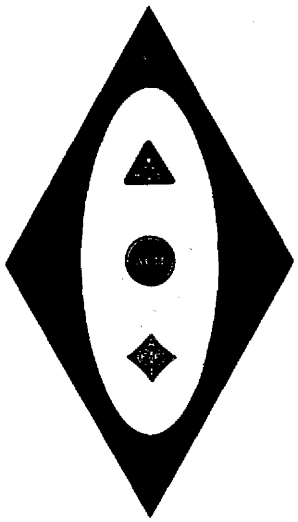
Proceedings of the

T-85

WESTERN JOINT COMPUTER CONFERENCE

February 7–9, 1956 San Francisco, Calif.



Sponsors:

THE AMERICAN INSTITUTE OF ELECTRICAL ENGINEERS

Committee on Computing Devices

THE ASSOCIATION FOR COMPUTING MACHINERY

THE INSTITUTE OF RADIO ENGINEERS

Professional Group on Electronic Computers

Printed in the United States of America

Price $3.00

PROCEEDINGS OF THE
WESTERN JOINT COMPUTER CONFERENCE

PAPERS PRESENTED AT
THE JOINT ACM-AIEE-IRE COMPUTER CONFERENCE,
SAN FRANCISCO, CALIF., FEBRUARY 7–9, 1956

Sponsors
THE AMERICAN INSTITUTE OF ELECTRICAL ENGINEERS
Committee on Computing Devices
THE ASSOCIATION FOR COMPUTING MACHINERY
THE INSTITUTE OF RADIO ENGINEERS
Professional Group on Electronic Computers

Published by the
American Institute of Electrical Engineers
33 West 39th Street, New York 18, N. Y.
for the
Joint Computer Committee

ADDITIONAL COPIES

Additional copies may be purchased from the following sponsoring societies at $3.00 per copy. Checks should be made payable to any one of the following societies:

AMERICAN INSTITUTE OF ELECTRICAL ENGINEERS
33 West 39th Street, New York 18, N. Y.

ASSOCIATION FOR COMPUTING MACHINERY
2 East 63d Street, New York 21, N. Y.

INSTITUTE OF RADIO ENGINEERS
1 East 79th Street, New York 21, N. Y.

Foreword

The ninth Joint Computer Conference which was the fourth Western Meeting under joint sponsorship, was held for the first time in San Francisco this year. The change in the site for this Western Meeting from Los Angeles to San Francisco bespeaks the increased activity and interest in computers on the West Coast.

More emphasis was placed this year than at the past few conferences on the strictly engineering phases of computers. Thirty-eight technical papers were presented, with parallel sessions being used on four occasions. As has become the custom, exhibits of computer systems and components formed a part of the conference program together with a number of inspection trips to local computer installations. The registration of over 1,000, as well as members of the public, was able to inspect exhibits in 39 booths prepared by 25 companies prominent in the computer field.

Oliver Whitby
Conference Manager

Joint Computer Committee of the AIEE—IRE—ACM

Secretary-Treasurer: M. M. Astrahan

Representing AIEE

ROBERT R. BENNETT
J. G. BRAINERD
D. C. ROSS
P. L. MORTON

Representing IRE

DANIEL HAAGENS
W. F. GUNNING
J. R. WEINER, Chairman—East
OLIVER WHITBY, Chairman—West

Representing ACM

C. W. ADAMS
J. BARNES
D. H. LEHMER
H. F. MITCHELL, JR.

Ex-Officio Representatives

E. L. HARDER, AIEE
J. H. FELKER, IRE-PGEC
A. S. HOUSEHOLDER, ACM

Conference Management Committees

Conference Manager

OLIVER WHITBY.....Stanford Research Institute

Secretary-Treasurer

J. W. HAANSTRA.....IBM Corporation

Technical Program

B. J. BENNETT.....Stanford Research Institute

Publications

A. S. HOAGLAND, Chairman.....University of California, Berkeley
GORDON MORRISON.....University of California, Berkeley

Exhibits

G. E. HULSTEDE, Chairman.....Pacific Telephone & Telegraph Co.
H. K. FARRAR.....Pacific Telephone & Telegraph Co.

Publicity

D. C. HOLMES, Chairman.....Shell Development
BILL EISENLORD.....Shell Development

Registration

H. W. HARRISON, Chairman.....NACA Ames Aerodynamics Laboratories
ALAN VOORSANGER.....Sperry-Rand Corporation
JACK MAUGHMER.....University of California, Berkeley

Hotel Arrangements

DEXTER STONER, Chairman.....Pacific Gas & Electric Co.
DENNIS FINNIGAN.....Stanford Research Institute
A. M. PETERSON.....Stanford University

Trips

L. C. NOFREY, Chairman.....University of California, Radiation
WARREN STUVEN.....University of California, Radiation

LIST OF EXHIBITORS

AIRCRAFT-MARINE PRODUCTS, INC.	Harrisburg, Pa.
AMPEX CORPORATION	Redwood City, Calif.
BECKMAN INSTRUMENTS, INC., Berkeley Division	Richmond, Calif.
BENSON-LEHMER CORPORATION	Los Angeles, Calif.
BURROUGHS CORPORATION	Philadelphia, Pa.
BURROUGHS CORPORATION, Electronics Instrument Division	Philadelphia, Pa.
CANNING, SISSON & ASSOCIATES	Los Angeles, Calif.
DENNISON MANUFACTURING COMPANY	Framingham, Mass.
E E C O PRODUCTION COMPANY	Los Angeles, Calif.
ELECTRO DATA CORPORATION	Pasadena, Calif.
ELECTRONIC ASSOCIATES, INC.	Long Branch, N. J.
ENCYCLOPEDIA BRITANNICA	San Francisco, Calif.
FRIDEN CALCULATING MACHINE COMPANY, INC.	Los Angeles, Calif.
HUGHES AIRCRAFT COMPANY	Culver City, Calif.
INTERNATIONAL BUSINESS MACHINES CORP.	New York, N. Y.
LIBRASCOPE, INC.	Glendale, Calif.
LITTON INDUSTRIES	Beverly Hills, Calif.
MOSELEY (F. L.) COMPANY	Pasadena, Calif.
MOXON (G. E.) SALES	Culver City, Calif.
NORTH AMERICAN AVIATION, INC.	Los Angeles, Calif.
REA (J. B.) COMPANY, INC.	Santa Monica, Calif.
REMINGTON-RAND CORPORATION	New York, N. Y.
SPRAGUE ELECTRIC COMPANY	North Adams, Mass.
SOROBAN ENGINEERING, INC.	Melbourne, Fla.
THE PACIFIC TELEPHONE AND TELEGRAPH CO.	San Francisco, Calif.
THE AMERICAN INSTITUTE OF ELECTRICAL ENGINEERS	New York, N. Y.
THE ASSOCIATION FOR COMPUTING MACHINERY	New York, N. Y.
THE INSTITUTE OF RADIO ENGINEERS	New York, N. Y.

Contents

Computers—From Youth to Manhood

NORMAN H. TAYLOR

YOU are very kind to honor me with the invitation to speak here today, and it is a pleasure to be here. I always appreciate a good excuse for leaving my overcoat and galoshes behind in Boston and coming out to sunny California. Furthermore, we are beginning to realize that although Boston is still of course the hub of the universe, nevertheless there are some pretty good ideas generated out here on the West Coast; and I hope I shall not return to New England without a few dozen of them in my carpet-bag.

At the first Joint Computer Conference in 1951 at Philadelphia, the computer industry was turning a corner. You might say that after a well-protected infancy, it was putting on short pants and getting outdoors and the neighbors were beginning to be aware of it. The machines, like some children I know, could be made to put on their "company manners" when their rich uncles came around, but this was all too often the result of intensive grooming and stern disciplinary measures behind the scenes. Our audience, while sympathetic and hopeful, still could not help feeling skeptical, and some may even have feared that we were raising a little monster who would turn out to be more of a liability than an asset.

Five rapid years of growth have served to vindicate our claims of 1951. I could easily spend an hour giving you chapter and verse on this, but suffice it to say that more than 75 large machines are in full-scale useful operation, and hundreds of smaller machines are doing important jobs for office and industry. The attendance figures at these professional conferences should be some sort of index; there were 880 of you at that first conference in 1951, now we are holding two conferences a year, and the 1955 Eastern Joint Computer Conference alone had approximately 2,000 people in attendance. This Western Joint Computer Conference and its counterpart, the Eastern Joint Computer Conference, will undoubtedly set a new record.

Thus, I think we can say that the computer business is getting out of its childhood, and putting on long pants. Adolescence is upon us, and although some of us, I am sure, look back with a certain nostalgia on the happy childhood days, nevertheless, like good parents we cannot shirk the problems of the present, and must try as best we can to prepare for the future. By and large, we are proud, you and I, of what we've done so far. But no engineer worth his salt is going to devote his time to admiring his past work. What the good engineer lives for, the air he breathes and the meat he eats, is the challenge of new problems. It is not too much to say that without new problems he suffocates and starves, and professionally he might as well be dead. If there were an engineer's Bible, I think this ought to be one of the two great commandments, so, using this text, my little sermon today is going to deal with the problems ahead of us. Please note that I'm presenting problems and asking questions, not trying to give you the answers. Even if I thought I knew a few of them, I'd be spoiling your fun, and that would be a dismal thing to do! I'll save my clairvoyance act for private chats, where I won't be embarrassed by seeing it in print and having to eat it five years later. You fellows are the ones who will have to answer these questions, just as today you have given us answers to some of the questions asked five years ago.

Let me return now and then to that notion of looking at computers as children that we've raised—it isn't really a bad analogy, if I don't squeeze it too hard; you know how children outgrow things, first their shoes, then their pants, then their coats, and so on. Or how a boy's mind in a way outgrows his body, and he imagines himself beating up all the tough guys in the neighborhood and fascinating all the pretty girls. Well, our computers have had and are still having just this sort of growing pain, the parts like arithmetic element, memory, and terminal equipment don't keep pace with each other, and we haven't yet got a mature, harmonious, balanced system. Until recently, our worst shortcoming has been the size, reliability, and speed of central memories; and we have put a lot of effort into improving them. The magnetic-core memory which has come out of this is now fairly well accepted as the central high-speed element of present-day machines, and is probably the largest single change of these past few years. Even so, we haven't quite caught up with arithmetic element speeds; the logical structure of most machines has been aimed at using memory time and capacity to the hilt. Elaborate buffer memories serve the purpose of emptying and filling the high-speed memory, and tricky instructions use other hardware to do jobs that might be done in the central memory if its time weren't so valuable.

