the BESM-6. The usual TEMP complex configuration presupposes availability of disks, and in this case the minimum BESM-6 configuration is: three magnetic disk storage units, one terminal (T-340 or RIN-609) and DISPAK OS (beginning with version 92.M). However, there is a version of the complex that operates on the BESM-6 without disks with the appropriate version of DISPAK OS.

Main complex components have the following performance referred to CPU time:

Autocoder translator	1000-1500 lines/minute
"El'brus-1" MVK interpreter	500-700 instructions/second
FORTRAN translator	4-5 lines/minute
ALGOL-60 translator	2-3 lines/minute

The average number of exchanges the translators have from autocoder is 100 per 1000 lines, and the interpreter has 1 per 100 executed MVK instructions. Maximum size of files storable in the archive is 377_8 zones. The complex permits operation almost totally on disks, which reduces the need for magnetic drums in short supply for the BESM-6 and facilitates entry of tasks for solution.

Overall efficiency of the complex can be illustrated by this example: A program with a 100 lines of autocoder can be debugged from start to finish by a programmer with average skills who has mastered the complex within an hour's session at a terminal with output of just the final results on the printer.

All user documentation (instructions) is in the TEMP complex archive and presented in the form needed for output through a complex component, the documentor; users can obtain instructions in any desired quantity. The instructions include the following programmer guides: TEMP complex prospectus, TEMP complex general description, job control language [JCL] (TEMP JCL monitor), documentor, task generation system, text editor, autocoder translator, "El'brus-1" MVK interpreter, object code file editor, autostructurer, cross reference dictionary, graphic procedure package, transfer of texts from TEMP complex to the MVK and vice versa, general service routines and the service for developers of translators.

There is a separate set of documentation for the "El'brus-1" MVK and its software; it is presented similarly and can be furnished with the TEMP complex.

Training personnel for skilled operation on the TEMP complex takes a month. It is conducted most efficiently in the interactive mode of operation.

Instructional materials on more correct (efficient and economic) use of the complex in individual and collective operation are being prepared for dissemination. There

are recommendations on methods of developing on the TEMP complex average programs and large (over, say, 30,000 lines in autocoder) software systems.

Complex Distribution and Maintenance. At present, the complex is distributed by developers, but the TEMP complex prospectus does contain exhaustive instructions for its standard generation, in which the complex takes up a whole basic disk (minimum size of the area allocated for the TEMP complex on disk is 300 zones).

Complex generation takes about an hour and is done in the background of foreground computation in DISPAK OS.

Complex maintenance is also performed by the developers; a group of three persons processes criticisms received, accepts and checks on a bank of tests the new versions of the components and those newly included in the TEMP complex, develops new versions and informs users of this.

Complex Development. Despite wide dissemination, the TEMP complex is in the status of intensive development. Here are the main directions of this development:
--enhancement of performance of components by increasing speed and improving interface with the user;
--development of new components, the most important of which are a debugger providing far more txtensive facilities for interactive debugging of MVK programs compared to the debugging mode mentioned above available in the interpreter, and a "large" "El'brus-1" MVK interpreter that will operate under control of the real MVK OS which will provide TEMP complex users with a full set of MVK OS functions in real execution, as well as an optimizer for the object code file and a guide/information provider for the complex;
--modification of the complex for the new model, the "El'brus-2" MVK;
--transition of the complex to a new set of general components with replacement of the internal support system [4].

Deserving separate mention is the effort on translating the TEMP complex for the Unified System of Computers and the BESM built with integrated circuits.

Conclusion. The TEMP complex is a large program system on the BESM-6 computer. Its extensive use is due to the conditions of development of the "El'brus-1" MVK, as well as the need for advance programming of many production tasks prior to delivery of the MVK. The complex was able to meet these needs to an adequate extent.

Among the programs debugged on the TEMP complex, one can cite the ALGOL-60 and FORTRAN translators that have already passed state tests. The total amount of programs debugged on the TEMP complex so far exceeds 250,000 lines in "El'brus" autocoder.

It can be assumed that the complex will be used intensively until the 1984-1885 period.

In addition to the authors of this article, the main TEMP complex developers are N. V. Golovleva, S. Yu. Dederer, V. V. Zabotin, I. F. Lesovaya, V. A. Mozzherin, N. A. Ozols, L. N. Pen'kova, V. N. Polivanov, V. P. Pyatkin, V. A. Chausov and N. A. Shishova.

Also participating in the development were A. F. Bondarenko, A. V. Kaygorodov and Ya. M. Kurlyandchik.

BIBLIOGRAPHY

1. Burtsev, V. S., "Printsipy postroyeniya mnogoprotsessornykh vychislitel'nykh kompleksov 'El'brus'" [Principles of Design of the "El'brus" Multiprocessor Computing Complexes], Moscow, 1977, 36 pages (Preprint No 1, ITM i VT AN SSSR [Institute of Precision Mechanics and Computer Engineering, USSR Academy of Sciences]).

2. Gololobov, V. I.; Cheblakov, B. G.; and Chinin, G. D., "High-Level Machine-Oriented Language for the BESM-6 Computer," in "Razvitiye programmnogo obespecheniya BESM-6" [Development of BESM-6 Software], Moscow, 1975, pp 50-51.

3. Usov, S. A., "Dialogovyy monitor DIMON" [DIMON Dialog Monitor], Moscow, Nauka, 1979, 128 pages.

4. Kurlyandchik, Ya. M., "Sistema upravleniya faylami" [File Management System], Moscow, 1980, 20 pages (Preprint No 13, Novosibirsk Branch, Institute of Precision Mechanics and Computer Engineering, USSR Academy of Sciences]).

8545
CSO: 1863/179

UDC 681.3.02

REAL TIME OS FOR COMPUTER COMPLEXES BASED ON PROBLEM-ORIENTED MINIPROCESSORS

Kiev UPRAVLYAYUSHCHIYE SISTEMY I MASHINY in Russian No 2, Mar-Apr 82 (manuscript received 30 Jun 81, after revision 3 Sep 81) pp 103-105

[Article by Vyacheslav Vladimirovich Gayduk, engineer, SKB MMS IK AN USSR [Special Design Bureau for Mathematical Machines and Systems, Institute of Cybernetics, UkSSR Academy of Sciences], Kiev

[Text] Introduction. Described in this article are the design principles and capabilities of the control component of a real-time operating system (OS RV) for a minicomputer intended for operation under field conditions.

A minicomputer with microprogram control [1] is being delivered in single or dual processor versions [2-3]. A second processor is needed to raise the overall real-time system performance by physical parallel execution of operations. At the functional level, the first processor performs the processes associated with unformatted input/output and characterized by numerous operations for retrieval, sorting and merging of data. The second processor is oriented to problems of visualization. Surplus resources of both processors can be used for execution of any computer processes.

Each processor has its own executive memory with 1024 80-bit words which holds all the microprograms for this processor. The single processor version comes with main storage and read-only memory [ROM] with 8K 16-bit words each; the dual version has main storage and ROM with 16K 16-bit words each. The main and ROM have general addressing and can be increased with blocks of 4K 16-bit words each (ROM with 8K each) to a total of 64K. Main blocks are added from tht zero block in ascending addresses, and ROM blocks from the thirty-first block in descending addresses. Processors in the dual processor system operate with a common field of memory (main and ROM) and differ only by the sets of microprograms in their own executive storage units.

The processors are synchronized by special interrupt registers. Each processor can interrupt the other by one of the external interrupt bits. Minicomputer processors have the four-level system of interrupts:
--interrupt from units on the block-multiplexer channel;
--external interrupts;
interrupts from units on the multiplexer channel; and
--interrupts for system calls.

Up to 16 and 128 external units, respectively, can be connected (through interface cards) to the block-multiplexer and the multiplexer channels.

External interruptions have eight bits, part of which for the first processor have been allocated as follows: system timer (0 bit), attention key (bit 1), synchronization of processors (bits 4 and 5), warning on voltage drop below rated (bit 7). Only two bits of external interruptions have been provided for the second processor; they are used to synchronize its operation with the first processor.

Requirements for the Real-Time OS Control Component. In developing control components for real-time OS for minicomputers, the main requirement is ensuring a standard interface bttween the hardware configuration and the application programs. Usually, different versions of operating systems, development of which entails considerable cost, are required for each area of minicomputer application for problems to be solved in real time.

The modern control component in the real-time OS for minicomputers implements functions previously encountered only in large computer OS's. The systems programs enabling these functions (multiprogram operation, file managtment, swapping) are the most labor-intensive when software is developed [4]. This requires a modular organization of the real-time OS that allows effecting interaction of specific application programs and hardware with substantial costs [5].

The second requirement on real-time OS pertains to tht set of system calls used in programming real processes. This set must be complete and convenient for the programmer and take up minimum time in execution.

The third requirement is to make real-time program execution independent of the number of processors in the system [6], i.e. user programs are written irrespective of the number of system processors and their execution in a dual processor system just reduces the system response time as a whole.

Basic Capabilities of Real-Time OS. Real-time OS permits simultaneous operation of from one to five user tasks, each of which has access to system resources (external units, general programs, files, processors) in accordance with its priority.

To organize real-time operation, the user can create up to 16 processes in each task and effect the needed synchronization for execution of them. These capabilities, as well as initialization of exchange with external units, are effected by system calls that are the sole method of calling real-time OS by the user. When a system call is executed, a program interruption occurs for a call to the system (lowest priority class of interruptions). In calling external units, a user uses their logical number, and the operator determines the correspondence of logical and physical numbers of these units immediately before the start of the task, which permits replacement of single-type external units (for example, CONSUL--teletype) or writing to magnetic disk the information intended for output to a multiplexer channel unit.

When several processes call for one resource, real-time OS reacts as a function of the type of resource and priority of the processes. Access to files and external units is controlled at two levels: logical and physical. The logical level is

maintained by the file system, in which protection of the resource is specified by the user himself. The user can have no effect on the mechanism of protecting external units at the physical level. When a process calls for an external unit that is busy, this process is put on hold until the device is free.

Program resources are divided into two types: program and data. The computing process on the minicomputer is organized in such a way that all programs (except programs for control components of real-time OS) are reenttrable, i.e. they are common resources for all processes and accessible by them at any time. Protection and processing sequence of data are organized by the user by using the mechanism of semaphores, calls to which are included in the set of system calls.

Access to the second processor is determined by the operator, who must specify the number of processes for the second processor before task execution. This solution is due to the inadvisability of using the SCHEDULER of higher level processors in real-time OS for minicomputers.

The job control language permits:
--establishing the number of tasks;
--allocating storage between tasks;
--initiating task execution;
--setting timer mode of operation;
--maintaining a system calendar;
--establsihing correspondence between logical and physical numbers of external units.

At the operator's request, real-time OS sends all current information on system status to the console.

Software Structure of the Dual Processor System. Three modules are added to the real-time OS control component when there are two system processors: the interrupt handler for the second processor (handles only program interrupts for system calls and two external interrupts), the service process for initialization of the second processor and the service process for handling its system calls. Each processor can execute all instructions and, consequently, can be allocated by the operating system for any process.

A differenct between the processors shows up only in handling interrupts, i.e. at the system level which is inaccessible to the user. The corresponding fields of the program status words [PSW] for the second processor are located after the same fields for the first processor and are adjusted for transfer of control to the second processor's interrupt handler.

At the system level, except for the current operation on execution of processes, the first processor implements all the functions of real-time OS and handlers of its interruptions and the operation of the drivers for the external units. The second processor also handles two external interrupts and has access to only two real-time OS modules: to the module for process preservation and to the module for full recovery of it.

Logical interaction of the processors is effected as follows. In operating with the module for selecting the highest priority process, the first processor

determines the number of the process (task) that must be executed in the second processor and starts the service process for initialization; after this, the second processor receives the signal to start execution of the new process. The start of execution of the new process is implemented in three stages:
--preservation of the current process;
--sending a confirmation to the first processor;
--restoration of the new process and transfer of control to it.

After receiving confirmation, the first processor adjusts the real-time OS system status block to properly reflect the new system status. In this situation, the first processor initiates interaction of the processors and gives the operation to the second.

In the second version of processor interaction, the second processor is the initiator. If a system call is encountered during execution of a series of instructions in the second processor, the second processor's handler writes the characteristics of this system call to the system status block, sends an interruption to the first processor and waits for execution of the system call by the first processor. After receiving the interruption, the first processor starts the service process for handling system calls. If the system call involves only updating the system status block, then after execution of this operation, the first processor informs the second that the process can be continued, starting with the next instruction after the system call. If the system call involves an external unit, then after its initialization, the second processor receives a message that it can execute another process for the time of exchange with the unit.

Conclusion. All real-time OS modules are located in the ROM, which reduces the time for executing them by one half. Modules for preservation, restoration and translation into positional code, performed in closed interruptions and used most often in operating with rtal-time OS, have been brought to the level of instructions and are placed in the executive storage of each processor, which reduces considerably the time for executing them compared to ordinary ROM (for example, for process preservation module, execution time is reduced from 82 to 22 microseconds). Also, a separate general-purpose register and pointer have been allocated just for real-time OS operation. They are used to efficiently organize the handlers for the four-level interrupt system for the minicomputer and reduce the total size of programs for real-time OS control component to 1100 16-bit words (1300 words for the dual processor version of the system).

BIBLIOGRAPHY

1. Chu, Ya., "Organization of Computers and Microprogramming," Moscow, Mir, 1975, 592 pages.

2. Korniyenko, G. I., "'Etalon' Problem-Oriented Digital Computer for Real-Time Systems," USiM, No 1, 1979, pp 104-106.

3. Korniyenko, G. I., "Digital Computer Complex for Multichannel Processing of Experimental Data in Real Time (TsVK PIRS)," USiM, No 6, 1979, pp 130-136.