At that, we haven't yet exploited magnetic-core memory to the full; the 64 by 64 memory plane is now pretty common, but one naturally asks the question, how much bigger can you build it? Preliminary work leads us to believe that the 64 by 64 plane can be extended to 128 by 128 and probably to 256 by 256 without any serious loss in speed or reliability. If this expectation becomes a reality, we shall have a 256 by 256 by 36 memory, that is to say, 2.5 million bits of storage with 6 to 7 microseconds random access. Surely this will be a handy tool for the future. How shall we use it? It certainly ought

---

Full text of the keynote address presented at the Western Joint Computer Conference, San Francisco, Calif., February 7, 1956.

NORMAN H. TAYLOR is with the Lincoln Laboratory of the Massachusetts Institute of Technology, Lexington, Mass.

The author wishes to acknowledge the helpful suggestions of W. A. Hosier in the preparation of this address.

to affect our present concept of providing large buffers feeding a small high-speed memory; it has about the capacity of eight conventional magnetic drums. Does this mean that drums are obsolescent?

But can the memory designer relax? Is he going to be free of demands for more speed and capacity from the direction of the arithmetic element? Vacuum tubes have been for several years the principal limitation on arithmetic element speed: we have pushed adders up to 1 or 2 megacycles, and with multipliers it is commonly felt that a law of diminishing returns sets in somewhere in the region of 1/2 to 1 microsecond per bit. I think the memory people, with the 6-microsecond memories, have provided, potentially at least, a good match for such circuits in speed and reliability.

But what about the future? Surface barrier transistors already give promise of operating at 5 megacycles in arithmetic and switching operations. What will come with the new gaseous diffused or drift transistors? I'm afraid the answer is inevitable: Memory will lag behind and the race will be on again. Why all this speed? Do we really need it? We are already being pressed to design larger and faster machines to tackle problems that are bulky and complex and have to be solved quickly. One way or another, I'm sure we'll try it. But if we had speed an order of magnitude over what we've got now, couldn't we do more time-sharing and substitute speed for equipment? Surely if something like this can be done, machines will be simpler, smaller, less expensive, and more reliable. This is a problem for the logical designer.

I have dwelt here on a few aspects of the internal consti ution of computers. In our analogy to the child, these would be his health, growth, and internal development. These are going well, with good momentum. I don't think we have any cause to fear that these children of ours are sickly or stunted. But as we all know, this is, if anything, the small end of the problem of rearing children: as they grow up, they leave more and more their old sheltered environment; they must be educated, they must (usually) do useful work, they must gradually accept more responsibility. I'm sure this isn't all, but these three requirements give me a convenient handle for the remaining things I have to say. Let's talk about this subject of useful work.

You can divide work roughly into two kinds: the kind that deals with symbols which you might call "white-collar work," and the kind that deals with matter and energy, which I suppose you would call manual labor or something like that. The kind of work that deals with people I don't feel has got into the picture yet. When we ask a computer to do the first kind of work, we call it data-processing; when it does the second kind, the current terminology would seem to be "automation." I am aware that the last word is often used more loosely, and that the dividing line between the two kinds of work is fuzzy in spots, nevertheless the division is a help in thinking about the problems, because the problems in the two cases are essentially different, and since there is no better word for the second kind, call it automation.

Data-processing deals with symbols, and symbols for the most part are discontinuous, discrete things: they come in chunks, and are grist for the digital computer mill. We are getting high-speed printers to grind them out. If we can get high-speed readers of print to grind them in without going through the clumsy media of punched cards or paper tape or the volatile and intangible medium of magnetic tape, we shall have gone a long way toward making the "white-collar" boys happy. So much for data-processing. Let's consider automation.

This field sometimes is referred to as real-time control, and it is quite a different story from data-processing. It seems to me to be the most challenging area of the future, heavy with problems, but also glittering with promised rewards The newspapers say it's here already, quoting from Mr. Kenney in the *Christian Science Monitor*:[1]

> "In this contemporary fantastic era of research and automation, people today are better fed, clothed, housed, heated, cooled, propelled, entertained, and defended than at any time in all history.
>
> The Alice in Wonderland, razzle-dazzle of new methods and processes are zooming living conditions and industrial activities into a never-never land undreamed of a decade ago and even today are leaving the most dramatic prognosticators almost baffled in attempts to describe what is coming up in the future.
>
> Automation is in its infancy. Yet the pattern is taking form. Transportation demands magic controls. Airplanes both commercial and military are speeding through the skies so fast that human reaction is not keen enough for proper operation. Electronic, radar, and automatic controls are guiding the giant ships through storm and fog, over land and sea ...."
>
> COMPUTER SETS PACE
>
> "One of the most dramatic and important developments in the automation field is the computer. It is often called the electronic 'brain.' The computer can do just that at enormous speeds. It can add, subtract and multiply, etc. ..... "

This puts me in mind of an experience we had with radar during the war, which some of you may have heard of: The Admiral on one of our battleships, flagship of a task force in 1942, didn't put much stock in the reports of his radar man. But one night the radar man called up to flag pilot, saying, "Carrier Saratoga 2,000 yards off the starboard bow." The Saratoga in some aspects gave a characteristic double-humped blip on an *A*-scope that an experienced operator could easily identify. But the Admiral knew different. "You're crazy as Hell," he said, "The Sara's been off the port bow all night." But the radar man stuck to his guns, so the Admiral finally had a blinker message flashed, asking the ship to identify herself, and, sure enough, damned if it wasn't the Saratoga off the starboard bow. Well, that really converted the Admiral to electronics. So much so, that when the radar man a half hour later said, "Destroyer 4,000 yards dead ahead," the Admiral asked, "What's its number?"

I'm sure you can see the moral for us computer engineers in this little tale. If we're not well aware of our limitations and can't explain them intelligibly to others, we're likely from time to time to be asked to bite off more than we can chew. And believe me, in this automation field we have plenty of limitations at the present time.

For one thing, the computer is no longer operating in a vacuum. It's got to operate in somebody else's system, and if you just try to plop the computer into the middle of the system the odds are about 99 to 1 you can't make it work. You've got to study the whole problem and probably re-engineer a large fraction of the system before you reach any sort of harmonious solution. And this process has to be detailed and careful, not general or hasty. For the engineer on this sort of job, a fast, reliable central computer is not enough: he must take off the blinders and look at a whole new group of problems; in short, we need systems engineers.

For another thing, the sort of information that the system feeds to the computer is not usually a nice chopped-up bunch of easily-digitized symbols. It is on the other hand usually a nasty continuous rope of information from something like a thermocouple, a strain gauge, or a radar set; these analogue inputs are apt to be as embarrassing as a long continuous piece of spaghetti if we can't devise ways of handling them. What I'm asking you is this, have we put enough effort into analogue-to-digital conversion devices?

Not only does our young computer have to learn how to simplify information so that he can comprehend it, he also has to learn to be discriminating, to tell the truth from lies, to tell the important from the trivial. This is the noise problem. It's almost negligible when you're dealing with symbols, but when sensitive analogue measuring instruments and transmission lines from remote places come into the picture, it can become pretty bothersome. Communications engineers over the years have devised means, such as statistical processes, for coping with noise. We are likely to need closer liaison with them, and doubtless will have to help to extend their work.

Finally, of course, this computer out in the world of work not only requires eyes and ears and a central nervous system, as it were; it also must have muscles and tools. The azimuth and elevation of a gun, the force of a hammer blow, the degree of opening of a valve, the current in a welding arc—here we are again with physical variables, continuous functions, analogue devices. So here we require digital-to-analogue converters and servomechanisms. Are we attacking these problems as hard as we ought to be?

Return for a moment to the problems of the adolescent child, in particular, education. For a digital computer, this means one thing: programming. This area of programming is a complex problem. I would like to be able to make a few sage comments and dispose of it quickly, but I cannot. I have talked with several program people in an attempt to define just what basic problems they have which might be relieved by machine organization or electronic aid. I have been unable to recognize any single great weakness but here are some of the problems that could be relieved.

In an automatic control system the efficiency desired in a program is high. Many parts of programs run over and over again. The program must be written to accomplish its results in a period of time compatible with the demands of the system which is under control.

The program must be versatile; it must take into account the human monitor when he acts, yet proceed according to the established routines when he does not. It must be flexible. As we learn about new variables pertinent to a given problem we advertise that our general-purpose computer can take these into account by a simple change in program. The program must be written to allow such program changes without a complete rewrite for the problem.

The cost of such programs is high, and time to get them is long. It may surprise you to learn that, in general, the cost of programming a computer for an automation job is roughly the same as the cost of the machine. The cost is justifiable and defendable, but I feel sure that I should ask the question: What are we going to do about simplifying the process? Some advances have been made by using the computer itself to do the bookkeeping tasks and program assembly, register assignment, and the like. What else can it be used for?

In the strictly mathematical area the language needed to express procedure to the machine is fairly well understood and can be generalized and automatized. This is not true in all areas. Can we learn enough about the problems in automatic control to express the variables so that the machine can understand without being given every detail?

Our childhood analogy here becomes tantalizing. As the child grows he learns. After having solved one problem he uses the previous knowledge to attack a new problem. Can we design a machine to learn? Enough work has gone into this to show that the road is hard. The present crop of computers are fast and frisky, but they have the intelligence of an earthworm. I shall not pursue this further, lest I be quoted to my sorrow.

When I compared our situation with these fledgling machines of ours to that of parents with children about to be thrust out to take their place in the adult social fabric, I mentioned one last aspect of the situation; the need for gradual increase of responsibility. No one expects a callow youth to be running much of a show on his own; for a long time he inevitably relies on the supervision and judgment of more experienced hands. In the computer's case, the experienced hands can only be human beings, and the early automation systems are going to require extensive human monitoring. To be sure, in the fullness of time we may hope to build such sound judgment into some of these systems that they can tick along unattended all by themselves, but that day, the day of the truly automatic "no hands" system, is not yet. The need to know what is going on is more than just curiosity; it is part of the evolution towards complete automation and provides the means of correcting and improving our understanding of present concepts. Human judgment is needed in the more complex 1956 central systems for several reasons. First, in order to be completely automatic one has to understand a process with all possible contingencies well enough to tell a computer how to respond in any combination of events. Second, many people have to be confident that this situation does,

in fact, exist. There are other problems too, but these two alone demand an adequate monitoring system and my question to this conference is: How do we provide adequate monitoring systems to help our computer through adolescence?

How can the important Charactron and Typotron display tubes which are now available best be used on this problem? More important: What information must be presented to the human monitor, and in what form? Have you looked in a modern airplane cockpit lately? This is a good example of the monitoring problem with literally hundreds of meters, scopes, lights and knobs. We have so much information to give to the monitor that we ask: How can he absorb it all? Once he has received intelligent and accurate information and made a decision he must make this known to the system. What techniques can be used for this? The famed pushbutton certainly is effective, but has limitations of speed and accuracy. Certain electronic aids, such as the photoelectric pickup, have shown promise. Here is an area where new ideas and ingenuity are welcome.

Could we possibly consider using human voice to talk back to a computer to tell it what to do? How difficult would this be? The telephone people are considering voice-operated dialing. Why not talk to the computer in some similar way?

Another suggestion, less ambitious, proposes that a manual typewriter be used and the computer be programmed to decode the English language for its instructions.

Conclusion

I have posed some difficult problems to accomplish the education, and to increase the usefulness, and responsibility of our adolescent computers.

In 1951, the coming childhood of these infant computers seemed fraught with problems, but we have conquered most of them with determination and enthusiasm. Now in 1956 we have added to that determination and enthusiasm the confidence that the child is healthy and growing and fairly well accepted in society.

If you step back and look at it with a prospective of a few hundred years it seems to me that this being in on the childhood and adolescence of the digital computer art is a rare privilege which most of us here are sharing. Haven't you ever wished that you'd been living back in the days when some of the great sweeping syntheses were made that changed man's viewpoint and ways of thinking? I mean, for example, Faraday's discovery of magnetic induction, Maxwell's mathematical handling of radiation, Descartes' analytical geometry or Newton and Leibniz's early calculus, or, in our own time, relativity, quantum theory and all that has proceeded from them. There was always a time, but it had to be just one time and for rather few interested people, when these concepts were hot off the griddle and were the meat of red-blooded arguments, then, for all time to come, they got taken for granted and salted away in textbooks. Now I'm not flattering myself into believing that digital techniques are as far-reaching as Maxwell's equations (or as neat either), but I do believe they will have many unforeseen consequences that none of us dreamed of when we started playing with them, and for digital techniques the time is now, the people are you, and who knows, Joe Doakes' principle of binary substitution may be something your grandchildren will sweat over in first-year graduate courses!

And thus it is, gentlemen, that I am sure these precocious adolescents of ours are here to stay and to make their presence increasingly felt, it is up to you and me to make first-class citizens out of them.

Reference

1. Automation Widens Vistas of Research Potentialities, H. C. Kenney. *The Christian Science Monitor*, Boston, Mass., Jan. 4, 1956, p. 16.

Gestalt Programming: A New Concept in Automatic Programming

DOUGLAS T. ROSS

Synopsis: In any human endeavor there are three major phases: conception, expression, and execution. Gestalt programming is an attempt to make these three phases as nearly identical to each other as possible with respect to computer programming. In this paper the word Gestalt is used to mean a concept of a task to be performed by a computer. In a Gestalt system of programming, the Gestalt, or idea, is expressed simply and unambiguously in a special language, rather than through the laborious assembling of machine codes, pseudocodes, subroutines, etc. Using a Gestalt system, the expression itself in effect ties together integrated units of computer behavior, which function singly or in interrelation, to achieve the desired effect. The purpose of a Gestalt system is to facilitate the transmission of general ideas as in a conversation, between a human and a computer, so that the maximum use of their respective capabilities can be made.

After presenting the abstract theory of Gestalt programming this paper discusses several Gestalt systems in use today at the Massachusetts Institute of Technology (MIT) and describes briefly the types of computer hardware which are best suited to this application.

As computer techniques have developed over the last few years, there has been a growing trend toward more sophisticated methods for connecting the human, who states the problem, to the computer, which is to solve the problem. Great strides in automatic coding schemes and algebraic coding schemes have been made, and the feasibility and value of these techniques are now well established.

Out of this trend has come, as a natural consequence of the maturing technology, a desire to use computers for solving problems which cannot be completely specified in terms which the computer can handle. This type of problem is united with automatic problem stating, referred to in the foregoing, in the general problem of using humans and computers together to solve problems. In the one case, the goal is to state the problem so that the computer can execute the solution, and in the other case, the goal is not only to state the problem to the computer, but also to assist the computer in obtaining the solution. In both cases, the human and the computer do only those parts for which they are best suited.

If the human and computer are to work together to solve a problem, there must be some means provided for the transmission of ideas or results between the two, since the contributions of each will depend upon the actions of the other. There is no known way in which ideas can be transmitted directly, so that an intermediate stage of expressing the idea in some language is always required. A language consists of two parts; a vocabulary and a set of syntactical rules. An idea is then transmitted by transmitting the expression of the idea; i.e., a sequence of words from the vocabulary. The final stage in the transmission is recognition by the receiver.

A major problem, then, in using humans and computers together is to choose an appropriate language for the interchange of ideas. This language must bridge the gap between the fundamentally incompatible characteristics of the two parties. The human is quick-witted but slow, while the computer is slow-witted but extremely fast.

Most people connected with the computer business seem to be superbly equipped for voluble discussion on any topic. It would therefore appear at first that the language should be chosen for the convenience of the slow-witted computer. Such is definitely not the case, however, because once the computer has been given a language, it becomes a very formidable associate, firing questions and answers at a rate which very quickly becomes alarming to the human. For this reason the first rule in establishing a language is that it must be as natural and convenient as possible for the human to use, not only in the interest of reliability, but for psychotherapeutic reasons as well. Programmers with persecution complexes are already far too numerous.

Since the language is to be used for the transmission of ideas, the most natural way to obtain convenience for the human is to have the language operate entirely at the idea or concept level. In other words, the language should be designed so that general statements can be made easily by the human, with the computer itself filling in the necessary details. This concept should work in the other direction too, i.e., the statements made by the computer to the human should be pertinent digests at the idea level, and not detailed reports.

In order to use the human and computer together efficiently, a statement in the language must lead to direct and immediate recognition and reaction. This may be accomplished by designing the language so that when a statement expressing an idea is made, the receiving party, human or computer, is able to recognize immediately the elemental concepts which are to be united to give the desired idea.

A word already exists which carries all of the connotations of simultaneity and sudden bringing-together of basic units into a single entity or pattern, and that word is "Gestalt" as it is used in the Gestalt theory of psychology. Since there is no single word in the existing computer terminology which works both ways between human and computer, and includes the connotations of being at a high level of communication and implicitly including active execution, the word Gestalt will be borrowed from psychology, and will be used in this paper with very nearly the same meaning in connection with computer programming.

The decision to introduce this new word is not capricious in any way, but is made to facilitate the presentation, and to assist in the establishment of a new emphasis and point of view with respect to the general problem of the interconnections between humans and computers. The actual material discussed in this paper is, for the most part, not new, but the way in which it is discussed is new. This new approach has been found to be very fruitful and clarifying, and is the primary motivation for this paper.

Although the idea of using humans and computers together to solve problems is relatively new, enough examples have been developed by various groups throughout the United States to demonstrate that these techniques show considerable promise. After mentioning a number of applications, (some of which have not yet been tried), to motivate the discussion, this paper considers in some detail the various stages involved in designing computer systems of this type by solving a hypothetical example. The abstract structure of such systems is then outlined, using the example for illustration. Finally several systems in

DOUGLAS T. ROSS is with the Massachusetts Institute of Technology, Cambridge, Mass.

daily use at MIT are described, and some concluding remarks about the probable impact of these techniques upon computer technology are made.

Conversation Versus Communication

A suitable definition of the word Gestalt as it applies to computer programming is that it is a concept of a task. This definition is meant to imply that the Gestalt is not the task itself nor even how the task is to be performed, but merely the idea or concept of that task. For example a Gestalt might be "Integrate $f(x)$," and this idea certainly is not equivalent to the task of integration nor does it tell how the integration is to be performed. The more specific Gestalt "Integrate $f(x)$ using Simpson's rule" still does not prescribe detailed steps of applying Simpson's rule to the particular function in question.

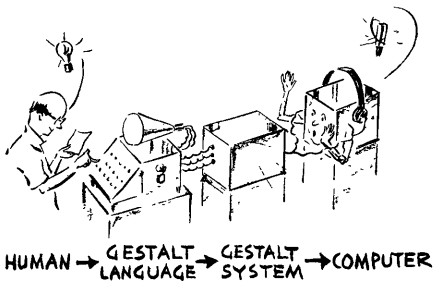


Fig. 1. Communication from human to computer

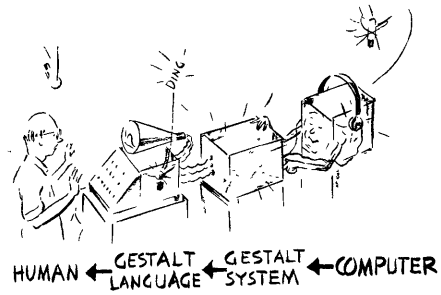


Fig. 2. Communication from computer to human

Fig. 1 shows schematically how a Gestalt is transmitted from the human to the computer. The human simply expresses his idea by pushing buttons which correspond to words or phrases in a special language, the Gestalt language. The Gestalt system then translates the expression into terms which the computer can understand, and the computer can then proceed with the execution of the task.

Fig. 2 shows the analogous situation from the computer to the human. By means of the Gestalt system the computer's idea is expressed in a special language which the human can easily understand. The human is then prepared to perform the task required by the computer.