4. Donovan, J., "Systems Programming," Moscow, Mir, 1975, 540 pages.

5. Sheverda, O. N., "Analysis of the Data General Real-Time Operating System," in "Sistemy ekspress-obrabotki dannykh v real'nom masshtabe vremeni" [Real-Time Express Data Processing Systems], Kiev, 1979, pp 39-50.

6. "IBM System/370, Principles of Operation," Moscow, Mir, 1975, 576 pages.

8545
CSO: 1863/179

SPEECH RECOGNITION DEVICE

Moscow GUDOK in Russian 16 Apr 82 p 4

[Article by A. Zelentsov, Kiev: "A Machine Listens and Speaks"]

[Excerpt] In the Kiev Institute of Cybernetics a speech recognition system for groups of human words is working and has a vocabulary of a thousand words, which is sufficient for conversations about industrial matters. But by what principle is the computer able to recognize this or that word and even whole sentences?

"We have realized a phoneme recognition principle," says T. Vintsyuk, a candidate of technical sciences. "What does this mean? There are 80 variants of different Russian language phonemes in the memory of the machine. Let's say that one and the same vowel is represented by eight variants. These variants represent means of expression, the way that the vowel 'a' is pronounced before and after consonents, after pauses, and other variants, that is, in various positions. A different phoneme, for example 'u', has nine variants, the phoneme 'sh' has one, and the phoneme 's' also has one. It is well-known that there are 41 phonemes in the Russian language. There are more in Ukrainian, because the same phoneme in Russian is represented by several variants.

"Further, how does the recognition process work? We speak a word in a microphone. An electronic signal from the end of the microphone enters the computer and is subjected to further processing in order for recognition to take place. A comparison is made of this speech signal with the standard signals of the 80 phonemes which are stored in the memory of the machine, and a generated answer comes out. The recognition-answer is formulated as a transcription of the word. In this way the speech signal in the course of analysis is transformed into a textual equivalent, which is written on the computer's lighted panel indicator. We say the word, and read the result on the indicator.

"How fast does the recognition take place? In other words, how much time does it take the machine to understand exactly what was said?

"When the vocabulary of the machine is not very great, for example, 100 words, the response is produced practically instantaneously, just after the word is said. In the case where the vocabulary is slightly larger, let's say 500 words, then we can perceive a certain delay which lasts three seconds.

...The computer room of the Institute of Cybernetics. Right before us is the BESM-6 electronic computer. On the left in front of the microphone sits an employee of the laboratory who is pronouncing phrases that are made up of words that have been entered into the dictionary that the machine understands: open, blank, close, data, colon, index, zero, stop.

Several seconds went by, and on the light panel appeared the running inscription of the words that were pronounced.

9946
CSO: 1863/156

UDC 534.782:621

FONEMOFON-3 VOICE OUTPUT DEVICE FOR COMPUTERS

Kiev UPRAVLYAYUSHCHIYE SISTEMY I MASHINY in Russian No 2, Mar-Apr 82 (manuscript received 20 Apr 81) pp 33-37

[Article by Boris Mefod'yevich Lobanov, candidate of engineering science; Vladimir Vladimirovich Minkevich, senior scientific associate; Boris Vasil'yevich Panchenko, candidate of engineering science; and Leonid Minovich Pervoy, engineer; all from the Minsk Branch of the TsNIIS [Central Scientific Research Institute of Communications]]

[Text] The speech method of interaction in man-machine systems has fundamental advantages. The main ones are:
--convenience, naturalness and simplicity of interaction, requiring no special preparation, which expands considerably the group of potential users of automated control systems and raises computer utilization efficiency;
--relief of the visual channel during information output and elimination of manual manipulations during input, which increases the speed of interacting with computers and reduces the number of operator errors; and
--the capability of using as terminals ordinary telephone instruments and the existing network of telephone communication, which reduces considerably the size of capital investment required in putting man-machine mass queueing systems into operation.

These advantages of voice input/output systems were a potent incentive for comprehensive research in this field [1]. Now, many studies are close to completion and reports on the start of industrial production of automatic speech recognition and synthesis devices have appeared [2]. The problems of voice output have been studied most thoroughly.

There are a number of finished developments of voice output devices (speech synthesizers) that enable vocal reproduction from several dozens to several hundreds of words [3]. Without belittling their practical value, say, in systems of autoresponse and talking clocks, let us note that this sort of device is hardly suitable as a standard computer peripheral. Actually, they perform the role of a digital magnetic tape recorder, enabling readout from storage and recreation of the speech signals of individual words or phrases, previously read by a speaker and recorded by using a particular method for compressing the speech signal.

For a speech synthesizer to become a competent member in a number of types of peripherals already existing, such as displays, printers, etc., it must enable in

Key:

1. computer
2. input/output interface
3. speech synthesizer
4. telephone automatic unit
5. telephone line

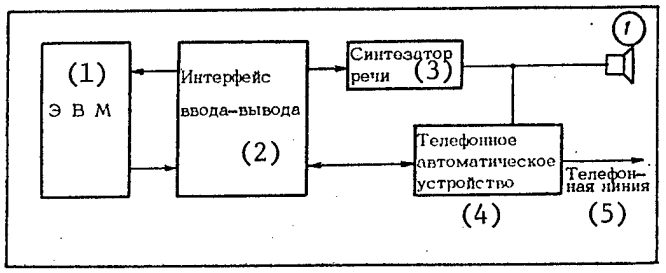

Fig. 1. Structural diagram of voice output unit

Key:

1. computer
2. intonation simulation unit
3. prosodic information extractor
4. intonation parameter generator
5. text conversion unit
6. buffer storage unit (1K bytes)
7. character-to-phoneme translator
8. syllable separator
9. articulation simulation unit
10. time coordinator of articulator movements
11. articulator instruction generator
12. articulator movement simulator
13. acoustics simulation unit
14. articulator-to-formant translator
15. excitation source
16. formant model of vocal tract

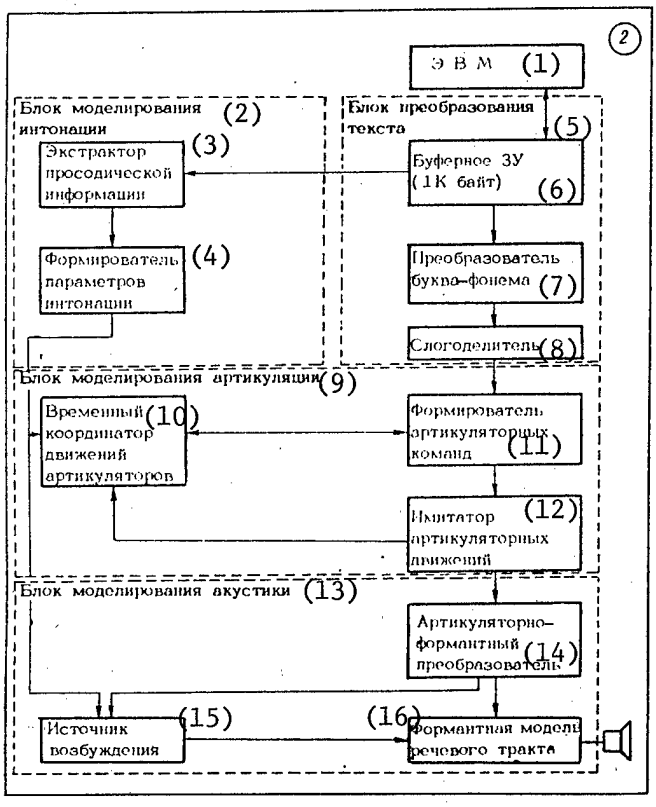

Fig. 2. Structural diagram of speech synthesizer

real time speech synthesis of an unlimited vocabulary for output alphabetic information, i.e. for text of arbitrary content. A speech synthesizer of this type, the Fonemofon-3, is described below. The research and development for this device has now been completed, and industrial production is to start in 1982.

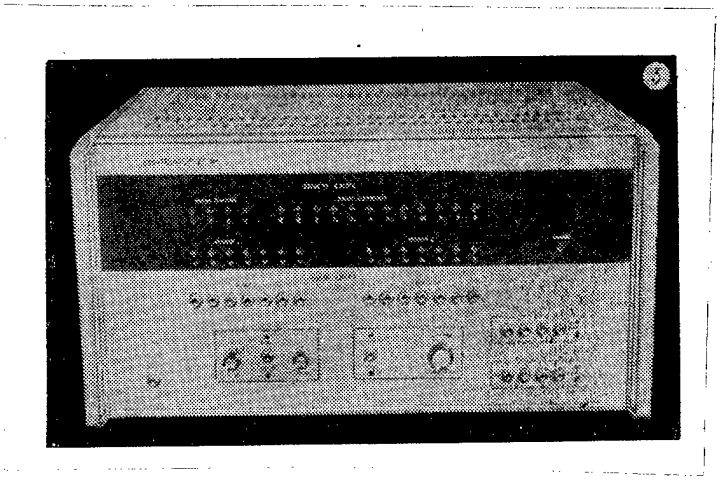

Fig. 3. External view of prototype of synthesizer

Key:

1. ATS [automatic telephone exchange]
2. telephone automatic unit
3. telephone line switching unit
4. telephone signal receiver unit
5. automatic dial
6. unit for generating information for computer
7. control unit
8. speech synthesizer
9. input interface
10. output interface
11. computer

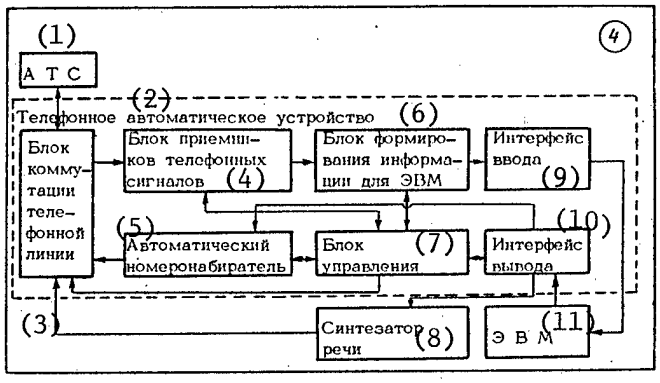

Fig. 4. Structural diagram of telephone automatic unit (TAU)

The structural diagram of the voice output device is shown in fig. 1. The device consists of the speech synthesizer, input/output interface and the telephone automatic unit (TAU). The speech synthesizer is designed to produce sound for the text coming from the computer; the input/output interface joins the speech synthesizer and TAU to the computer channels, and the TAU joins the computer and speech synthesizer to a telephone circuit. Thus, in addition to its main function of vocal representation of information coming in, the device also enables interaction between the computer and telephone network subscribers by using the TAU. In the process, the TAU performs the role of a unique telephone instrument used by the computer to call up and transmit information in speech form to any subscriber, and on the other hand, a subscriber can call up the computer to obtain needed information.

The device can also operate directly with a computer or system operator (for example, in computer-aided design systems) for vocal comments on the problem being solved or for sound accompaniment of graphic images on a display. In this case, the TAU is not used, and a speaker or headphone is connected to the device.

Let us consider the operation of the individual assemblies in the device.

Shown in fig. 2 is a structural diagram of the speech synthesizer. The speech synthesis implemented by the device is based on simulation of natural speech formation. By functional purpose, the device is divided into four units: the text conversion unit, and the units for simulating intonation, articulation and acoustics of speech formation. The text information from the computer is written in individual portions (a sentence or group of sentences) in the 1K-byte buffer storage unit (BZU). The buffer enables storage of information for synthesis, up to 120 words on the average, i.e. about one minute of speech. After pronunciation of the entire text stored in the buffer, the synthesizer sends a request for input of a new portion of text.

The text in the buffer is analyzed for translation from orthographic form to phoneme (letter-to-phoneme translator) and for isolating prosodic information: type and location of punctuation marks, stresses, position of the i-th phoneme element in a word, syntagma, phrase and others. The prosodic information extractor provides for proper operation of the generator of the basic intonation speech parameters: pitch of the basic tone (melodics) and flowing tempo (rhythm). The signals generated by the generator go to the control inputs of the time coordinator for articulator movements (rhythm control) and source of vocal excitation (melodics control).

From the output of the letter-to-phoneme translator, the phoneme text goes to the syllable divider that marks off the text into elementary open syllables. Given below is an example of translation of orthographic text into phoneme and division into open syllables.

Orthographic text: Дегали сдают в склад у остановки. [Parts will be turned into the warehouse at the stop.]

Phoneme text: д'этал'и␣здайут␣фсклат␣уастанофк'и

Text marked off into syllables: |д'э|та|л'и|зγ|да|йу|.тγ|фγ|сγ|кγ|
|ла|тγ|σу|σа|сγ|та|но|фγ|к'и|

In this example, the accent mark ['] at a consonant phoneme indicates its softness, the "␣" mark indicates the space between words, the "γ" indicates a fictitious vowel, and the "σ" indicates a fictitious consonant. The breakdown into elementary open syllables consists in dividing the phoneme text into "consonant-vowel" groups. When there is a confluence of two or more consonants, the fictitious vowel γ is inserted betwten them, and in a confluence of vowels, the fictitious consonant σ is inserted.

At the articulation level, an open syllable corresponds to an elementary articulator action, the simultaneous flow of certain articulators. Based on analysis of the phoneme composition of a syllable, a set of articulator instructions is formed that specify the movement of articulators, such as the lips, tip of the tongue,

body of the tongue, uvula and others. Under the effect of the instructions, the articulator movement simulator generates signals that simulate the trajectory of movement of the individual articulators. The overall organization of the process of articulating a syllable is effected by the time coordinator of articulator movements.

Formed at the output of the articulation simulation unit is a set of signals that simulate speech movement. The task of the acoustics simulation unit is to translate these signals into an electrical speech signal. The immediate source of the synthesized speech signal is the formant model of the human vocal tract excited by vocal or sound sources of excitation. In the formant model, the transfer function of the vocal tract is approximated by a corresponding set of resonators (formant filters), the frequency and level of excitation of which can be retuned within certain limits. Current values of formant parameters (frequencies and amplitudes for excitation of formants) are computed in the articulator-to-formant translator that realizes the nonlinear operator for translating the articulator movement space vector into a formant parameter space vector.

Shown in fig. 3 is the external view of the synthesizer prototype. Structurally, the device has been implemented with 30 TEZ's [standard exchange cards] based on the standard in the Unified System of Computers. The element base is series 155 and 140 integrated circuits. The main assemblies in the synthesizer are protected by nine patents [4-12]. Here are the specifications:

Type of input information	GOST 13052-67 binary codes
Alphabet of input symbols	Russian letters, punctuation marks
Size of vocabulary	unlimited
Rate, pitch of voice, loudness	adjustable
Sound intelligibility	not less than 93 percent according to GOST 16600-72
Power consumption	no more than 60 W

A structural diagram of the telephone automatic unit and the input/output interface is shown in fig. 4.

The devices include: the telephone line switching unit, telephone signal receiver unit, automatic dial, the unit for generating information for the computer, the control unit and the interfaces for information input and output.

In the initial state, the TAU is ready to receive a call signal. A subscriber dials the number assigned to the information-inquiry system; when the connection is established, a call signal goes to the device from the telephone network through the telephone line switching unit to the telephone signal receiver unit where a corresponding pulse is generated. By this pulse, an eight-bit byte is generated in the computer information generation unit and then goes to the computer through the input interface.

The computer analyzes the byte received, generates a series of bytes for the corresponding message and sends it through the output interface to the control unit and to the input of the speech synthesizer. The first byte in the series is the service byte that controls the connection of the speech synthesizer to the telephone line. The computer speech message goes to the subscriber through the telephone network. The last byte in the series is a service byte that controls the connection of the telephone signal receiver unit to the telephone line. By using the telephone instrument dial, the subscriber generates the tonal code of a request which goes through the telephone network to the device, where the corresponding byte is generated for the computer. Output of a message for the request occurs similarly to output of a message for the call signal.

The availability of the tonal request code in the telephone signal receiver unit enables a subscriber to conduct a dialog with the computer. In this case, the computer asks the subscriber questions, and the subscriber dials a 1 or 0, corresponding to yes or no respectively.

The subscriber-computer dialog can be terminated at the initiative of the subscriber or the computer. To terminate a dialog, the subscriber hangs up the telephone receiver; in the process, the release signal generated in the telephone network goes to the device, where the corresponding byte is generated for the computer, which tells the program the dialog is terminated. The release signal also disconnects the TAU and telephone network and sets the device in a state of readiness to accept a call signal from another subscriber. The dialog is terminated at computer initiative when the service byte controlling disconnection of the device from the telephone network is placed at the end of the message byte series generated in the computer.

The availability of the automatic dial in the TAU permits the computer to call up any telephone network subscriber and establish a connection with him. In this case, the computer generates the service byte corresponding to connection of the device to the telephone line and waits for the signal "ATS [automatic telephone exchange] READY."

When the corresponding byte goes to the computer, the computer generates a series of bytes consisting of a service byte that sets the dial memory in the write mode, the several bytes corresponding to the subscriber's telephone number, and another service byte that tells the dial when writing of the telephone number is finished.

The automatic dial generates a series of mark pulses for the ATS that corresponds to the subscriber's telephone number, and the computer goes into the mode of waiting for the byte that corresponds to the fact that the subscriber has lifted his receiver and said something, like "Hello" or whatever.

If a subscriber does not answer within a certain time, the computer sends a service byte to disconnect the TAU from the telephone network and continues calling other subscribers. But when the subscriber answers within this time interval, the TAU generates the byte that initiates the computer for generation of the appropriate message which is translated by the speech synthesizer into a speech signal and given to the subscriber through the telephone network.

Speech output device operation is controlled by a special program which is the heart of the software. It performs the following functions: provides for communication between system units and the computer, processes requests received by the system, generates responses and conducts the dialog with the subscriber.

To illustrate the use of a telephone ISS [information-inquiry system] for solving a specific problem, let us discuss the operation of the experimental "Subscriber-MTS [long distance telephone exchange] dialog system. The system is designed to report on credit services of the intercity telephone exchange for those staying in hotels. The system subscriber is the hotel manager. The system permits obtaining the following information: name of the city and telephone number of subscriber called, date, duration and cost of the conversation.

The system operates in the "computer as initiator" mode right after the end of a long distance call from any hotel telephone or at certain intervals established in advance. The system automatically establishes communication with the hotel manager and in the interactive mode gives him the information on calls made. The information obtained by the manager is used to bill the caller for the long distance charges.

When necessary, the manager can call the system on his own initiative. In this case, after confirming establishment of communication with the system, the latter requests the subscriber's telephone number of interest. The manager inputs the number using the dial into the system, where it is analyzed, translated into text and read back to the user to confirm proper reception of the information. After confirmation, the system makes a retrieval from the appropriate file. The key is the subscriber telephone number received by it. The program provides a seven-second delay for subscriber actions. At the expiration of this time, the system asks the subscriber to perform the actions needed. If the action is not completed, the system disconnects itself from the subscriber.

Here is an example of a dialog between a subscriber and the system.

Subscriber. (dials system code).

System. Long distance service. Dial the code of your request.

Subscriber. (dials the telephone number) 21-42-84.

System. You are interested in the number twenty-one, forty-two, eighty-four? Please confirm.

Subscriber. Yes.

System. Write this down. Call to Kiev on the twenty-ninth of March, to telephone number: twenty-three, thirty-one, forty. Duration: five minutes. Cost of call: zero rubles, seventy kopecks. Confirm receipt.

Subscriber. Yes.

System. Continuing. Call to Riga on the fourth of May, to telephone number: twenty-one, seventy-nine, seventy-nine. Duration: three minutes. Cost of call: zero rubles, forty-five kopecks. Confirm receipt.

Subscriber. No.

System. I repeat. Call to Riga on the fourth of May, to telephone number: twenty-one, seventy-nine, seventy-nine. Duration: three minutes. Cost of call: zero rubles, forty-five kopecks. Confirm receipt.

Subcriber. (response code does not correspond to that authorized).

System. Repeat response clearly.

Subscriber. Yes.

System. There is no more information for this item. Do you have more requests?

Subscriber. Yes.

System. Dial the code of your request.

(The dialog continues further if the subscriber dials a request code; otherwise, see below.)

System. Thank you for calling.

After this, subscriber and system communication is interrupted.

Naturally, this example does not exhaust the fields of application for the voice output unit (URV). The following main fields of application can be listed.

Management Information Systems (ASUP, OASU). In these systems, the voice output unit is connected to the local telephone network and provides output of information over the telephone at the user's or computer's initiative. Examples of use: telephone reference service for ministries and departments, information-inquiry services in subsystems for keeping track of execution of orders, supply, personnel and others.

Telephone Information-Inquiry Queueing Systems. Here the voice output unit is connected to a municipal telephone network. Examples of use: systems for settling accounts with subscribers for communication services, sending telegrams by telephone, telephone inquiry systems, telephone services.

Man-Machine Online Control Systems. In these systems, the voice output unit is used to supplement a visual display. Examples of use: power system control services, air traffic control systems.

Systems for Computer-Aided Design, Adjustment or Assembly. Here the voice output unit is used by itself or together with a visual display. Examples of use: system for computer-aided design of integrated microcircuits, automated system for adjustment of complex electronic equipment, automated system for cross connection wiring.

In conclusion, it should be noted that applying voice output in systems will raise considerably the efficiency of utilization of computer hardware and make the computer accessible to the broadest group of users.

BIBLIOGRAPHY

1. Flanagan, J., "
TIIER [Proceedings of IEEE], Vol 64, No 4, 1976, pp 5-17.

2. LeBoss, B., "
ELEKTRONIKA, No 12, 1980, pp 78-89.

3. Rabiner, L. and Shaeffer, P., "

TIIER, Vol 64, No 4, 1976, pp 18-37.

4. Lobanov, B. M., "USSR Patent 459797: Device for Speech Synthesis," pub. in B.I. [Bulletin of Inventions], No 35, 1975.

5. Lobanov, B. M.; Panchenko, B. V.; and Slutsker, G. S., "USSR Patent 485492: Device for Speech Synthesis," published in B. I., No 35, 1975.

6. Lobanov, B. M., "USSR Patent 479107: Speech Formant Synthesizer," published in B. I., No 28, 1975.

7. Panchenko, B. V., "USSR Patent 533925: Device for Output of Speech Signals," published in B. I., No 40, 1976.

8. Lobanov, B. M. and Panchenko, "USSR Patent 492914: Tonal Excitation Pulse Shaper," published IN B. I., No 43, 1975.

9. Lobanov, B. M., "USSR Patent 533966: Speech Synthesizer," published in B. I., No 40, 1976.

10. Lobanov, B. M. and Panchenko, B. V., "USSR Patent 607211: Device for Output of Speech Information," published in B. I., No 18, 1978.

11. Lobanov, B. M. and Panchenko, B. V., "USSR Patent 568853: Device for Speech Synthesis," published in B. I., No 30, 1977.

12. Lobanov, B. M., "USSR Patent 568962: Device for Speech Synthesis," published in B. I., No 30, 1977.

8545
CSO: 1863/179

UDC 621.396

SIMULATOR FOR TRAWLER NAVIGATOR

Kiev UPRAVLYAYUSHCHIYE SISTEMY I MASHINY in Russian No 2, Mar-Apr 82 (manuscript received 14 Nov 80, after revision 24 Jun 81) pp 120-123

[Article by Vladimir Aleksandrovich Tsuranov, general director of PO [planning department] of the fish industry; Mark Georgiyevich Kogon, candidate of engineering sciences, Trawler Fleet Base; and Boris Yevgen'yevich Pal'tsev, engineer, Trawler Fleet Base; all in Kaliningrad]

[Excerpt] At the Center for Ship Electronic Simulators in Kalinigrad, a simulator has been developed that is designed to train trawler navigators in finding and catching shoals of fish. The simulator can operate in the standalone mode or in combination with simulators for a ship's power plant and radar.

Fig. 1 [not reproduced] shows a functional diagram of the simulator that describes its makeup and main links.

The simulator is based on a hybrid control computer complex (UVK) consisting of the M6000 digital and the MN-18M analog computers.

8545
CSO: 1863/179

UDC 681.3.071

AUTOMATED SYSTEM FOR CHECKING KNOWLEDGE USING ASVT M6000 MINICOMPUTER

Kiev UPRAVLYAYUSHCHIYE SISTEMY I MASHINY in Russian No 2, Mar-Apr 82 (manuscript received 30 Jun 81, after revision 16 Sep 81) pp 123-126

[Article by Aleksandr Andreyevich Bessonov, candidate of engineering science, LMI [Leningrad Institute of Mechanics]; Mikhail Mikhaylovich Gerner, engineer; and Aleksandr Semenovich Petrov, junior scientific associate; all from Leningrad]

[Excerpts] The most important and laborious task in designing modern automated training systems (AOS), particularly automated systems for checking knowledge (ASKZ), is the development of the appropriate software.

The practice of using these systems shows raising training effectiveness is promoted by improving management of the process of checking knowledge of trainees by special software. The latter must form the internal operating system which implements the functioning of the entire hardware complex in the time sharing mode, direct and inverse translation of knowledge checking algorithms from external languages into machine, and the library of programs oriented to specific training courses.

Analysis of known methods for managing the process of knowledge checking and evaluation by using automated systems (for example, [1]) shows that it can be implemented only with high-performance computers with large amounts of storage. Using a knowledge checking method not adaptable to the level of instruction and capabilities of the trainees, but which takes into account only the time spent on the answer [2], does not afford the required validity of check resudts.

Implemented in the suggested automated system is a method of managing the process of checking knowledge and generating an evaluation that affords validity of results with minimal outlays for machine time and loading of memory.

The system is made up of ASVT M6000 modules. The system prototype includes: four A211-8 main storage units, an A151-1 add-on memory unit, an A131-1 arithmetic extender, the A131-3 M6000 processor, the A531-3 keyboard printer, five A531-2 input/output units, four interface circuits (interface cards developed in-house), four trainee terminals (units developed in-house), the A411-4 perforated tape input unit, the A421-2 perforated tape output unit, the A121-1 timer and the A491-1 input/output expander.

Each trainee work station is made up of an input/output unit (VVU) and a trainee terminal (TERM), connected through an interface and expander for input/output to the M6000 processor. A TERM consists of an information presentation unit, a program controlled slide projector with a carousel of the images of the test problems and a console for input of answers that has 11 keys and the appropriate electrical circuitry. The keys allow keying in numbers from 0 to 999 and afford input of the number keyed into main storage. A TERM has a panel to display CORRECT, INCORRECT and TIME EXPIRED, as well as the training course, the knowledge of which is being tested.

The prototype of the automated system for checking knowledge, made on the basis of the second set of the ASVT M6000 minicomputer, is intended for simultaneous and separate testing of the knowledge of four trainees by using individual TERM's and VVU [input/output units].