These two illustrations show the process of communication from the human to the computer and communication from the computer to the human. If the human and computer are to work together to solve the problem, the intermediate languages and translating systems must be designed not merely for the purpose of communication, but for the convenience of fluent conversation. In other words, as Fig. 3 shows, the solution to the problem will, in general, be found only by a more or less extended conversation between the two working as a team, work being divided up so that optimum efficiency and reliability are achieved.

The remarkable flexibility of modern computers makes it possible for them to assume many guises. When more than one role is assumed by a computer in the solution of a problem, it sometimes becomes difficult to talk about the general aspects of that solution because the same mechanism has such different characteristics. This is quite definitely the case when Gestalt programming is discussed, because the computer serves in two capacities; one with respect to stating the problem and one with respect to solving the problem. In this paper the word "computer" usually means the aspect of the computer which is concerned directly with the problem to be solved. The term "Gestalt system" usually means the set of computer programs which aid in the stating of the problem by performing the necessary translation between the Gestalt language and the computer, as shown in the aforementioned illustrations. Often, however, the meaning of Gestalt system is expanded to include the Gestalt language and the physical representation of that language as well, as in the statement, "This problem can be solved by the design of an appropriate Gestalt system." The context makes clear which is intended.

Applications

Before developing the theory of Gestalt programming, several examples of problems will be presented which require or could greatly benefit from the use of human participation. It should be borne in mind, however, that although it is

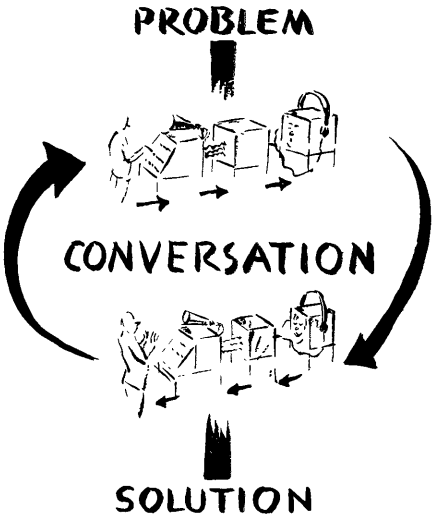


Fig. 3. Solution by conversation

problems such as these which have led to the concept of Gestalt programming, many problems which at present are not considered to require human participation could more effectively be attacked through the use of these techniques.

In almost any control application of computers, whether air-traffic control or automatic factory control, it is necessary to have at least human monitoring with the ability to make sudden changes in the computing scheme. This type of application usually places high priority on reliability and speed.

Large-scale data-reduction problems, basically automatic, often require a human choice between several alternate procedures, a choice dependent in a complex and sometimes whimsical way upon a number of intermediate results. Often partial results can be salvaged from an otherwise worthless set of data, providing appropriate techniques are chosen. By using an appropriate Gestalt system and human participation, these results can be obtained at almost normal processing speed.

Even in strictly computational work a human could greatly expedite the obtaining of solutions if the proper techniques are used. For example, in the solution of complicated partial differential equations or in linear programming problems and game theory, it seems probable that methods could be devised whereby the human could "steer" the computer directly to the solution, rather than obtaining an enormous mesh of solutions, most of which are not of real interest. This type of operation could very well be instrumental in the application of computers to aid in management decisions. A properly designed language would allow executives to converse with the

computer directly and without costly delays.

Perhaps the most intriguing application of human participation in this sense is the use of a Gestalt system in experimental programming, since such a system can be used to generate other Gestalt systems. By experimental programming is meant the programming of a large, complicated program for which the basic steps in the solution are not known. As the programming develops, the programmer must be able to do his design work at the concept level, leaving to the Gestalt system all of the details of translating his growing concepts into actual computer behavior.

Additional applications for human–computer team work can easily be found, but this brief listing should serve to show that the possible uses cover a wide range of problems. This paper does not treat programming details, for these will vary widely for each application, but does try to establish the general problems which are common to all of these applications. In addition to recognizing these problems, a general methodology or plan of attack for solving them is formulated.

Example of Gestalt System Design

The general principles of the design of a Gestalt language are best illustrated by carrying a single example through all of the various stages. Consider the case of an automatic factory whose main features are shown in Fig. 4. Three main processes are involved, followed by an assembly process. These processes are flanked by a raw materials input section and a shipping output section. Besides the primary product, it may be desired to ship directly the outputs from processes two and three. The main duty of operating this factory is to be the responsibility of a computer, but a human operator is to be in charge of setting the requirements for the various stages and overseeing the entire operation.

Present-day computers are not equipped for oral input so that some means other than a spoken language must be used to enable the operator to converse with the computer. Written languages using an intermediate medium such as punched tape or cards have long been used for communicating with computers, but a closer approach to the ease, speed, and flexibility of a spoken language can be achieved by letting each word or phrase which is to be used be represented by a single unique switch or push button. A statement is then "spoken" by pushing appropriate buttons.

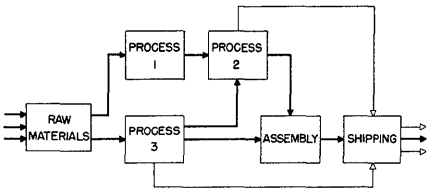


Fig. 4. Diagram of automatic factory

In the automatic factory it will at times be necessary for the operator to refer to each part of the diagram of Fig. 4. The easiest way to do this is to construct a panel with a toggle switch associated with each part as shown in Fig. 5. Now assume that the statements about the factory which the operator must make are of the form: "Increase, decrease, hold, or set the rate, amount, or storage of the input, output, or mixture of the products at the points indicated by switches which are on." The facilities for making such statements are shown in Fig. 5, where the circles connected by lines indicate push buttons with mechanical linkages so that only one button in the column can be pushed at any one time. Provisions should also be made for specifying numerical quantities so that a particular rate or amount can be specified. This facility might be in the form of keyboards or perhaps dials which can be set.

To continue the example, a statement of the foregoing form may be a demand, meaning that the computer is to jeopardize the efficient operation of other sections of the factory, if necessary, in order to comply with the statement. On the other hand, the operator may wish the computer to adjust the factory gradually to comply with the statement, but at all times maintain previous requirements; i.e., the statement is a goal toward which the computer should strive. Finally, the operator may have an estimate from a market survey, that a certain product may be in greater demand soon, so he wishes the computer to adjust the factory toward this tentative condition if it can do so with no loss of efficiency at any point. These three qualifications may be placed on the general statement by pressing one of the buttons labelled demand, goal, or estimate. In other words, the meaning of the statement expressed in the other buttons is modified by these buttons as in a language: e.g., "Run to the store, slowly."

Another whole level of meaning is made possible by considering the computer to be able to simulate the factory as well as control it. The general statement, modified as shown, may be further modified by requesting the computer either to evaluate the effect of the statement, by simulation, or to execute the statement by controlling the factory. This is assumed to be the final modification of the statement so that, the words evaluate and execute are associated with special buttons called activate buttons.

The panel should be wired so that the computer does not look at any of the buttons until one of the activate buttons has been pushed. At this time all of the items necessary to express the idea have been pushed so that when the computer looks at the buttons it is immediately confronted with a complete Gestalt. For example the Gestalt might be, "Evaluate the effect of a demand for an increase in the amount of output from process 2."

The completed panel, shown in Fig. 5, is the physical representation of the Gestalt language for this example. That it does in fact constitute a language may be seen by noting that it does have both a vocabulary and grammatical rules. The vocabulary consists of the various buttons and keyboards, and the rules are contained in the mechanical linkages of the columns of buttons and the fact that buttons modify the meanings of other buttons.

The corresponding language from the computer to the human will not be given in detail. It probably would consist of graphical displays, numerical displays, flashings of indicator lights, and audible alarms. The indicator lights probably would be located in the control panel beside the toggle switches to give them easily understood meanings. For example, the computer might reply to the foregoing Gestalt by saying that if the amount of output from process 2 is increased by demand, the rate of mixture at process 3 must be increased, which will require an increase in one of the raw materials. This Gestalt might be shown by lights at process 3 and the raw material arrow, and a graph showing the dependence of these quantities on the amount of increase at process 2. The computer would not only be able to answer questions posed by the operator but could ask policy-type decisions on operating the factory when two

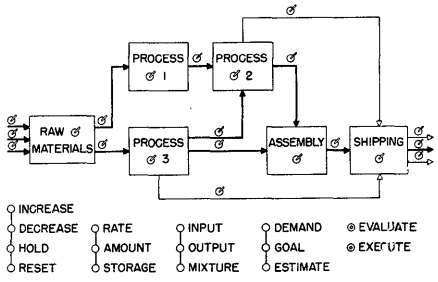


Fig. 5. Gestalt language for automatic factory

equally efficient modes of operation were discovered by the computer. Naturally the computer would keep track of all inventories and would request new supplies of raw materials whenever necessary, with sufficient lead time to maintain operation.

Note that the Gestalt language in both directions has been designed for the convenience of the human operator. In this way the human can always keep his thinking at the problem level and never has to be concerned with how a given task is to be accomplished. Because the language is more a human than a computer language, there is a routine but complicated translation to be done before the computer can actually use the language. This translation is the job of the Gestalt system proper, which is mechanized as a computer program on the same computer which controls the factory. Besides the major job of translating between the Gestalt language and the basic computer characteristics, the Gestalt system also must check statements made by the operator for consistency and completeness. In other words, the Gestalt system supplements the mechanical linkages and layout of the push buttons in ensuring that the rules of the language are obeyed. In this way any ambiguous statements made by the operator are caught before they are acted upon and, in the other direction, the computer cannot speak gibberish.

Principles of Gestalt System Design

With the experience of the example given in the foregoing, the basic principles of designing Gestalt systems can briefly be summarized. The general field to which the system is to be applied may be considered as a topic for conversation between the human and the computer. Usually this topic will be a broad generalization of the problem which initiates the interest in a Gestalt system. In the example, the topic would be control of an automatic factory. At the present state of the art, it is essential that the scope of discussion about a topic be limited to only those aspects which are of immediate interest; in the example, the scope is restricted to the particular factory.