The ASVT M6000 modules used in the system have the following technical parameters:

Mean processor speed	80×10^3 operations per second
Main storage capacity	16,384 binary 16-bit words
Range of numbers representable	from +32,768 to -32,767
Maximum rate of information input using the perforated tape input unit	1500 rows/second
Rate of information output on five- or eight-channel perforated tape using the UVPL [perforated tape output unit]	90-150 rows/second
Maximum printing rate	10 characters/second
Information input/output rate using the VVU [input/output unit]	up to 7 characters/second
Range of time intervals specifiable by timer	from 64 to 0.524×10^3 microseconds

A year's operation of the prototype system has shown that system response time to each successive inquiry from a TERM has not exceeded 1.2 microsecond, and the time for changing a frame with a test task has been 1.5 second.

From the criterion of storage capacity, the system can support operation of 50-60, and from the criterion of speed, 800-1000 terminals in accordance with the characteristics of the software and technical parameters of the ASVT I TERM modules. Depending on the number of channels for communication with TERM, the system can operate with 96 terminals, i.e. 8 for each of the 12 interface cards installed in the input/output expander.

BIBLIOGRAPHY

1. Medem, Ye. M. and Uspenskiy, V. K., "Testing Knowledge and Effectiveness of Instruction by Using Automated Training Units," USiM, No 3, 1976, pp 136-137.

2. Rudenko, V. D., "On One Method of Assessing Knowledge in Automated Training Systems," USiM, No 3, 1980, pp 122-124.

3. Petrov, A. S., "USSR Patent 758234: Device for Testing Knowledge of Students," published in B. I. [Bulletin of Inventions], No 31, 1980.

4. Petrov, A. S., "USSR Patent 780024: Device for Testing Knowledge of Students," published in B. I., No 42, 1980.

5. Sviridov, A. P., "Possible Approaches for Assessing Reading and Thinking Time during Reading," in "Sbornik dokladov MEI po voprosu ob effektivnosti obucheniya" [Collection of Papers from the Moscow Power Engineering Institute on the Problem of Instructional Effectiveness], Moscow, 1966, part 2, pp 88-101.

6. Petrov, A. S. and Filin, Yu. G., "USSR Patent 780025: Device for Testing Knowledge of Students," published IN B. I., No 42, 1980.

8545
CSO: 1863/179

UDC 658.012.011.56"ASU":681.3:622.33

AUTOMATED CONTROL SYSTEMS IN COAL INDUSTRY

Moscow UGOL' in Russian No 6, Jun 82 pp 3-8

[Article by G. I. Nuzhdikhin, deputy minister of the USSR Coal Industry]

[Excerpts] The number of operating computers in the sector doubled during the 10th Five-Year Plan and their computing capacity increased sevenfold. The actual load of computers of the Minsk series increased by 31.8 percent while that of YeS EVM [Unified computer system] increased fourfold. The total useful operating time of computers increased 2.6-fold. The relative number of workers servicing one computer was reduced by 23.6 percent.

The GVTs [Main computer center] of USSR Minugleprom [Ministry of the Coal Industry], the GVTs of the Ukrainian SSR Minugleprom, 35 IVTs [information computer center] of PO [production association] and six IVTs at coal machine building plants and 17 computer centers of scientific research, design and planning institutes functioned in the coal industry on 1 January 1982.

A large part of the computer centers of the sector is equipped with third-generation computers (YeS-100, YeS-1033). Powerful computer complexes have been put into operation at the main computer center of USSR Minugleprom and the main computer center of the Ukrainian SSR Minugleprom.

The information computer centers are equipped with data receiving and transmitting devices, computer-communications channel integration devices, data preparation equipment on machine carriers, video terminal devices and document duplication equipment. Computers of the Minsk series, which will be replaced during the 11th Five-Year Plan, are also used in information computer centers. Teletypes and user stations (AP), agency communications lines and telephone-telegraph trunk channels of USSR Minsvyaz' [Ministry of Communications] are used to transmit data.

A sector fund of algorithms and programs (OFAP), which numbers more than 200 algorithms and programs developed by organizations of the sector and that have come in from the state fund, has been organized and is functioning for centralized support of all computer centers of the sector with effective programs. Part of the sector developments of applied program packs (PPP) has been turned over to other organizations and the state fund of algorithms and programs.

Organizational ASU [Automated control system]. Organizational automated control systems include those of USSR Minugleprom and the Ukrainian SSR Minugleprom, 31 automated control systems and four information computer centers of production associations and six automated control systems of coal machine building plants.

The second unit of the automated control system of the USSR Minugleprom, the first unit of the automated control system of the Ukrainian SSR Minugleprom were put into operation during the 10th Five-Year Plan, six ASU PO [Automated control system for production association] of All-Union subordination (Vostsibugol', Intaugol', Kemerovougol', Tulaugol', Ekibastuzugol' and Estonslanets), the ASU PO Ordzhonikidzeugol' of the Ukrainian SSR Minugleprom, three automated control systems of coal machine building plants and the automated control system of the Shakhtostroy Trust were put into operation. The ASU PO Primorskugol' and the IVTs PO [Information computer center of production association] Sakhalinugol' were turned over for operation in 1981. A total of 47 coal mining and concentration associations, which account for 92.2 percent of the total production volume, is encompassed by machine processing of data. Thus, for example, the information computer center alone of the Donetskugol' Association services 55 enterprises.

Information computer centers are subordinate to the coal producing associations. Most of them are essentially collective-use centers which perform operations not only for the coal mines and open pit mines, but for other organizations of the sector as well (construction, repair, transport and so on).

A number of computer centers in the Donbass and Kuzbass service 2-3 territorially adjacent associations each. It is planned to expand the network of users of the information computer centers during the 11th Five-Year Plan. Specifically, territorial collective-use computer centers are being developed in the Donetsk, Kuznetsk, Karaganda and other large coal regions on the basis of existing IVTs PO. The information computer center of the Donbassantratsit Association services four associations and more than 80 enterprises.

A total of 127 complex problems for 19 functional subsystems is now solved at the level of the USSR Minugleprom; 165 accounting-control analytical and planning problems for 15 functional subsystems are solved in the ASU for production associations and in this case approximately half of them are typical for all associations.

At the same time, some highly effective complexes of problems that have been checked in industrial operation on the YeS EVM have not achieved application at all facilities where they could be utilized. These complexes of problems primarily include extra charges, accounting and analysis of use of wages (approximately one million personal accounts are processed on the computer and 1.9 million could be processed), planning of predicted planned repairs of equipment (60 percent of the mines are included), accounting and analysis of equipment idle times (40 percent of the mines are included), calculation of plans to eliminate emergencies (15 percent of the mines), accounting of the movement of materials (30 percent of the possible volume of introduction) and accounting and analysis of the use of basic funds (60 percent of the

possible volume of introduction. Complexes of problems on accounting and analysis of loading and sales of coal and the use of large-capacity trucks can be realized in many associations.

The main cause of incomplete introduction of available developments is the insufficient attention on the part of the managers of subsystems in USSR Minugleprom and in the production associations. Total introduction of the enumerated complexes of problems of mass accounting will permit an increase of the load and utilization efficiency of YeS EVM.

The M-600 computers, which could have been used during the 10th Five-Year Plan for the ASU TP [Automated production process control system], are not compatible with the YeS EVM, which makes joint operation of them extremely difficult. Minicomputers, the SM-4, SM-1300 and SM-1800, which are compatible with the YeS EVM and which can be used in the ASU TP, have now begun to come into the sector. Minicomputers will also be used in organizational ASU for easing the load of large computers and to improve the production process.

6521
CSO: 1863/200

EXPERIMENTAL ZONE OF AUTOMATED CONTROL SYSTEM FOR GOSBANK

Moscow DEN'GI I KREDIT in Russian No 5, May 82 pp 28-32

[Article by Candidate of Economic Sciences V. F. Bliznets, deputy manager of Belorussian Republic office of Gosbank, and N. A. Kuz'mich, chief of Economic Planning Administration of Office]

[Excerpt] An experimental zone of the sector automated control system of bank operations (experimental zone OASU, USSR Gosbank) was created to work out planning decisions with respect to the USSR State Bank as a whole, to carry out operational planning within the republic and to introduce new methods of operation of Gosbank institutions under conditions of automated information processing based on the Belorussian Republic office of Gosbank. Its creation is based on the already established system for management of banking operations and complex application of modern YeS EVM [Unified computer system], data transmission equipment, economic and mathematical methods and administrative methods and organizational forms of management.

The main general systems requirements and tasks predetermined the trends of development of the experimental zone of the OASU, USSR Gosbank, directed toward improving the execution of banking operations, the decision-making process and verification of their execution to work out an optimum functional mode of institutions of USSR Gosbank and to intensify their actions on the economic bodies for more complete mobilization of reserves and of increasing the efficiency of social production.

The structure of the experimental zone of OASU, USSR Gosbank, is two computer centers and a network of peripheral hardware complexes installed in rayon, municipal and oblast institutions of Gosbank. Construction of the building for the computer center of the Belorussian Republic office of Gosbank was completed within a short time in 1964-1978. There are especially equipped machine rooms, storehouses, service rooms and so on in it for arrangement of the computer equipment. Construction of the building for the computer center of the Mogilevskaya Oblast office of Gosbank was begun in 1978.

During the period since 1975, specialized organizations have implemented a number of preliminary planning and organizational and technical measures that provide as a final goal the guarantee of communication of all institutions of the Belorussian Republic office of USSR Gosbank with the computer centers servicing them.

Two two-machine VK-1033 computer complexes with expanded set of machine equipment have now been established at the computer center of the Belorussian Republic office of USSR Gosbank. Technical equipping of the data transmission (remote input-output) system consists of specialized devices and peripheral complexes of hardware that includes message concentrators and equipment of operator's positions of Gosbank institutions. Taking the specifics of banking operations into account, the operator's positions are equipped with two types of special terminal devices: one for cash operations and the second for check operations. Moreover, the capability of using teletypes (T-63, RTA-80 and so on) is provided. Experimental models of this equipment have now been installed at the computer center of the Belorussian Republic office and the Minskaya Oblast office of Gosbank, where they were tested.

It should be noted that the prerequisite for selecting the structure of computer complexes of the experimental zone and their capability to function under real conditions were creation and introduction of systems for automation of banking operations into industrial operation earlier at other computer centers of USSR Gosbank and also organization of an experimental bench using equipment of YeS EVM [Unified computer system] and specially developed peripheral devices of the system.

Both the VK-1033 computer complexes and the group integration equipment installed at computer centers and also the peripheral hardware complexes are programmable. In this regard primary attention is devoted to problems of software design, which should guarantee control of all production processes of the traffic of banking information in a complex data transmission network.

Applied programs (external operating programs of banking problems) were developed within the primary tasks determined by the "Technical assignment for the experimental zone of OASU, USSR Gosbank" with respect to the subsystems "Accounting and operational work," "Management of money handling," "Management of credit" and "Management of payment turnover" by specialists of the main computer center, USSR Gosbank, and of the computer centers of the Belorussian Republic, Volgograd, Khark'kov, Gor'kiy, Leningrad, Rostov and other offices of Gosbank.

One of the most laborious, volumetric and operational sections of working with respect to automation in the system of USSR Gosbank is processing operational accounting and statistical information of subordinate institutions. Automation of the complex of operational accounting problems under conditions of OASU, besides computer processing of operations completed daily with respect to clients' accounts by institutions of Gosbank, prepares the necessary data in the form of intramachine files to other subsystems for formation of planning indicators, analysis, forecasting, optimization and so on of banking activity using economic mathematical methods and models.

The computer center of the Belorussian Republic office, associated in 1979 with the mechanized accounting office at Minsk, which processes information on the complex of problems "The operating day" of 16 institutions of Gosbank on keypunch equipment, began development of programs for these problems the same year to process this information on computers. Moreover, unlike their programs developed earlier in the Gosbank system, these problems were planned

on the basis of the operating system OS YeS 6.1. This first made it possible to convert processing of information on the VK-1033 computer on the complex of problems "The operating day" of six divisions of Gosbank in Minsakay Oblast, second it made it possible for the specialists of the computer center to acquire the necessary programming experience, and third, it made it possible to train personnel of the computer center and workers of Gosbank institutions to work under conditions of automated information processing.

Accounting information on the indicated complex of problems of six institutions of Gosbank is now being processed in the experimental zone and preparatory work for turning over 47 tasks for experimental industrial operation, related to starting processes of "The operating day" of the subsystem "Operational accounting work," in development of which one of the scientific research institutes and the computer centers of the Belorussian Republic and Volgograd Oblast office were involved under the methodological supervision of the main computer center, are being completed.

6521
CSO: 1863/200

UDC 656.254.5:621.395.4

APPLICATION OF DIGITAL TRANSMISSION SYSTEMS IN COMMUNICATION NETWORKS FOR RAILROAD BRANCHES

Moscow AVTOMATIKA, TELEMEKHANIKA I SVYAZ' in Russian No 6, Jun 82 pp 6-11

[Article by S. A. Kolbasyuk, post-graduate student, Moscow Institute of Railroad Transportation Engineers]

[Excerpts] Comprehensive development of a branch communication system presupposes introduction of digital transmission systems (TsSP) in an extensive manner along with analog systems. Due to specific features of digital systems, however, using them in branch communication networks entails certain complexities. Described in this article is the capability of using digital transmission systems in a branch communications network.