The topic for conversation is broken down into the finest logical divisions necessary to cover the entire scope unambiguously and with a minimum of rules for combination. In the example these divisions are the words, numbers, and locations which were assigned buttons and the various basic units of the computer to human language. All of these various types of basic units will be called items, and the complete set of items forms the vocabulary of the Gestalt language. Thus a Gestalt is expressed in this language by combining items according to syntactical rules. In particular, an item may modify other items. A well-designed language will have a proper balance between items with very specific meanings, to give entry to broad areas of discussion, and items with very general meanings and thus high modifying potentialities, so that a very large and comprehensive body of Gestalts can be expressed with very little equipment.

When a Gestalt language is being designed, the items are always chosen for the convenience of the human, the goal being to have a language which is as natural to use as is possible. This statement may lead to the question why the language should not be English since that is certainly the most natural for the human. This question is clearly answered by the automatic factory example, since obviously it is more natural to select a switch associated with a box or line in a diagram than to try to describe that box by an English phrase. A similar remark applies to the computer-to-human language because a graph or diagram often conveys a complicated idea more readily than a description.

One basic principle on the choice of items cannot be overemphasized, and that is that their meanings must be unique. In other words, a button labelled "increase" must never result in a decrease as a result of modification by another item. Note that this requirement of uniqueness of meaning of individual items does not conflict with previous statements that the meaning of an item is modified by another item, since the modification is an elaboration of meaning, not a change of meaning. For a complicated problem it is often very difficult to find the minimum set of items which completely cover the scope with absolutely invariant meanings, but it is foolhardy to stop short of this goal since the only way to have a workable system is to have the human remember the pathological cases, which defeats the fundamental principle of having the human always think only at the problem level.

Implementing a Gestalt System

The considerations of the previous section have shown that the special Gestalt language is designed entirely on the basis of the problem and the convenience of the human. Once this language has been designed the human is allowed to discuss the problem only in that language so that, in effect, a part of the programming of the problem has been accomplished by programming the human. Note that this is not purely a characteristic of Gestalt languages, since every time any coding scheme or particular computer is applied to a problem, a large number of possible solutions are automatically eliminated by the characteristics of the computer or coding scheme. The aspect which is characteristic of Gestalt languages is that ideally, at least, the programming of the human is entirely beneficial.

The next step, and it is by no means a trivial one, is to program the computer so that it can converse in the Gestalt language as well, i.e., to construct the Gestalt system proper. Because the language was designed for the convenience of the human, it is usually a difficult programming task, but since everything is well defined, it can always be done.

Almost every recognized programming technique can be used to advantage in the realization of Gestalt systems. On the other hand, as might be expected, the peculiar problems which arise often lead to new techniques, or to strong desires for modification of computer logic itself. The cross-fertilization between advanced programming techniques and computer design will be more and more fruitful as these applications expand.

The final important part of the implementation of a Gestalt is the choice of a suitable medium to represent the language. The automatic factory example has already shown the advantages of diagrams and push buttons, but each problem will have its own most appropriate media. The governing criteria on the choice of representations are the rate at which the conversation is to take place and the complexity of the Gestalts when expressed as statements in the Gestalt language.

For low rates of conversation and very complex expressions, the standard input-output media, such as punched tapes or cards and high-speed printers, are probably most appropriate. The spectrum of possibilities also includes intervention switch devices for high rates of conversation. These devices, of which toggle switches and activate buttons are examples, are all characterized by the fact that a unique binary digit accessible to the computer is set to a 0 or a 1 by the setting of the device. Finally, at the ultrahigh conversation rate, there are such mechanisms as steering wheels and joysticks whose positions can be sensed by the computer. For the computer-to-human vocabulary there is a large number of audible and visual indicators, and, of course, the high-speed, very flexible oscilloscope-type output tubes.

In many applications it is desirable to give the human a variety of media for expressing the same Gestalt so that he may choose the most convenient at the time. In all cases, whatever medium is used, the principle of uniqueness should always be observed and the rules of syntax should be positively included by either mechanical or programmed interlocks.

The major steps in the design of Gestalt systems are summarized in the following. In any particular application the considerations of the various sections would undoubtedly be intermingled, but this listing can be used as a check-list summary of the basic points.

Steps in Design of Gestalt System

From the problem:
- Pick a topic for conversation.
- Restrict the scope of discussion.

Design the Gestalt language:
- Choose items which cover the scope.
- Define rules of syntax for combining items.

Design the Gestalt system:
- Determine rate of conversation.
- Choose unique representations for items.
- Establish interlocks by programs or linkages.

Present-Day Examples of Gestalt Systems

There is, of course, a growing number of computer systems which have many of the attributes which have been discussed. In general, however, most of these systems operate at medium to low rates of conversation. Three systems which operate at high rates and are in daily use on the MIT Whirlwind I computer will be briefly described to illustrate more concretely than the applications cited previously, that these techniques are not futuristic in any way, but are sound, practical investments for today.

The Comprehensive System

The MIT comprehensive system (CS), with the topic of "operating a computing facility," has elaborate utility programs, as well as automatic programming aids, under intervention-switch control. In this system Gestalts from the human to the computer may be expressed either in typewritten form, using appropriate mnemonic codes, or by pushing sequences of buttons. These Gestalts automatically call in any one of many programming systems including the CS system for the Whirlwind computer, several simulated computers used in academic courses in programming, a system for programming the Univac Scientific *1103* computer, and a system for programming numerically controlled machine tools. In addition, a large number of post-mortem and error-diagnosis programs are instantly available, as well as a growing number of data-handling and computer-operating routines. Through the use of one of these routines, the director tape program, it is possible to replace the human operator by written Gestalts to operate any number of programs in any sequence with the pushing of only one button.

The Automatic Trouble Locator

The automatic trouble locator program, with the topic of "maintaining a computing facility," is a good example of humans and computers working together. This program is primarily under intervention-switch control and automatically operates the marginal checking equipment of the Whirlwind I computer. The human sets up the general sequence of tests which are to be made by expressing his desires to the computer. The computer then proceeds with the tests, and, since the computer does not have facilities for visual input, it may ask the human to look at the wave forms at critical points, which are displayed on a monitoring scope. The operator need only tell what general type of wave form is being displayed and then the computer proceeds with the analysis. If the computer encounters a marginal piece of equipment, it types out English phrases telling which individual tubes or components require replacement, and then tells how long it took to do the job by a phrase such as "That only took 2 minutes and 33 seconds, are you sure you did it right?", which must be acknowledged by the Gestalt "Yes" before the checking can continue.

The Gestalt system approach will probably find its widest application, at least initially, in the development of similar elaborate systems for greatly improved routine operation and maintainance of other computing facilities. Experience has shown that the results are well worth the effort of devising such systems.

Data Reduction and Experimental Programming

The third Gestalt system in use at MIT is one whose topic is "automatic reduction of armament test data and experimental programming for armament control." This system is the one which has led to the analysis of this paper and is being developed for the Air Force Weapons Guidance Laboratory by the Servomechanisms Laboratory, MIT, using the Whirlwind I computer as a research tool.

This system is so designed that it includes all of the facilities of the MIT comprehensive system. Besides the comprehensive system vocabulary, this system has a large and growing vocabulary of items represented by uniquely assigned push buttons and switches. The rules of syntax which must be remembered by the human are almost entirely conjunctive in nature, the other syntactical rules being inherent in mechanical and programmed interlocks. Any syntactical error, i.e., a meaningless or contradictory combination of switches set by the human, is immediately followed by a unique and explanatory alarm. Conversely, any meaningful statement is properly understood by the computer. The computer-to-human vocabulary primarily uses output oscilloscope displays to express Gestalts, but indicator lights and audible alarms are used where appropriate. The rules of syntax are almost entirely programmed into the computer, i.e., the computer cannot speak gibberish or give misleading information.

Every item in each vocabulary requires a section of programming in the Gestalt system, some absurdly simple and some extremely elaborate. The combined sections are much too large to fit into the magnetic core memory of the computer, so that an essential part of the system is a control program which establishes the proper connections between the various program sections. This facility is so designed that individual sections can be changed easily at any time using the comprehensive system, and still mesh properly with all other sections.

This Gestalt system also has a logging program which provides a written record of the complete conversation between human and computer. This log can also be played back by the Gestalt system, the computer simulating the human actions for rerun purposes. In this way, an interrupted conversation can automatically be resumed.

In operation this system is designed so that the human can interject comments or questions into the computer's operation almost instantaneously and at any time. Some alarm conditions are automatically corrected, with suitable indication, and various types of trouble-shooting can automatically be carried out by the computer on request.

This Gestalt system is in a continual state of flux and improvement. As soon as a new feature is completed, it is usually obsolete in terms of future plans. There are an amazing number of challenges which appear with each new phase, but the results are rewarding. One of the biggest deterrents to progress is the

large amount of work involved in changing the Gestalt system program to correspond to the change of vocabulary required to include some new feature. It is hoped that a solution to this difficulty will be found by writing a program to generate translation programs which will translate from statements in arbitrary Gestalt languages into selections of computer behavior.

The goal of the experimental programming phase of this work is to allow the programmer to alter drastically his planned attack on a very large and complex problem, and try out the new solution within a matter of days, while the new approach is fresh in his mind. All too often a volatile thought pattern disappears in the months of arduous toil required to program a complex problem using ordinary techniques. It is unlikely that present and future problems being considered at the Servomechanisms Laboratory could be solved with limited manpower without the use of these techniques.

Concluding Remarks

It seems appropriate to close this paper by again acknowledging the very real debt which is owed to all of the various schools of computer programming for substantial contributions upon which this paper is based. The emergence and development of these various techniques in the past several years have established firmly the intellectual climate necessary for continued expansion in these directions. There are several groups in the United States which for some time have been developing systems for using computers which have many, if not all, of the attributes of Gestalt programming systems as defined here. The purpose of this paper has been to try to establish the outlines of the abstract structure of this type of system. It is hoped that this analysis will prove useful to all who are interested in connecting humans and computers by clarifying the problems and relationships involved.

In its full generality Gestalt programming is not just a computer technique, but is a problem-solving technique, i.e., a point is reached where it is difficult to tell which is more important, the human, the problem, or the computer. The extension of these techniques and concepts is sure to have a profound influence on the design and operation of future computers, so much so that it seems probable that the term "computer" for describing these mechanisms will become less and less appropriate. The day is fast approaching, if it is not already here, when the arithmetic capabilities of a machine will be its least valuable attributes. If the logical trend toward more and more elaborate systems of this type continues, the primary attribute of a computing machine will be its flexibility in the most general sense. Even if significant advances in the speed of computer elements can be achieved, these gains will be swiftly swallowed up if the logical design of these machines is not advanced to fit the peculiar requirements of these techniques, to obtain the same results with much fewer operations.

At the present state of the art, these future developments can only be sensed in a most intuitive way, although, for example, the growing concept of a microprogrammed computer appears to be a well-founded first step. Continued and rapid advance in these directions both in programming techniques and in computer design, can only be achieved by building on experience gained in studies using present-day facilities. It is hoped that the presentation of these ideas will encourage the participation of other groups in this fascinating line of endeavor.

A Truly Automatic Computing System

MANDALAY GREMS R. E. PORTER

LIKE many comparable groups, members of the computing facility at the Boeing Airplane Company feel that it takes too long to prepare a problem for a digital computing machine. The daily repetition of effort expended in outlining a problem for coding, the tedious task of coding the instructions, and the time consumed in checking-out or "debugging" the instructions all emphasize this fact. In this jet age, it is vital to shorten the time from the definition of a problem to its solution.

A new plan of attack for problem setup is necessary to shorten the elapsed time by shifting more of the monotonous burden of coding to the machine. It is a generally accepted belief that whenever rules for computing can be definitely established, they can be defined as a set of machine instructions. Therefore, the starting point for an automatic computing system is clarifying these rules to fit the requirements of a general problem.

A natural way to communicate a mathematical problem to a computer is by the written equation. This can be accomplished by a system allowing a digital computing machine to accept a problem directly in equation form together with a list of input data. The elapsed time for a problem is therefore shortened because this system eliminates the tedious task of coding the machine instructions. The setup time for each problem is then more dependent on the complete understanding of the mathematics and the logic rather than on the physical characteristics of one special computer.

MANDALAY GREMS and R. E. PORTER are with the Boeing Airplane Company, Seattle, Wash.

The BACAIC System

The Boeing Airplane Company Algebraic Interpretive Computing System, commonly called BACAIC, is a means of communicating directly with a machine. It is a self-contained system for solving a mathematical problem on a digital computer. This problem must be of a type which can be completely described by a set of algebraic and logical expressions. A working record of the entire system, including a file of library subprograms, is kept on magnetic tape. The library is made up of pieces originally constructed in a consistent fashion. This is important in order to establish a general pattern of rules for a system to follow.

The integrated system performs two distinct functions for each problem:

Table I. Definition of Symbols

	Symbol	Use	Explanation
Data reference	A–Z	$A+B$	Refer to all parameters by the letters A through Z, (except K)
	$K1$–$K99$	$K1+B$	Refer to all constants by a K-number
	1–50	$1+B$	Refer to the value (computed or estimated) of an expression by its expression number
Mathematical operations	+	$X+Y$	Addition
	−	$X-Y$	Subtraction
	·	$X \cdot Y$	Multiplication
	/	X/Y	Division
	PWR	X PWR N	$(X)^N$, the quantity X raised to the power N
	SRT	SRT X	$\sqrt{X}$, the square root of the quantity X
	SQR	SQR$[X+Y]$	The quantity following this symbol is squared
Transcendental functions	SIN	SIN A	Sine of angle A. A is in radians
	COS	COS A	Cosine of angle A. A is in radians
	ASN	ASN A	Arcsine A, the angle is in radians
	ACN	ACN A	Arccosine A, the angle is in radians
	EXP	EXP X	$(e)^x$, exponential to the X
	LOG	LOG X	The natural logarithm of X
Logical control	[or $	$[A+B$	Front bracket for a term
	] or ,	$A+B]$	Back bracket for a term
	*	$A-B*Y$	A substitution symbol. Compute the quantity on the left side of the symbol,* and substitute it for the parameter, constant or expression number on the right side of the symbol
	TRN	TRN 8	Transfer to execute expression number 8
	WHN, GRT	WBN A GRT B USE 8.	When the value for A is $\geqq$ the value for B, compute expression number 8 next. Otherwise, compute the following consecutive expression
	USE, LES	WBN A LES B USE 8.	When the value for A is $\leqq$ the value for B, compute expression number 8 next. Otherwise, compute the following consecutive expression
Modification of data	MOD, LIM	MOD $H+B$ LIM R..	Modify the value for H by adding the increment B to H to form a new H. The operation symbols +, −, ·, and / can be used with the increment. Test this new value for H against the limit R. If the limit is exceeded, additional input data for the next case are read by the card reader at the appropriate time. If the limit is not exceeded, the reading of input data is by-passed and the next case is computed using this new H value of input. This procedure is a means for computing families of cases of data when one value of input is repeatedly altered by a preset amount. If more than one "Modify and Limit" expression is tested, the last LIM tested is the effective one
Table look-up and interpolation	ARG, TBL*	ARG X TBL $K2$*Y...	To find $Y=f(x)$. The number in $K2$ is the number of the table to investigate. The tables are consecutively numbered as they are read by the card reader and stored in memory. The selected table is scanned and the corresponding linearly interpolated value for the argument X is computed. This value is substituted for Y
Selection of results	TAB	TAB A $K1$ 3 29 T...	Select the values for the indicated data symbols and store the values on a tape for later printing. Multiple TAB expressions are allowed with a maximum of six symbols per card. When the computing is finished for all cases of data, the stored values are printed. Data for all cases for one TAB card are printed prior to any printing for the next following card. The data are printed as decimal numbers and in the same order as indicated on the card. TAB cards always immediately follow the final equation or control expression
	PCH	PCH B 42 A $K19$....	The PCH expression is similar to the TAB expression, except that the stored data values are punched on cards as decimal numbers rather than printed as columns, of data. PCH cards always immediately follow the final TAB card (when TAB is used) or the final equation or control expression

1. It reads the algebraic expressions and translates them to machine language.

2. It computes results from given data, using the coded machine language instructions.

The choice of one of these functions is selected manually by the machine operator through controls on the console panel. This choice causes certain portions of the system to be operative and other portions to be by-passed.

The algebraic equations and logical controls which describe a problem are punched directly on cards to be read by the machine. These expressions are then translated by the computer to machine language instructions. The resulting machine instructions are automatically punched in binary form on cards. The time required for translating and machine-coding a problem usually averages 2 to 5 minutes; e.g., 10 expressions require about 2 minutes and 50 expressions require about 5 minutes.

The machine-coded instruction cards, accompanied by a set of given values for input data, are fed to the computer whenever computing is to take place. The results of the computing for one set of input data is printed (or stored for later printing) before another set of given values for input data is read. Computing of results for one problem continues for all sets of input data which are ready in the machine. The computing time per problem is dependent on the number of sets of given data and on the complexity of the computing pattern. The computing time usually ranges from a few seconds to 1 minute for each data case.

The algebraic equations and controls for a problem are written in terms of familiar symbols for reference to data values, mathematical operations, transcendental functions, logical control, table look-up and interpolation, systematic modification of data, and selection of results for printing or punching. The mathematical symbols such as +, −, ·, /, SIN, COS, LOG, and EXP are familiar to most people and an endeavor is made to assign mnemonic symbols to other operations.

Definition of Symbols

The mnemonic symbols for writing the expressions are grouped as shown in Table I.

Input Data Form

The input data for a problem are prepared in the same manner as for desk computing; i.e., a list of the values in terms of a reference symbol, a coefficient with a decimal point, and a possible power of 10 for this coefficient. When

the power 10 is zero, the zero is omitted, e.g.

$A=3.75$

$W=0.00375\times10^{3}$

$M=375\times10^{-2}$

Preparing the data in this manner provides the opportunity for entering items of data in either a floating or stated system of notation, and eliminates the necessity for changing each item of data to a preset notation system. Once the power of 10 is established for each value of data in the system, it is automatically adjusted for all operations performed on that value.

PROBLEM SETUP

Example 1. To illustrate the ease of preparing a problem for the BACAIC system, evaluate

$$Y=e^{-x^2}\sin CX \qquad (1)$$

for values of X from -0.99 to 1.00 in intervals of 0.01, and tabulate the corresponding values for X and Y.

This problem is written as three expressions for BACAIC:

1. MOD $X+K1$ LIM $K2$
   Modify a value for X
2. EXP $[K3-X\cdot X]\cdot$SIN $[C\cdot X]*Y$
   Compute Y
3. TAB X Y Tabulate X and Y

Each expression is punched on a card and the three cards are read by the machine. These expressions are machine-coded by the BACAIC system and a resulting set of instructions is punched on cards by the system. These instruction cards are fed to the machine with the following values of input data:

$C=5.0$ $K1=0.1$ $K3=0.0$
$X=-1.0$ $K2=0.99$

The expressions for this problem are executed consecutively in the foregoing order unless otherwise indicated. The machine accepts the data and repeatedly computes values for X and Y until the limiting value for X is exceeded. When the computing is finished, the 199 sets of values for the X and Y results are tabulated.

Example 2. Compute both roots for multiple values of C with constant values for A and B.

$$AX^2+BX+C=0 \qquad (2)$$

To solve for both roots, rewrite the equations as follows:

$$\frac{-B+\sqrt{B^2-4AC}}{2A}=X$$

$$\frac{-B-\sqrt{B^2-4AC}}{2A}=Y$$

To illustrate a comparison and selection, assume the following conditions:

When the discriminant (B^2-4AC) is positive, use its true value.

When the discriminant (B^2-4AC) is negative, use a value of zero.