The favorable factors for introducing digital transmission systems in a communications network for a railroad branch (OD) are the availability in railroad transportation of cable lines, multiplexed and nonmultiplexed with high-frequency systems; the capability of placing units for retransmission stations without outdoor installation housings in the heated spaces of station atenndants; and the capability of placing the retransmitters on open lines in the relay boxes at signal stations for automatic blocking when space is available in them. In doing so, one does not have to use remote power supplies for the retranmitters since the sites proposed for locating them have a reliable system for power supply (current consumption by the retransmitters is extremely low; for example, the IKM-30, accepted for introduction in a railroad branch communication network, consumes only 110±10 mA at 9.6 V ±10%).

Despite the obvious advantages for introducing digital transmission systems on railroad branch communication networks, there are a number of unresolved problems requiring special consideration and study. One is the problem of establishing a shared transmission path, the digital group channel (TsGK) for expeditious-technological communication; a general description of it was given in AVTOMATIKA, TELEMEKHANIKA I SVYAZ', No 2, 1982. Let us discuss the digital group channel in more detail from the structural viewpoint that allows simplifying the equipment and organizing allocation of channels at intermediate stations.

Complexities of developing a digital group channel based on the IKM-30 system are due to this system having a four-wire path for transmission that hinders connecting subscribers in a group call. Also, used in the equipment is nonlinear coding of speech signals, in which circuits for summing units are complicated. Therefore,

the problem of designing the digital group channel comes down to summing of the speech signals in the channel and selecting the channel structure.

Reliability of the Digital Group Channel

A substantial shortcoming in the digital group channel with a ring structure is that when the ring linear path is damaged, communication is interrupted essentially from all intermediate points. The reasons for damage to the KLT [ring linear path] may be damage to the cable line, damage or unstable operation of retransmitters, or malfunction of the group equipment for the intermediate apparatus. A complex of measures permits reducing this deficiency to the minimum.

Eliminating the consequences of the first reason entails using alternate channels of other communication systems (for example, radio relay lines or channels leased from the Ministry of Communications) backing up all or the most critical channels. In this case, to restore the ring structure of the digital group channel in the damaged places of the ring linear path, it is necessary to establish a loop, through which the synchro signals and those of the corresponding KI [expansion unknown] will be retransmitted. In doing so, it is advisable to connect the telephone instruments in accordance with the scheme shown in fig. 10 [not reproduced]. Transmission of synchro signals and generation of a linear signal are effected by the equipment installed on the side of the RS [expansion unknown], the first after the damaged location. For this, the intermediate equipment is supplemented with units for the FLS [line signal conditioner] and the master oscillator which are switched on only when the units forming the loop are tripped.

In the second case, operation of the ring linear path is restored by superimposing a loop on the damaged retransmitter. In doing so, it is necessary that the next retransmitter after it overlap the loss of the doubled retransmission section. For this, input amplifiers for the retransmitters must have the logarithmic shape of the transmitting characteristic or must be equipped with devices for automatic gain control.

In the third case, operation of the ring linear path can also be restored by forming a loop on the damaged intermediate point. In doing so, only the damaged intermediate point itself loses communication. Also, the intermediate equipment can be wired so that all circuits pass through in transit without loss when there is damage.

In any case, however, to raise the reliability of the circuits, devices are required for diagnostics and automatic circuit backup. A prototype of these systems is now under development for devices for technical diagnostics of the paths of one-fourth of the cables multiplexed with a digital transmission system.

8545
CSO: 1863/193

PLANS FOR TELECOMMUNICATIONS NETWORKS

Riga SOVETSKAYA LATVIYA in Russian 2 Apr 82 p 2

[Article by S. Vladimirova: "In the Interests of the Economy"]

[Excerpt] At an open party meeting in January of this year the collective of the Institute of Electronics and Computer Technology of the Latvian SSR Academy of Sciences discussed and took on socialist pledges in honor of the 60th anniversary of the formation of the USSR. The collective of this institute is the USSR head for the design of hardware and software for the computer network of the USSR Academy of Sciences and the academies of the union republics, part of the nationwide unified information system.

The work is being carried out within the framework of a goal-directed integrated program that has been approved by the nation's government and the USSR State Committee for Science and Technology.

According to last year's results, the winners of the socialist competition at the Institute of Electronics and Computer Technology were the scientific collectives of the fourth subdivision, who introduced the results of their research into practice for the creation of computer networks. These are the collectives of the laboratories that deal with logical systems, terminal complexes, wide-band transmitters, and also, the design department. For example, the laboratory of terminal complexes, led by V. Pirogov, who is a corresponding member of the Latvian Academy of Sciences, created an ensemble of computer devices that are connected to each other by specialized means of information transmission, and introduced these in several institutes of our academy.

An important assignment that is part of the socialist obligations for 1982 is a complex for teleprocessing of information which includes a powerful machine of the YeS EVM series and microcomputers. The design, testing, and delivery of this complex for use in the economy by the end of this year has been projected. However, the scientists at the institute intend to complete this task sooner.

9946
CSO: 1863/156

UDC 681.324

ORGANIZATION OF INTERCOMPUTER EXCHANGES IN CENTRAL COMPLEX OF MULTIACCESS COMPUTER CENTER WITH NETWORK ARCHITECTURE

Kiev UPRAVLYAYUSHCHIYE SISTEMY I MASHINY in Russian No 2, Mar-Apr 82 (manuscript received 13 Aug 81) pp 55-60

[Article by Sergey Sergeyevich Shalugin, candidate of engineering sciences, SKB MMS IK AN USSR [Special Design Bureau for Mathematical Machines and Systems, Institute of Cybernetics, UkSSR Academy of Sciences], Kiev]

[Text] Introduction. Modern multiaccess computer centers (VTsKP) must offer a broad set of services to a large number of local and remote users that generate streams of diverse computational and informational problems. Such centers can be built on the base of a major multimachine central computer complex supporting multipurpose data processing and combining, as a rule, different computers spaced several hundreds of meters apart.

Such complexes are created based on the continuity and stage-by-stage development of the capabilities and architecture of the multiaccess computer center. Until recently, these centers were built on the physical structure of individual computers and data teleprocessing systems. Now, however, the need has arisen for development and assimilation of new software and hardware oriented to organization of network modes of solving problems in the central processing complex in the center.

Prerequisites and Problems of Development of Local Computer Networks. In organizing the computing process at major centers, two contradictory requirements have to be met: on one hand, a functionally complete set of services has to be offered to the various groups of users, and on the other, the efficiency and performance of the complex of computers operated in the center has to be raised. To this end, an integrated computer complex has to be established that enables execution of the following functions:
--distributed planning of the load on the different computers;
--equalization of the load on computers with different capacity;
--specialization of the individual computers (for batch processing, program debugging, data base input and others);
--shared use of the equipment, software and data of the individual computers;
--high viability of the computer complex;
--development of a local information network of centers and others.

Existing facilities for computer complexing through external storage units, channel-to-channel adapters, direct control channels and common main storage units allow building a central complex for a center based on system architecture. The latter takes the structural features of each computer into account and enables data

exchange between computer pairs through an individual physical communications channel laid previously. These facilities for complexing do not permit combining a large number different types of large and small computers remote from each other, and consequently do not permit supporting functions entrusted to the center in a fairly simple way.

To solve this problem, it is advisable to make use of network architecture, under which any computer in the complex exchanges information through a common transmitting medium, establishing logical channels of communication with each other.

A multimachine complex with network architecture can be developed using the principles worked out for territorially distributed computer networks. The ways and means of controlling communication between computers in such networks are defined by standards adopted for narrow-band telephone circuits. This led to the emergence of redundant multilevel hierarchy of protocols for translating the streams of data exchanged by computers through the communication circuit network. Using these facilities in setting up a central complex for a center slows down exchange of information between computers and reduces the efficiency of utilization of computer resources.

Shown in [1] is the need of organizing a form of interaction between computers that allows the distributed machines to "exchange information at a rate natural for them," creating in the process temporary communication channels with the appropriate capacity. Multimachine complexes built according to this principle are now called local computer networks (LVS).

Using local networks in multiaccess computer centers permits substantial simplification of structure and implementation of protocols for intermachine exchange of information, frees computer resources for direct data processing, ensures high viability of the central complex, and allows implementation of other capabilities that are not realizable or difficult to realize in multimachine complexes with exchange through individual physical communication channels.

Building a multiaccess computer center based on a local network requires a compatible solution to the following basic problems [2]:

1. Development of inexpensive hardware to enable high-speed reliable data exchange between different types of computers.

2. Development of protocols and software for automatic network control to achieve a balance between internal and external resources of the center.

3. Development of convenient forms and facilities for user interaction with the local network.

4. Development of facilities for interaction between local networks and the latter with regional and territorially distributed networks to unite a set of centers into the state-wide network of computer centers.

Discussed below are implementation principles and facilities for solving these problems that apply to domestic computers.

Logic Structure of Local Computer Network. A local computer network is an open distributed information and computing environment implemented, as a rule, in a limited territory by using special hardware and software facilities. These facilities unite a set (in the general case) of various computers, the logical elements of the network called systems that perform the functions of data input, storage, processing and output.

By functional purpose, the different network systems are divided into the operating, intended for data processing; the terminal, that effect input and output of information from the network; the dispatching, that provide for planning of the work, control of the computer process in the network and granting of service to users; and the converter, which effect interaction between the local and other networks.

In accordance with the open systems architecture adopted by the International Standards Organization (ISO) [3], the logic structure of a local network embraces three virtual networks embedded in each other: a communications, transport and computer. The communications network enables data transmission between different systems and includes communication modules and the physical circuits connecting them.

The transport network enables data transfer between processes (a process may be an applications program, a control planning program and a computer operator) by transport modules that make use of the communications network. The computer network enables functioning of processes and interaction between them in different computers by using logical channels, the terminals of which are transport network ports.

These three virtual networks and the organization of computer operations in a local network are implemented by using six interacting elements of the physical and program structure (fig. 1):
hardware elements:
--the transmitting environment which is the aggregate of the facilities for building the physical connections between systems;
--network controllers that control the transfer of information through the transmitting environment;
--network interfacers, which enable interfacing the network controllers to systems;
program elements:
--network communicators that control the transport network;
--network schedulers that allocate the work in the computer network;
--network preprocessors that enable linguistic intercourse between users in the local network.

Three levels of network functioning control can be isolated. The first (lowest) level is implemented by hardware by the first three structural elements in the local network that control establishment, maintenance and disconnection of physical channels, effect monitoring and ensure transparency of information transfer through the communication network in the local network. The procedures implemented at this level are similar to the procedures in the first and part of the second levels of the ISO hierarchical model of protocols.

Key:

1. system A
2. system B
3. system C
4. communications network
5. transport network
6. computer network
7. symbols
8. transmitting environment
9. network controller
10. network interfacer
11. logical channel
12. network communicator
13. network scheduler
14. network preprocessor
15. user programs

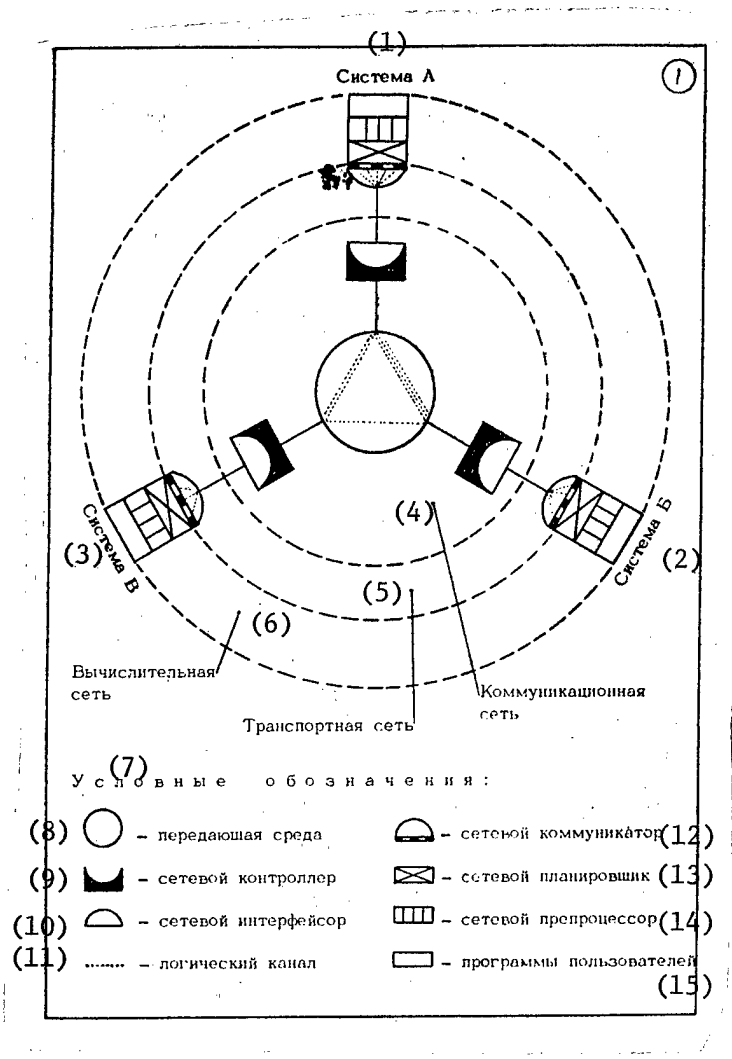

Fig. 1. Hardware and software elements implementing three virtual networks and computer operations in local network

The second level is implemented by software by network communicators of the processes that control setup, maintenance and disconnection of logical channels between computer network processes that effect initialization and de-initialization of an information channel and control flows of data in both directions of the physical channel. Implemented here are the procedures of the fourth and some of the second levels of the ISO model of protocols. It is shown below that implementation in the local network of the network level procedures in the ISO model is inadvisable.