This problem is written as five expressions for BACAIC:

1. $B\cdot B-K4\cdot A\cdot C$
   Evaluate discriminant
2. WHN 1 GRT $K5$ USE 4
   Compare values and select
3. $K5*1$
   Substitute zero
4. $[K1\cdot B+$SRT 1$]/[K2\cdot A]*X$
   Compute X
5. $[K1\cdot B-$SRT 1$]/[K2\cdot A]*Y$
   Compute Y

The five expressions are machine-coded by the system. The punched instruction cards are fed to the machine with the given input data.
The input data are:

$B=6.0$ $K4=4.0$ $K1=-1.0$
$A=1.0$ $K5=0$ $K2=2.0$

(Case 1) $C=5.0$
(Case 2) $C=4.5$
(Case 3) $C=9.0$
(Case 4) $C=18.0$

The machine accepts the data for case 1, computes a result for each expression, and prints these five results for case 1. The machine then reads the second value for C, computes each result and prints the five results for case 2. This procedure continues for the four given values of C.

The two illustrated examples demonstrate the general plan for writing the expressions where each expression is punched on an individual card. The examples also demonstrate the difference between selective printing of results and the printing of all results for each case. There is a noticeable difference in the printing time for the two methods. This factor should be considered at setup time, as the needs of the problem or the needs of the programmer determine the type of printing.

Criteria for Coding

It may be asked how a machine can consistently interpret and translate the algebraic equations so quickly and so accurately. This idea is plausible when it is accepted that a set of rules for the machine in its own language is sufficient for translating. These rules must be definite and exact for all situations. The BACAIC system now appears straightforward and relatively simple. The present system differs considerably from the original plan, as the former includes more details and special features.

In order to establish an over-all plan for interpreting the equations directly from the cards, many decisions for writing the equations and controls had to be formulated. Some of these decisions were mandatory as they depend on the particular computer in use. The BACAIC system was written especially for the International Business Machines Corporation (IBM) Model *701*. However, much of the planning and organizing of the system can easily be transferred to another digital computer. The reader of the IBM *701* reads a maximum of 72 uppercase letters, numerals, and symbols. This dictates that one level of punching or printing is recognized by the machine, thereby eliminating the possibility of punching or printing subscripts or superscripts in the familiar way. This limitation is easily overcome by an appropriate symbol to signify the operation or meaning to the machine.

Some of the early decisions depended entirely on the anticipated types of problems to be studied and the characteristics of their data. The question of floating point arithmetic versus stated point arithmetic arose with stronger arguments in favor of the floating point system. In the floating point system, the elimination of the problem of scaling values of input data is very satisfying. The use of this arithmetical system for BACAIC is proving to be an attractive feature for inexperienced personnel. The rules for machine computing in the floating point system were firmly established at an earlier time when the library subprograms were written. These library subprograms for floating point arithmetic were incorporated in the system and the rules governing them were accepted unchanged. Fortunately, a standard pattern for the input-output to these library subprograms had been adhered to and was readily adaptable to a system.

One of the next questions to be solved concerned the values for constants and data. If the actual values of data are included in the expressions, the digits of the numbers occupy too many of the 72 available card columns, so a scheme for referring to all data by symbols was developed. The values for the corresponding symbols are entered at computing time. This scheme has the added feature of making it very convenient to alter values without rewriting the expres-

sions. Originally, the 25 alphabetic letters A through Z (except K) and the 99 K's ($K1$-$K99$) seemed sufficient for data reference, but this is proving to be inadequate for some problems. The numbers 1–50 are data reference symbols for the values of the corresponding expression results. These values are estimated values for the expressions or computed values for the expressions. A reference of this type provides a simple means for using a computed result for one expression as an input value for another expression. In the second expression of example 2

WHN 1 GRT $K5$ USE 4

the 1 refers to the result of the first expression; i.e., the value of the discriminant (B^2-4AC). This reference is especially convenient in a problem when an estimated value of a result is needed to start the computing, but after the first computation, the symbol refers to the most recently computed result.

Many mathematical problems require a choice of operations at various levels of the solution. The designers of computing machines recognize the need to select and transfer, as they invariably include machine codes for "transfer on plus," "transfer on minus," or "transfer on zero." In an automatic system, this need for a conditional transfer is even more urgent. It is the only means for describing a problem as a complete picture when part of the picture is dependent on a previous computation in the same problem. When this select and transfer feature is included in a system, problems dealing with iteration, integration, and progressive summation are easily manipulated. Without this feature, a system is very limited in its application.

The foregoing information helps to outline a general plan for an automatic computing system. After these notions are settled and accepted, the rules for writing the expressions are considered with respect to the capabilities of the machine. The limitation of any computing machine is that it executes *exactly* all instructions which it receives and it remembers *only* the information it is told to remember.

The mnemonic symbols for certain operations are readily recognized and accepted as three adjacent letters, such as SIN, TAB, LOG. This starts a pattern for mnemonic symbols for all operations, and recognition for the exact symbols is easier when the first two letters of a symbol are not the same as the first two letters of another symbol.

The use of parentheses for the grouping of terms within terms is very essential when writing equations. It is natural to use parentheses or brackets in equations for the purpose of grouping terms to be used as one operation; e.g., SIN $(A+B+C)$. It is necessary to close all bracketed groups; i.e., the brackets must travel in pairs. Therefore, a separate symbol is needed for the front bracket and a separate symbol is needed for the back bracket. This ability must be available in an automatic system, and from experience must be increased in an automatic system to include equivocal situations. In the second expression of Example 1, the sine term is coded as "SIN $[C\cdot X]$." This removes the doubtful meaning for

the sine of C to be multiplied by X

or

the sine of the product, C multiplied by X

Without the ability to group operations, a system is extremely limited in its usefulness. It is a toy and not a tool for computing.

The arithmetic operation for division is another stumbling block to a smooth system. The division concept presents a few difficulties, as up to this time all operations are assumed to be in the numerator. Obviously, an exception to the rule is necessary. The revised rule for writing expressions states that all operations are in the numerator except those following a division symbol. Then, only the symbol or bracketed term immediately following the division symbol is in the denominator. This practice is successful and is relatively simple to contend with for all situations. This rule is demonstrated in the fourth and fifth expressions for example 2, where the numerator is divided by a product.

Interpreting an Expression

The ability of a machine program to analyze a given algebraic expression and determine the unambiguous sequence of computations intended by the originator of the expression is subject both to the natural rules of algebra and to the restrictions imposed by the machine programmer. Certain restrictions result in the consistency so vital to machine programs yet impose no hardship upon the person writing an expression; e.g., the substitution of 3-letter mnemonic codes such as SIN, COS, and SQR for sine, cosine, and square. This makes machine decoding much simpler without detracting from the natural appearance of the expression. Any restrictions on the use of arithmetic symbols or the grouping of terms are more difficult to justify. The number of permissible symbols and the length of any one algebraic expression is usually influenced by the data input and internal storage capabilities of the machine used. It is in the best interests of the machine program's users to concede everything to the naturalness of writing an expression. Only the limit of the programmer's ingenuity dictates the restrictions which need apply.

The principal problem in interpreting an expression is that of defining the rules which the machine must follow to produce an unambiguous operating sequence.

A few of the contingencies encountered are illustrated in the following examples:

Example 3.

$$a \sin b+\frac{c}{d}-x+y \qquad (3)$$

Example 4,

$$ay+\left\{q+(n^2-r)(a+b)+\frac{c-d}{\frac{d}{x}+q}\right\}\text{SIN } v \qquad (4)$$

The first contingency is the "understood multiplication" illustrated in the terms "$a \sin b$" or "ay." This type of operation was eliminated from BACAIC expressions by making it illegal (the simplest way out of any coding dilemma). The rule that all arithmetic operations must be indicated by the appropriate symbol simplifies the initial translation step. It is possible to have the machine itself supply the understood operation symbols at the cost of extra programming.

The next contingency is that of having a choice as to which operation to perform first. This choice can neither be eliminated by a rule nor left to the discretion of the machine. A human computer has a choice of either of two operations when starting to compute the result of example 3. He may divide c by d or compute the sine of b. Five such choices are possible in example 4. These choices cannot be left to a machine. Instead, a way must be determined of defining an order of operations having no chance of duplication or ambiguity during machine interpretation.

The normal rule of algebra that all multiplication and division must be performed before terms are combined is only a partial answer to the problem. In example 3, the function operation "sine b" must be performed before it can be multiplied by "a" and this multiplication must be performed before the entire term ($a \sin b$) can be added to the quotient of c divided by d. Possible ambiguities in operation sequences may be avoided by combining the normal rules of algebra

Table II. Right Operand Condition Table

Operation Requiring Operand	Adjacent Right Item	Item Following Right Item	Right Operand Corresponding to the Stated Conditions
A function such as SIN, COS, LOG	An item symbol such as *A*, *B*, *K*15	. or /, + or −,], special expression termination symbols	The item indicated by *A*, *B*, *K*15
	A group of terms indicated by [ ]	. or /, + or −,], special expression termination symbols	The result of the last operation performed within the brackets
Multiply or divide (. or /)	An item symbol such as *A*, *B*, *K*15	. or /, + or −,], special expression termination symbols	The item indicated by *A*, *B*, *K*15
	A group of terms indicated by [ ]	. or /, + or −,], special expression termination symbols	The result of the last operation performed within the brackets
	A function such as SIN, COS, LOG	Irrelevant	The result of the function operation
Add or subtract (+ or −)	An item symbol such as *A*, *B*, *K*15	. or /	The result of the last multiplication, division or function operation performed before the next add or subtract operation is encountered
		+ or −,], special expression termination symbols	The item indicated by *A*, *B*, *K*15
	A group of terms indicated by [ ]	. or /	The result of the last multiplication, division or function operation performed before the next add or subtract operation is encountered
		+ or −,], special expression termination symbols	The result of the last operation performed within the brackets
	A function such as SIN, COS, LOG	Irrelevant	The result of the function operation

Note 1. All other sequences of symbols are violations of expression writing rules.

Note 2. All operation symbols in the following right items must be in the same group as that of the operation requiring a right operand.

with the rule that operations are performed in the order they are encountered in the expression from left to right. The resulting combination is specifically stated by the following rules.

RULES FOR DETERMINING OPERATION SEQUENCE

1. Scan the expression from left to right assigning ascending operation sequence numbers to every function operation; e.g., sin *b* in example 3.
2. Rescan the expression from left to right continuing the assignment of ascending operation sequence numbers to every multiplication or division symbol; e.g., $a \cdot \sin b$ and c/d in example 3.
3. Again rescan the expression from left to right continuing the assignment of ascending operation sequence numbers to every addition or subtraction symbol; e.g., $a \sin b + c/d$ and that result $-x$, etc., in example 3.