The third functional control level is implemented by software by the network schedulers and preprocessors. The schedulers effect decentralized distribution of work between systems in the local network and organize communication sessions between application processes of one or more systems. The preprocessors provide for control of the computer network from the aspect of users and application processes. Both network schedulers and preprocessors are computer network processes and together with user programs implement the procedures corresponding to the fifth, sixth and seventh levels of the ISO model of protocols.

Physical Structure of Local Computer Network. The aggregate of the first three structural elements form the physical structure of the communications environment in the local network, the network controller and interfacer making up the communications network control unit [4]. When the network controller is an invariable part of the unit, the structure of the network interfacer is determined by the type

of computer included in the local network. These structural elements are designed to enable high-speed reliable transfer of data between systems at negligible cost, i.e. to solve the first problem of design of a central processing complex for a multiaccess computer center with the ideology of the local network. In the process, different alternatives are possible for implementing the topology and methods of controlling the communications network. A detailed description of them can be found in [5, 6].

The local network transmitting environment must meet the requirements of high reliability, simplicity, low cost, no noise, high capacity and mechanical durability. Data transfer on the local network is largely implemented by using serial direct transfer of unmodulated binary signals over coaxial cable of a dedicated unmultiplexed telephone communication line or of a symmetrical physical circuit at a rate of from 480 to 20,000K baud at a distance from several thousands to several hundreds of meters, respectively. Using television cables of light guides affords high data transfer rates at great distances.

Alternatives for implementing the topology of a communications network depend on how the physical connections between systems in the local network are laid. Arbitrary topologies, used in territorially distributed networks, are characterized by high cost for optimization of communication circuit capacity and routing of information flows through the circuits. It is inadvisable to use these topologies in a local network, since here these costs are eliminated by using communication networks with ordered disposition of systems in accordance with radial, mainline or ring topology.

In a network with radial topology, centralized control of data transfer is implemented in the central system, which removes control procedures from the other systems. This control principle is inefficient since relatively high inputs of resources are required from the central system to support reliable operation of the entire network.

A network with mainline topology affords decentralized control of data transfer, effecting a bidirectional flow of information on the bus from one transmitting system to the other systems in the network. The receiver-system reads out information on the bus by the address, recognized by the communications network control unit.

A network with ring topology also affords decentralized control of data transfer, effecting unidirectional flow of information serially from one system to another over physical communication lines. Each communications network control unit either extracts the information from the transmitting environment, if it is addressed to the given system, or sends it to the next unit.

Network controllers are designed to control the exchange of information blocks between systems. They perform the procedures of establishing, maintaining and disconnecting the channels for data transfer between systems, implementing a communications network control method, such as the series circuit, control marker, segmenting, insertion of registers and competition.

In implementing the series circuit method, each communications network control unit has to transmit control information serially from the network controller of one system to the controller of another system over separate communication lines. In the

process, the network controllers are given the task of setting up, maintaining and disconnecting the connection to a physical line and synchronization of information and control blocks sent through the communications network.

In implementing the control marker method, each communications network control unit effects transfer of control from one system to another by using a special byte, the control marker, sent serially through the ring. When a message has to be sent, the system awaits the arrival of the control marker at its controller. The network controller extracts the marker from the transmitting environment, sends the message and then sends the marker to the next controller. Each controller analyzes the address portion of the message passing through it, and when it recognizes its own address, extracts the message from the ring; otherwise, the message and marker sent after it are relayed to the controller in the next system.

In implementing the segmenting method, one of the network controllers breaks up the entire series of communications in the ring into time segments, the duration of which is adequate for transferring a full message. There are free and busy segments. Any controller, after receiving a free segment, can when needed place in it the message of its system and mark it as busy. The controller, to which the message is addressed, extracts it from the ring, sends it to its system and marks the busy segment as free, etc.

Implementing the register insertion method presupposes preloading of a message into the shift register of the network controller for each communications network control unit; then when the network has no communications or in an interval between two adjacent communications, the ring is interrupted, the shift register is inserted in it and the contents of the register are then output to the ring. In the process of outputting the message from the network controller, loaded in its place is a meassage coming in at this time from the transmitting environment.

Implementing the competition method presuposses independent transfer of messages of any system network at any time. In doing so, it is possible that several systems will initiate the start of a transfer at the same time, and thus, conflicts between requests can occur. Network controllers must have the capability of recognizing these situations and then in a random time interval repeating the transmission or implementing the capability of "listening" to the transmitting environment and sending messages when the network has no communications.

Network interfacers are designed to match the data transfer rate in a communications network to the data processing rate in the system and perform the functions of interfacing network controllers to systems. The structure of the interfaces is defined by the structure of the input/output interface for the specific type of computer included in the local network.

Considering the most widespread computers in the country, one can isolate three basic groups of network interfacers. The first group includes the network interfacers enabling an interface to the third-generation large Unified System computers and the "El'brus" as well as to the large second-generation BESM-6 computers.

The second group includes the network interfacers designed to interface network controllers to the small computers in the SM series.

The third group includes network interfacers designed to connect different types of microcomputers to the local network.

Local Network Program Structure. The local network program structure consists of the operating systems for the different types of computers, the sets of system programs that extend the OS capabilities, network operation control software, and user application programs. Examples of systems programs that extend OS capabilities are KAMA [a data base teleprocessing system], OKA [a data base management system], KROS [a job management system], SRV [real-time supervisor], and a number of specialized user programs. Using these programs on a local network permits retention of the capabilities users are accustomed to when operating with Unified System subscriber stations. Systems programs can be interfaced to network operation control programs by using special software packages called logical interface translators (possible versions for implementing them in systems are given in [7, 8]).

Logical interface translators effect the required interface translations between the various access method systems (remote, basic telecommunications, general telecommunications) used in the systems programs listed above and the network access method used by the local network control programs.

The software package for controlling network operation, implemented in each operational or terminal system, includes local network structural elements such as network communicators of processes, network schedulers and network processors.

The network communicator of processes affords control of transmissions between ports of the transport network, implementing the protocol of conveying messages, and performs network access method functions such as initiation, checking and maintenance of a physical connection; setup, maintenance and interruption of logical communication channels; synchronization of processes; addressing and transfer of messages; detection and handling of errors in a logical communications channel and others.

In the network communicator of processes, four modes of process interaction can be implemented: transfer of packets, transfer of messages, rapid exchange and local communication. In the packet transfer mode, the transmitting environment is monopolized for the time of output of just one packet of information, and then is released for transfer in the general case of a random sequence of single packets pertaining to messages of different processes. In this mode, a priority queue of requests for transfer is implemented. Servicing the requests arriving from processes occurs in the order of absolute dispatcher priorities with storage. After completion of servicing, the request is removed from the queue and the process that initiated the request is informed of this.

The message transfer mode corresponds to the virtual channel mode in which the transmitting environment is monopolized by one message for the time of output of a determinate sequence of packets belonging to it. In this mode, the difference of servicing of a stream of requests from processes consists in that servicing can be interrupted only by the process that initiated the request.

The rapid exchange mode is an evolution of the preceding mode and carries out an exchange of data between processes according to specified physical addresses of messages. This allows eliminating intermediate buffering of messages in the network communicator of processes and reducing time spent on log table searching. With

that, processes can change addresses of message areas only after closing of the channel. In other respects, the message handling procedure is the same as the conventional one.

The local communication mode supports data exchange only between processes belonging to the same computer.

The network scheduler is intended for distributed balanced distribution of computer operations generated by internal and external processes of the local network. Planning execution of jobs is determined at any time by the status of the information and computing resources of the entire network. This extends the capabilities of optimizing job planning and allows regulating the load on equipment and reducing the time for execution of an individual job. The aggregate of the last two structural elements for the local network is intended for solving the second of the problems in designing the multiaccess computer center listed above.

The network preprocessor is designed to create convenient and natural forms of human intercourse with application programs. Network preprocessors are implemented in operational or terminal systems based on a high-level language. In the process, network preprocessors effect syntactic analysis of the language and mapping of input strings of user instructions into protocols for controlling application processes. Development of preprocessors will allow achieving essential user independence of hardware and individual software packages functioning in the local computer structure. And this allows solving the third problem in designing the multiaccess computer center with network architecture.

Interaction of Networks. In creating networks of multiaccess computer centers, the problem that arises naturally is that of creating means for interfacing local networks with each other and with regional and territorially distributed networks.

Key:
1. symbols
2. T - territorially distributed network
3. R - regional network
4. L - local network

Shown in fig. 2 are examples of uniting different computer networks. Several local networks L, belonging to one region, are united with each other into one regional network R. When users of some local network need access to networks in other regions, unification of several regional networks is effected. This requires providing for interaction between application programs by means of the transmitting environment for the territorially distributed networks.

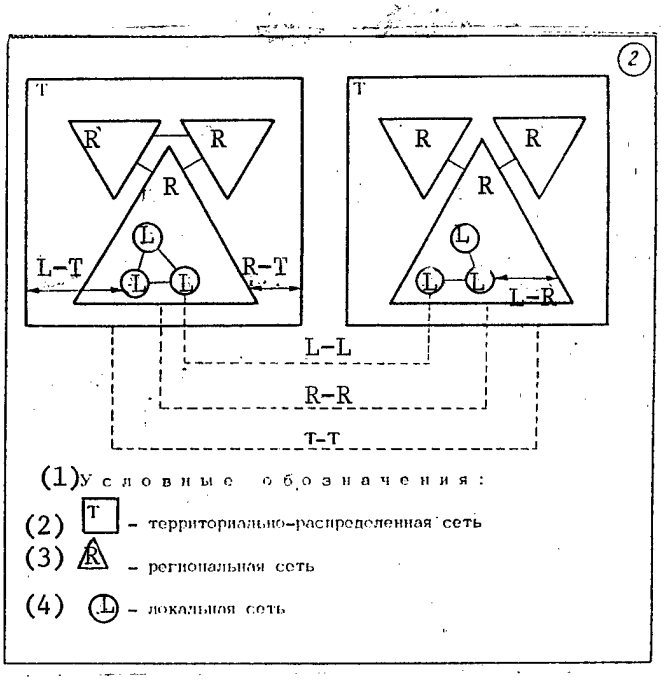

Fig. 2. Examples of uniting different computer networks

The types of interfaces and protocols that have to be maintained in organizing interaction between processes flowing in different networks are also shown in fig. 2. With such a structure for network interaction, it is necessary to implement internetwork interfaces (L-R between a local and regional network; L-T between a local and territorially distributed network; and R-T between a regional and territorially distributed network) and implement internetwork protocols (L-L between local networks; R-R between regional networks; and T-T between territorially distributed networks in different countries through an international data communication network).

At all levels, internetwork connections are implemented (when necessary) by using programmable sluices [9] or network converters that support mutual translation of protocols for the networks to be connected, the task of internetwork services, and others. The selection of functions and structure of network converters is a function also of the methods of internetwork connection (serial, ring, star and other types of connections) and a number of other factors.

With serial connection, unrelated local networks can interact only through the resources of a territorially distributed, regional or intermediate local network.

With the ring connection, each local network has the right to select one of two possible directions for access to the resources of another network. In both cases, the converter enables paired connection of networks and enables dipole conversion of protocols only for two related networks. In the process, systems of unrelated networks in the general case communicate with each other through several different converters.

With the star connection, any network issues a request for internetwork services to one general converter, in which protocols of several networks are converted.

Different versions of implementation of converter processes are possible: in the form of one individual physical device, half-converters in two individual physical devices, program half-converters placed in systems of related networks, and others.

Conclusion. These principles for implementing the physical and program structure of a local network allow building homogeneous and heterogeneous central processing complexes of multiaccess computer centers based on different types of domestic computers. Supported in doing so is the continuity and staged development of the functions and types of services offered to users of multiaccess computer centers, modernization of already existing and development of new, more efficient systemwide software and hardware for data processing, storage and transmission.

BIBLIOGRAPHY

1. Glushkov, V. M., "Seti EVM" [Computer Networks], Kiev, Institute of Cybernetics, UkSSR Academy of Sciences, 1978, 16 pages.

2. Stogniy, A. A.; Nikitin, A. I.; and Shalugin, S. S., "Problems of Designing Central Complex of Multiaccess Computer Centers with Network Architecture," in "Problemy proyektirovaniya i sozdaniya VTs i sistem kollektivnogo pol'zovaniya: Tez. dokl. 4-y Vsesoyuz. nauch.-tekhn. konf." [Problems of Design and Development of Computer Centers and Multiaccess Systems: Theses of Papers from the Fourth All-Union Scientific and Technical Conference], Minsk, 1981, pp 16-20.

3. ISO/TC 97/SC 16, "Reference Model of Open Systems Architecture (Version 3)," 1978, 84 pages.

4. Zubatenko, A. Ya.; Shalugin, S. S.; and Shkolyarenko, A. K., "Local Computer Network Hardware Structure," in "[same title as No. 2]", Minsk, 1981, pp 84-86.

5. Clark, D. D.; Pogran, K. T.; and Reed, D. P., "Local Networks," TIIER [Proceedings of IEEE], Vol 66, No 11, 1978, pp 248-272.

6. Bass, C.; Kennedy, J.; and Davidson, J., "Local Network Gives New Flexibility to Distributed Data Processing," ELEKTRONIKA [Electronics], Vol 53., No 21, 1980, pp 52-68.

7. Yakubaytis, E. A., "Mnogomashinnyye assotsiatsii" [Multimachine Associations], Riga, IEVT [Institute of Electronics and Computer Technology], Latvian SSR Academy of Sciences, 54 pages.

8. _____, "Setevaya arkhitektura vychislitel'nykh tsentrov kollektivnogo pol'zovaniya" [Network Architecture for Multiaccess Computer Centers], Riga, IEVT, Latvian SSR Academy of Sciences, 44 pages.

9. Cerf, V. G. and Kirstein, P. T., "Issues in Packet-Network Interconnection," TIIER, Vol 66, No 11, 1978, pp 112-138.

8545
CSO: 1863/179

UDC 681.3.06./91

DEVELOPING A PROGRAM TO CONTROL MESSAGES AND A TERMINAL NETWORK FOR MIXED ENVIRONMENT OF FUNCTIONING OF DATA TELEPROCESSING SYSTEM

Kiev UPRAVLYAYUSHCHIYE SISTEMY I MASHINY in Russian No 2, Mar-Apr 82 (manuscript received 24 Jun 81, after revision 10 Dec 81) pp 61-67

[Article by Aleksandr Aleksandrovich Sergeyev, Vladimir Mikhaylovich Rossada, and Aleksandr Alekseyevich Kotukhov; all engineers from the GlavNIVTs [Main Scientific Research Computer Center] of the UkSSR State Planning Commission, Kiev]

[Excerpts] Introduction. The modern stage of development of automated systems for planning and management of the national economy is characterized by extensive incorporation of teleaccess facilities into the practice of solving economic planning problems. To implement the process of user interaction with computers in the interactive mode, we developed the general systems software for a remote data processing system (STD) on the base of the graphic (GMD) and telecommunications access methods: the basic (BTAM) and the general (TCAM) that are part of the Unified System operating system (OS).

Functioning in the YeS OS environment are both special-purpose systems software packages, in particular the time sharing system (TSS), the OKA data base management system, the conversational remote job entry system, and a large set of functional interactive programs that use the teleaccess mode. Therefore, in developing a remote data processing system, a relevant problem is the development of common general systems software (OSMO) for the remote data processing system that is acceptable to the majority of users. Prerequisites for developing it are the availability of a common terminal network (TS) and simultaneous execution by the computer system of jobs using the same terminals in the teleaccess mode.

The requirements imposed by users on the general systems software for a remote data processing system used to solve problems in an automated system for planning and management of the national economy consist in implementation of these functions:
--terminal debugging of programs in the interactive mode;
--interactive mode for solving economic planning problems; and
--terminal access to data bases.

This work is devoted to a consideration of a main aspect in the development of the general systems software for a remote data processing system: the development of software to control messages and the terminal network (PUSTS) for the mixed environment of the functioning of the remote data processing system.

Definition of the Mixed Environment for the Functioning of the Remote Data Processing System. The design for the automated national economic planning and management system provides for phased development and implementation of software, i.e. programs being debugged and being operated are executed simultaneously in the computing environment. Of special interest for consideration are the programs that use the teleaccess mode. In the process, the same terminals making up the common terminal network, as a rule, are used to debug programs, access data bases and solve problems in the interactive mode.

The TSS, designed for terminal debugging of programs in the interactive mode, is now widespread in the major computer centers. The standard TSS MCP [message control program] is generated to control the exchange of messages between the TSS and the terminals. The MCP has no interface to communicate with batch-processed programs (PPO) being executed outside the time sharing environment. TSS is independent vnd unique with respect to other components in the general systems software for the remote data processing system. During its operation, TSS monopolizes the terminals and a good part of main storage (minimum of 300K bytes) which denies access to these resources to users external in relation to it.

To overcome this shortcoming, time-shared execution of debugging operations and batch processing is required or one has to develop common system-wide software for a remote data processing system that supports simultaneous execution by the computer system of TSS jobs and user jobs input in the batch mode. The second method is preferred since the user is then not restricted in choosing the time for running his own jobs. But in doing so, an additional allocation of main storage (about 50K bytes) is needed to hold the software to control the messages exchanged by batch processing programs and terminals.

In accordance with this, what we mean by a mixed environment for operating the remote data processing system is the capability of simultaneous execution by the system of TSS and batch jobs. This definition calls for the availability of remote access software that allows the operator at any terminal to operate in either the time sharing or the batch environment. A terminal that is supported by software for logical switching from one environment to the other is called a partitioned terminal.

The complex of service modules for exchange of messages between application programs and terminals allows enabling operation with the unformatted screens of the YeS7066 and Videoton-340 video terminals or with the formatted screens of the YeS7927 displays that are part of the local or remote YeS7920 video terminal stations. The functional capabilities of tht complex of modules for exchanging messages with terminals are shown in fig. 4 [not reproduced] and can be expanded by including new exchange modules in the complex.

Conclusion. The general systems software for remote data processing described in this work, which includes message and terminal network control software, can be used to develop a common remote data processing system at major computer centers that supports running jobs in the interactive mode and remote debugging of software in the time sharing mode. Using the message and terminal network control software as the basic element in a remote data processing system relieves the computer center administration of the necessity of allocating terminals and machine time by categories of operations executed in the remote access mode (remote debugging of

programs and running jobs in the interactive mode) and also saves storage since the operations of terminal network control and exchange of messages with terminals are removed from application programs. The use by applications programmers of the facilities offered by the complex of service modules for exchanging messages with terminals will allow unification of operations for input/output of messages on any type of terminal supported by the YeS OS version being used. This will consequently enhance the quality and reliability of the programs being developed and reduce the time spent of developing those elements of an intervctive program that are responsible for exchanging messages with terminals.

This systems software that we developed is oriented to use of the following Unified System telecommunications equipment: the MPD-3 communications controller together with the TA-1 and SA-2 adapters, unswitched medium-rate start-stop and synchronous data transmission channels, the YeS7906 and YeS7920 (local and remote) video terminal stations and the Videoton-340 video terminals. This equipment is used to set up a terminal network with a radial structure with centralized control effecttd by the message and terminal network control software. Users interact with the computer primarily by using the YeS7066, YeS7927 and Videoton-340 displays and the DZM-180 and YeS7936 matrix printers intended for producing hard copies either from the display screen or directly from the computer.

8545
CSO: 1863/179

UDC 621.398:681.3

KOLOS DATA TELEPROCESSING SYSTEM ON UNIFED SYSTEM COMPUTERS

Kiev UPRAVLYAYUSHCHIYE SISTEMY I MASHINY in Russian No 2, Mar-Apr 82 (manuscript received 30 Oct 78, after revision 5 Oct 81) pp 75-78

[Article by Petr Matsovich Ivanov, candidate of physicomathematical sciences, Kabardino-Balkarskiy branch of the VPTO ASU "Rossel'khoztekhnsistema", Nal'chik]

[Excerpt] Introduction. The problems of optimization and efficient operation of agro-industrial formations in our country are being studied both at the level of national economic and republic agro-industrial complexes (APK) [AIC] and at the level of production associations and enterprises [1-5]. Also being studied are the problems of optimization of AIC's formed at the level of rural territorial-administrative rayons and oblasts [6].

It should be noted at the same time that inadequate attention is being paid to the problems of creating automated systems for management of the AIC's at different levels. An attempt was made in [7] to overcome this shortcoming; suggested was the concept of territorial-intersectorial ASU's [management information systems = MIS] for the AIC's in the rural administrative rayons, oblasts (ASSR's, krays) and republics, which considers the systems approach to improving the methods and structure of managing all sectors of the AIC's at the various levels and unites all sector MIS's on the base of multiaccess computer networks.

The technology of information processing in the MIS's for the territorially distributed hierarchical facilities in current use, where the stages of getting the data into a computer center are considered independently of the stages of data processing on a computer, has become an obstacle to further development of these MIS's. The same situation has also been observed in the agricultural sector MIS's (for the Minsel'khoz [Ministry of Agriculture], Minvodkhoz [Ministry of Land Reclamation and Water Resources], Goskomsel'khoztekhnika [State Committee for Supply of Production Equipment for Agriculture], etc.). The paradox in this technology is that machine processing of information is possible only after an enormous amount of manual operations on data preparation, which reduces computer capabilities to zero. This fact and the specifics of agriculture and the sectors serving it with a required management pace measured within the limits of an hour and at times even within several minutes (especially during seasonal operations) require development of interactive MIS's in the sectors serving agriculture and in agriculture itself. In other words, is a problem of switching from offline use of computers to setting up new combinations and configurations of computers linked to each other by communication lines. Such a complex of computers and terminals (AP), which makes joint use of hardware,

software and data files, which serves all the ministries and departments involved with agriculture, and which enables connecting the computer and the remote user in the on-line mode, is called a multiaccess teleprocessing system (STD).

Despite the available theoretical results on the questions of analysis and synthesis of computer networks [8-13], the methodology for synthesis of computer networks and data communication networks for MIS's has not yet been developed. Because of this, the experience and intuition of the designers and their knowledge of the object still play a substantial role in designing multiaccess teleprocessing systems and computer networks for specific MIS's along with the use of graphic-analytic, numeric and other methods.

The different stages in design of teleprocessing systems (determining the number and location of terminals, mode of servicing users, determining the capabilities of switched circuits, selecting modems and other equipment, etc.) depend on the structure of the specific object and the nature of activity of the organizations to be served. These can be determined by the MIS designer by using relatively simple analytic computations.

The following is an approach to development of these systems based on the example of the Kolos teleprocessing system.

Selection of Teleprocessing Hardware and Standard Configuration for the Kolos System. Just as for any other teleprocessing system, the Kolos system hardware consists of three main groups: user terminals, data communication hardware and computer interfaces to the communication circuits.

In deciding on system structure, the developers took into account the two main limitations on tht concept of teleprocessing in the first phase of the Unified System, which prevented development of teleprocessing systems in agriculture and in the sectors serving agriculture:
--the lack of terminals that perform certain functions of a concentrator to which remote consoles could be connected (such terminals are necessary, for example, in major trade-storage centers, where remote consoles are concentrated in warehouses, or in repair plants where consoles are placed in shops, as well as in kolkhozes and sovkhozes with remote consoles in brigades and sections);
--the lack of programmable terminals which prevented designing teleprocessing [TP] systems for MIS's in sectors serving agriculture and for agriculture itself, where it is necessary to genervte the initial documents that require performance of arithmetic operations (when such a terminal is not available, initial processing of input data falls to the computer, which takes up a good part of circuit and machine time and limits substantially the TP system capabilities).

In developing the Kolos TP system, also considered was the capability of making use of both first and second phase Unified System TP facilities. Second phase features include the capability data exchange between computers, where second phase computers (of the YeS1035, YeS1045, etc. type) are needed as central (oblast, kray) computers, with which the independent TP systems on the base of the YeS1022 or other first phase computer can operate. In the process, the YeS1022 is linked on the one hand to the central computers through the multiplexer channel and the YeS8403 communications controller, and on the other, through a communications controller (YeS8401, YeS8400) or a grouped adapter created on the base of the adapter [16],

to the set of terminals, which properly implements the shift from a first phase Unified System TP system to the network TP system.

Kolos system terminals were developed with consideration for eliminating the above restrictions on the first phase Unified System TP system and are based on elements in series production (displays, data communication equipment (APD) and printers) [14]. The presence of the APD-MA equipment (see [15]) and the Iskra-2302 minicomputer in the "APD-MA--Videoton-340--Iskra-2302" terminal (for convenience, let us call it the "Nal'chik") (see [14]) eliminates the limitations mentioned above.

The APD-MA allows parallel connection of up to eight consoles consisting of displays and Iskra-2302 minicomputers which are serviced on a priority basis by using one telephone line, which is important for setting up inexpensive multiconsole systems. To set up multiconsole systems when the consoles are located a considerable distance away, the APD-MPP is used. This equipment enables half-duplex exchange of alphanumeric information between terminals over dedicated or switched telephone lines and physical cable and overhead wires. Reliability of information is one garbled character per 10^6 sent with a probability of garbling of one bit in a channel of 10^{-3}. In sending data 14 km, the APD-MPP affords a transfer rate of 1200 bits/s, and 2400 for 10 km, 4800 for 7 km and 9600 for 4km.

Developed in the Kolos system were units to interface the APD-MA to the APD-MPP and the APD-MPP to a display and the Iskra-2302 minicomputer. The APD-MPP affords code-independent communication between the central station and eight terminals (a terminal consists of a display and the Iskra-2302) through alternate polling of requests for transmission of information performed by the central station.

Including displays in the "Nal'chik" terminal is dictated by the large volume of information that occurs in the sectors serving agriculture and in agriculture itself, and by the different forms of data representation in the interactive mode.

Another advantage of the "Nal'chik" terminal is the capability of obtaining a copy of a source document on perforated tape. This is a factor in the reliability of the data communication network when the communication channels are out of order. Connectable to this terminal in addition to the Videoton-340 are the domestic VTA-2000, EP-1 and EP-2 displays.

In the Kolos system standard configuration (fig. 1), the SM-1 minicomputer is used as a communications processor at the top level. The SM-1 is equipped with special software and interfaces to the Unified System computer and the APD-MA. At the "rayon-farm" level, the grouped adapter (GRAD), based on the adapter described in [16], is used as the MPD [communications controller].

Information is exchanged between the "Nal'chik" consoles and the computer at the initiative of both the operator and the computer. A terminal operates in the half-duplex mode with the computer and a channel is not kept busy for a response to some message input from one of the eight consoles operating on a channel; another input message can occupy the channel without waiting for the response to the first one (in the process, priority is given to an output message).

When an operator initiates sending data, the data are output to the display screen, checked and sent to the computer. When the subscriber has to send data having a

Key:

1. Unified System computer
2. SM-1 minicomputer
3. GRAD [group adapter]
4. APD [data communication equipment] - MA
5. APD [data communication equipment] - MPP
6. up to 14 km
7. "Nal'chik" type terminals
8. symbols:
9. SM-1 interface to APD-MA
10. SM-1 interface to Unified System computer
11. APD-MA interface to APD-MPP

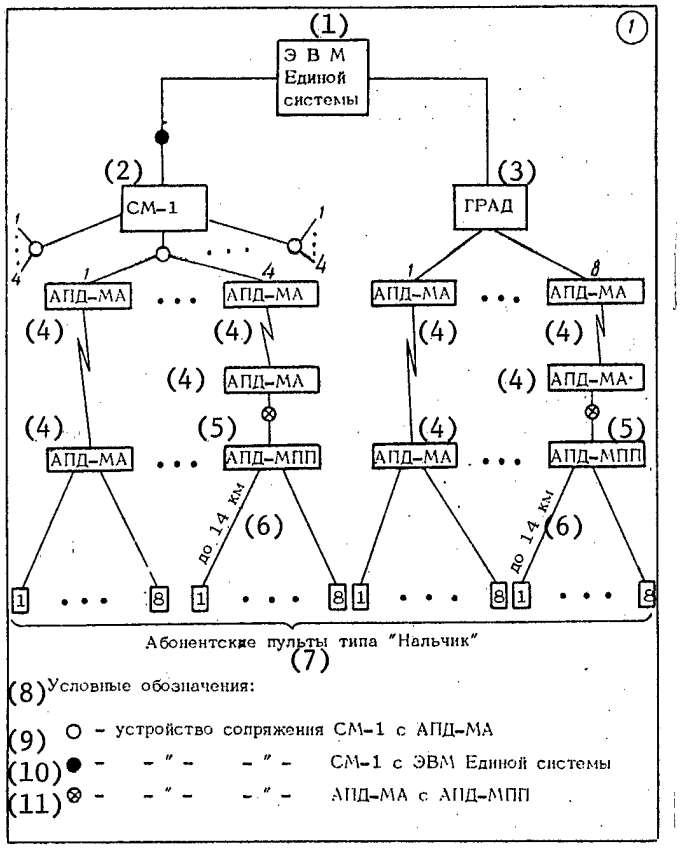

Fig. 1. Kolos system standard configuration

complex structure, he does so by using the Iskra-2302 computer (in the process, the data are automatically displayed on the display scrten for checking prior to sending it to the computer). Operation of a "Nal'chik" terminal is described in more detail in [14].

Computer response time in the "query-response" mode does not exceed 10-15 seconds when operating with a large information file. The answer is displayed on the screen and also printed out whtn required (at the operator's command). Work on including the "Daro-1720" minicomputer with a memory instead of the "Iskra-2302" in the "Nal'chik" terminals has been completed; this will sharply increase the capabilities of this terminal.

Application programs that implement specific user requests are needed for specific Kolos system applications.

There are four terminal classes in the Kolos system.

A first class terminal (AP_{1K}) includes one or more medium throughput (YeS1022 type) Unified System computers. Such terminals may be located at major repair plants, oblast trade-storage centers, and at rayon agricultural associations. Connected to these computers at the lower end are "Nal'chik" type interactive terminals, and at the upper end the AP_{1K} go out in the batch mode through the MPD-3 (YeS8403) to the central processors in the system.

A second class terminal (AP_{2K}) includes one or more minicomputers. These terminals are intended for inter-farm associations, agro-industrial complexes, rayon divisions of the State Committee for Supply of Production Equipment for Agriculture,

major facilities of the Ministry of Land Reclamation and Water Resources, medium size centers for the State Committee for Supply of Production Equipment for Agriculture, etc. These terminals have minicomputers from the SM [System of Small Computers] series [17] that have the capability of communicating with computers at a higher level, operating in the real-time and time-sharing modes, automatically acquiring information on equipment status and controlling equipment by programs.

Third class terminals (AP_{3K}) include the "Nal'chik" type (installed at trade centers for the State Committee for Supply of Production Equipment for Agriculture, kolkhozes, sovkhozes, etc.) and the AP-61 and AP-63 (installed in administrative agencies for ministries, departments, trusts, associations, etc.).

Fourth class terminals (AP_{4K}) include the AP-70 type (YeS8570) or console information recorders of the RI-6401 and RI-6402 types together with the clustered RI-3901 and RI-8901 (P-2311) [18], as well as terminals with a set of remote sensors. Information recorders enable concentrating information in shops directly from the work stations of the foremen and machine operators, and initial processing and checking of data; terminals operating with sensors are used in land reclamation and water resource systems.

In time, modular systems with built-in microcomputers [19] will also be used as communication processors in the Kolos TP system. Microcomputer selection will depend on the availability of developed software.

Data Communication Network Topology. At present, in determining a TP system configuration, one cannot but consider the "oblast--rayon--farm" telephone circuit configuration that has been established in our country and which has largely radial topology. In the early period of Kolos system development, this dictated the radial (centralized) topology (fig. 2). As communication circuits, data TP equipment and number of system users increase, the Kolos system will evolve into the mixed (radial-ring [20]) topology (fig. 3).

Leased (dedicated) circuits are used in the Kolos system under the conditions of shared use by all agricultural ministries and departments.

The distributed automated data bank system (RABD) for the Kolos is built on the hierarchical principle and is a common network of the data banks located in the central computer system, and at first and second class terminals, and which are linked to each other by data exchange channels and facilities. The RABD structure and its functions are largely determined by the structure of the Kolos system and the problems being solved with it. The network of data banks located at the first and second class terminals together with the data exchange channels and central automated data bank, located in the central computer system and which effects overall management of the functioning of the entire RABD, operates as a common system in the interests of all enterprises and organizations under the agricultural ministries and departments.

The specific implementation of the RABD at each terminal and second class terminal at the rayon level must be generated as a function of the problems to be solved and the communication facilities; being developed for this is a language to describe the status of the RABD common to the entire data communication network, a data

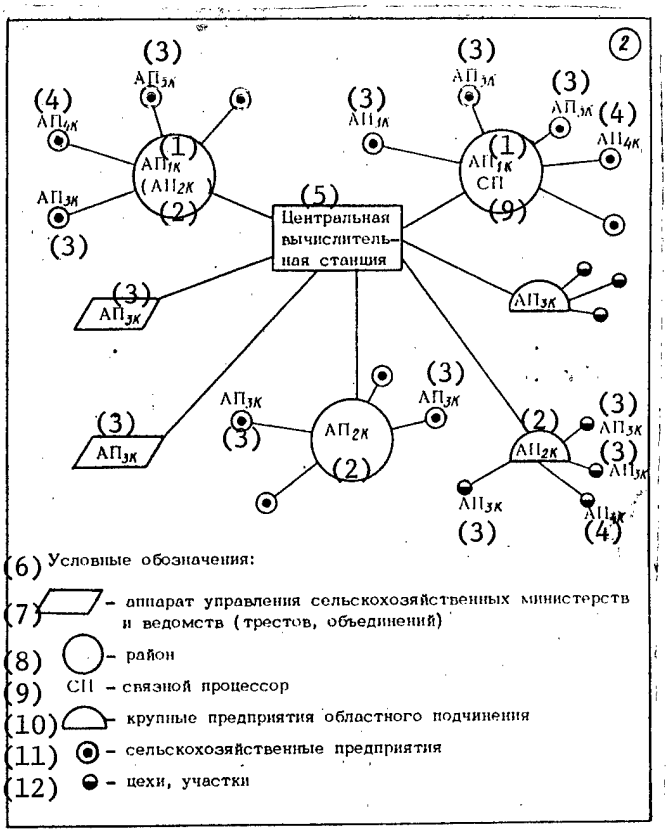

Fig. 2. Kolos system, radial (centralized) topology

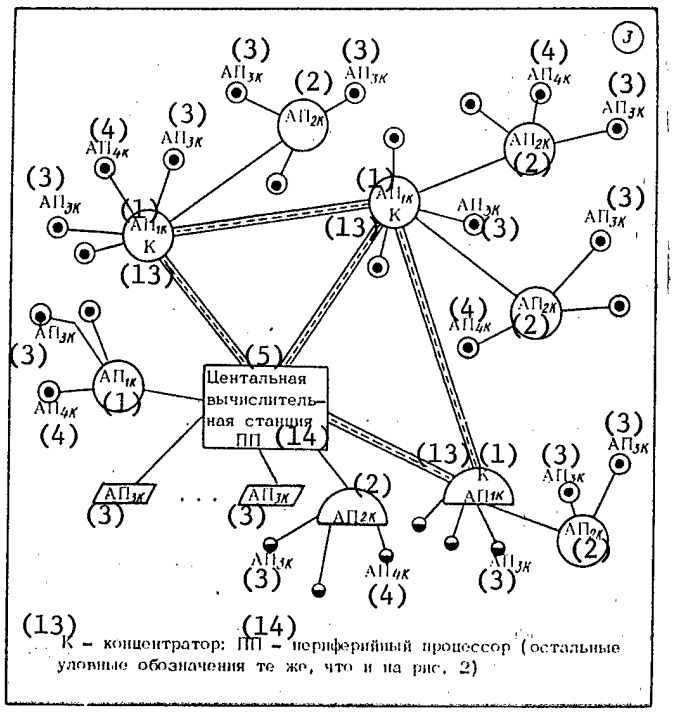

Fig. 3. Kolos system, mixed (radial-ring) topology

Key:

1. AP_{1K} [first class terminal]
2. AP_{2K} [second class terminal]
3. AP_{3K} [third class terminal]
4. AP_{4K} [fourth class terminal]
5. central computer station
6. symbols:
7. agricultural ministerial and departmental administrative agency (trusts, associations)
8. rayon
9. SP -- communications processor
10. major enterprises, oblast subordination
11. agricultural enterprises
12. shops, sections
13. K -- concentrator
14. PP - peripheral processor (other symbols are same as in fig. 2)

description language and a software package to convert data format into the data base management system [DBMS] language.

Used in the Kolos system is the OKA DBMS; the system-wide network software is based on YeS OS.

This approach to developing a shared use TP system for the agricultural ministries and departments of an ASSR (kray, oblast) is being implemented in the Kabardino-Balkarskaya ASSR. The TP system has now been implemented in one sector serving agriculture, the State Committee for Supply of Production Equipment for Agriculture. Here, the application programs have been developed that manage the input and output of information for the "Nal'chik" remote terminals: a program for online data acquisition, query programs, a dialog program and a remote access program.

The Kolos system has now assumed fully the function of online distribution (in the interactive mode) and issuing of spare parts coming in and stored at the republic trade center for rayon users without human intervention. In the near future, it will effect online production planning of the activity of the rayon divisions of the Sel'khoztekhnika [Agricultural Equipment Association] in the mode of interaction with the rayon.

8545
CSO: 1863/179

LANDMARKS OF YEREVAN SCIENTIFIC-RESEARCH INSTITUTE OF MATHEMATICAL MACHINES

Yerevan KOMMUNIST in Russian 20 Apr 82 p 1

[Text] The flagman of computer technology in the republic is the Yerevan Scientific-Research Institute of Mathematical Machines, which has been awarded the Red Challenge Banner of the Central Committee of the Communist Party of Armenia, the Council of Ministers of the Armenian SSR, the Council of Unions and the Central Committee of the Komsomol of Armenia eight times in a row. The institute has been awarded the Red Labor Banner. Here are its glorious landmarks:

- 1957-1960: A series of original pieces of equipment and the first computer successfully underwent state testing. The first among these was the computer "Aragats." By its technical characteristics and design-engineering level it was in the ranks of the best in the country.

- Work on creation of the "Yerevan" and a series of other specialized machines was started in parallel. In the computer "Yerevan" circuit-structures and design-engineering solutions that were accepted in the previous computer were used.

- From July, 1959 until the end of 1960 the computer "Razdan-1" was designed: the first semi-conductor machine created in the country. After a year a second, improved model, the "Razdan-2" was put into operation.

The creation of the "Razdan-1" and "Razdan-2" computers marks the beginning of a new stage in the standing of the institute and in the hastened development of the computer industry in the republic.

- In 1963 the design of the "Razdan-3" computer was begun. By its engineering characteristics, productivity, developed logic, wide choice of peripherals and storage devices in the largest main memory - it was one of the best machines in the country.

In 1965, the "Razdan-3" computer was designed, manufactured, and passed state testing.

- An important direction in computer technology is the creation of small machines. The small computer "Nairi-1" was designed and put into series production, and was intended for engineering calculations. With small dimensions and a low cost, simple and accessible is use, not requiring special preparation - this computer found wide application.

- The design of the first computer using integrated microcircuits was begun at the Institute in 1967. In 1970 the first such computer was designed, tested, and put into production in the country.

- One of the most important machines that was worked out - the "YeS-1030" computer - is a model of the Unified System of third-generation computers, worked out by institutes in the USSR and the socialist countries. A series of new and complicated tasks was solved by the collective of designers. On the basis of an "YeS-1030" computer a computer complex which brought together two computers with a common memory field was worked out.

- The creation and introduction into production in a short time of small computers of the "Nairi" series, the "Nairi-4", the "Nairi-4 Arm" and the "YeS-1045", was possible thanks to an automated design system. The work of the institute has entered production and has received high evaluations from a state commission.

9946
CSO: 1863/156

PUBLICATIONS

TABLE OF CONTENTS FROM JOURNAL 'CONTROL SYSTEMS AND MACHINES', MARCH-APRIL 1982

Kiev UPRAVLYAYUSHCHIYE SISTEMY I MASHINY in Russian No 2, Mar-Apr 82 pp 1-2

8545
CSO: 1863/179

- END -