The three foregoing rules are easily programmed and permit the machine to choose automatically an unambiguous sequence of operations for an algebraic expression.

Example 5.

②① ④ ③ ⑤ ⑥ ← Operation Sequence Numbers (5)

$A \cdot \mathrm{SIN}\ B + C/D - X + Y$

This example shows the original expression of example 3 in BACAIC form with the sequence of operations indicated above the operation symbols.

Example 6

⑫ ⑭ ⑨ ① ③ ⑦ ④ ⑩ ⑤ ⑧ ② ⑥ ⑬⑪ (6)

$A \cdot Y + [Q + [\underbrace{N\mathrm{PWRZ} - R}_{2}] \cdot [\underbrace{A + B}_{2}] + [\underbrace{C - D}_{2}] / [\underbrace{D/X + Q}_{2}]] \cdot \mathrm{SIN}\ V$

Group 1: the expression within the outer brackets; Group 0: the whole expression.

Examples 3 and 5 ignore the problem which arises when operations are grouped by parenthesis or "bracket" symbols (see example 4). This grouping of operations is very necessary to the writer of an algebraic expression. It is essentially a mathematical shorthand notation which permits him to specify the general order in which he desires computations performed. Since the human computer handles these groups of terms by working "from the inside out," a machine must do the same. This is accomplished by the assignment of a "group number" to every significant symbol in the expression in accordance with the following rules.

RULES FOR GROUP NUMBER ASSIGNMENT

1. Scan all significant items of the expression from left to right assigning the same group number to each item until a left (front) bracket symbol is encountered. (Note: If no left bracket is present in an expression, all items will have the same group number.)
2. When a left (front) bracket symbol is encountered, increase the current group number by one and assign this increased number to that bracket symbol and to all successive items until another bracket symbol is encountered.
3. When a right (back) bracket symbol is encountered, assign the current group number to that bracket symbol and then decrease the current group number by one, assigning this decreased count to all successive items until either another bracket symbol or the end of the expression is reached. (Note: All groups must be completely enclosed; e.g., there must be an equal number of left and right brackets.)

The machine system is programmed to work "from the inside out" by first assigning group numbers to all expression items in accordance with the preceding rules, second determining the maximum group number and applying the "rules for determining operation sequence" to that group, third decreasing that number by one and reapplying the operation sequencing rules to this next group, etc., until all groups have received their operation sequence numbers.

Example 6 shows the original expression of example 4 in BACAIC form with the group numbers indicated below each group and the corresponding sequence numbers indicated above each operation symbol.

Observe that the group number distinguishes a level of grouping rather than a particular group; e.g., there are several group 2's in example 6. An examination of example 6 also reveals that although the operations are performed in a seemingly heterogeneous manner, the operand needed by each operation is calculated in time to permit an uninterrupted sequence of operations.

After the operation sequence is defined, the final problem is defining how the machine is to find the proper operand or operands for each operation. The most

common type of operation requires two operands, that is, a quantity both to the left and to the right of the operation symbol. Other operations such as the sine function require only one operand which is normally written to the right of the operation symbol. Therefore, as the machine examines each operation symbol, it must have a means of distinguishing those operations requiring a single operand from those requiring both a left and a right operand.

Examples 5 and 6 indicate that either operand may be the result of a previous calculation rather than an item symbolized in the original expression. They also indicate results of previous operations are not necessarily used in the next operation. These facts require that the result of every operation be stored separately within the machine for use at any later time while computing that expression. In other words, the computing sequence is such that the result of an operation cannot automatically become one of the operands for the following operation. The use of brackets in an expression requires that either operand may be the result of a group of operations as well as a single quantity or previous result. A summary of the conditions governing the selection of a right operand for the various operations and the corresponding expression context is given in Table II. A similar summary for the selection of a left operand is given in Table III.

The previously assigned group numbers and operation sequence numbers are used in the selection of operands to meet the conditions summarized in Tables II and III. Application of the following rules by the machine enables it to select the proper right operand for each indicated operation.

Table III. Left Operand Condition Table

Left Operand Corresponding to the Stated Conditions	Item Preceding Left Item	Adjacent Left Item	Operation Requiring Operand
The result of the function operation	A function such as SIN, COS, LOG	An item symbol such as *A*, *B*, *K*15	A function such as PWR, GRT, LES
The item indicated by *A*, *B*, *K*15	or /, + or −, [, start of expression		
The result of the function operation	A function such as SIN, COS, LOG	A group of items indicated by [ ]	
The result of the last operation performed within the brackets	or /, + or −, [, start of expression		
The result of the last multiplication or function operation performed after the first preceding add, subtract, [symbol or the start of the expression is encountered	A function such as SIN, COS, LOG, or . or /	An item symbol such as *A*, *B*, *K*15	Multiply or divide (. or /)
The item indicated by *A*, *B*, *K*15	+ or −, [, start of expression		
The result of the last multiplication division or function operation performed after the first preceding add, subtract, [symbol or the start of the expression is encountered	A function such as SIN, COS, LOG, or . or /	A group of items indicated by [ ]	
The result of the last operation performed within the brackets	+ or −, [, start of expression		
The result of the last preceding addition or subtraction operation after the [symbol or the start of the expression is encountered	A function such as SIN, COS, LOG, or . or / or + or −	An item symbol such as *A*, *B*, *K*15	Add or subtract (+ or −)
The item indicated by *A*, *B*, *K*15	[, start of expression		
The result of the last preceding addition or subtraction operation after the [symbol or the start of the expression is encountered	A function such as SIN, COS, LOG, or . or / or + or −	A group of items indicated by [ ]	
The result of the last operation performed within brackets	[, start of expression		

Note 1. All other sequences of symbols are violations of expression writing rules

Note 2 All operation symbols in the preceding left items must be in the same group as that of the operation requiring a left operand.

Rules for Determining Right Operand

1. Beginning at the operation requiring a right operand, scan all expression items to its right having group numbers equal to or greater than that of the operation itself. Record the maximum operation sequence number encountered before:

(*a*) An operation sequence number, with the same group number, is encountered which is greater than that of the original operation sequence number, or

(*b*) A right (back) bracket symbol with a group number equal to that of the original operation's group number is encountered, or

(*c*) An item having a group number less than that of the operation itself is encountered, or

(*d*) The end of the expression is reached.

2. When no operation sequence number is recorded prior to meeting conditions 1(*a*), 1(*b*), 1(*c*), or 1(*d*), the item immediately to the right of the original operation is its proper right operand.

3. When an operation sequence number is recorded prior to meeting condition 1(*a*), 1(*b*), 1(*c*), or 1(*d*), the result corresponding to the maximum operation sequence number recorded is the proper right operand.

Similarly, the machine selects left operands for each indicated operation which requires one by applying the following rules.

Rules for Determining Left Operand

1. Beginning at the operation requiring a left operand, scan all expression items to its left having group numbers equal to or greater than that of the operation itself. Record the maximum operation sequence number encountered before:

(*a*) An operation sequence number, with the same group number, is encountered which is greater than that of the original operation sequence number, or

(*b*) A left (front) bracket symbol with a group number equal to that of the original operation's group number is encountered, or

(*c*) An item having a group number less than that of the operation itself is encountered, or

(*d*) The start of the expression is reached.

2. When no operation sequence number is recorded prior to meeting condition 1(*a*), 1(*b*), 1(*c*), or 1(*d*), the item immediately to the left of the original operation is its proper left operand.

3. When an operation sequence number is recorded prior to meeting condition 1(*a*), 1(*b*), 1(*c*), or 1(*d*), the result corresponding to the maximum operation sequence number recorded is the proper left operand.

The machine system determines the operation sequence and the corresponding operands, and records its findings in a sequence table. A 3-address operation sequence table is a familiar way of recording such information. Table IV illustrates the BACAIC Operation Sequence Table for the expression given in example 3.

Expression Interpretation Rules

The steps involved in the machine interpretation of an algebraic expression are summarized in the following rules.

1. Scan the expression's characters classifying them into operation, operand, expression result, grouping, computation control, and expression termination symbols.

2. While performing the classification, eliminate all extraneous spaces and mnemonic characters and fill in appropriate items for all "understood" symbols.

3. Assign group counts to all expression items.

Table IV. Operation Sequence Table

Seq. No.	Right Operand	Operation	Left Operand	Result
1	None	Sine	B	1
2	A	Multiply	1	2
3	C	Divide	D	3
4	2	Add	3	4
5	4	Subtract	X	5
6	5	Add	Y	6

Note 1. The contents of this table is given symbolically rather than in machine codes and storage location addresses.

Note 2. The result for the entire expression corresponds to that for the maximum sequence number in the entire expression; e.g., 6

4. Assign operation sequence numbers in accordance with these groupings and the sequence determination rules.

5. Determine the operands corresponding to each operation.

6. Record the operating data in a form from which actual machine instruction sequences may be assembled.

Translating a Sequence Table to Machine Instructions

Once a sequence table is prepared, a few more specific decisions are necessary before it can be translated to machine instructions. Probably the most important is that concerning the storage of the values of input data. A need for a convenient method of reference to either an address of a data value in the sequence table or an actual location in storage is evident. This need is handled by reserving an area in storage for values of data. This concept is similar to the need for boxes at a post office. In this reserved area, one box or location is set aside for each data reference symbol, i.e., (*A*-*Z*), (*K*1–*K*99) and (1–50).

Each box originally contains zero, and remains at zero until a value is placed in it. A value can be entered in each box either as an item of input data or as a computational result. The current value in any box is the only value available at any time.

Another decision is whether the entire contents of the master tape or only the coded machine instruction deck is available at computing time. When the entire tape is available, the machine is able to print comments and other information appropriate to any situation.

A minor detail (one which is probably assumed to be a fact) is the packing of high-speed storage in a unique fashion for each different problem. This utilizes the storage more advantageously and requires less reading of records from the master tape.

The procedure involved in translating each sequence table to machine instructions is as follows. The sequence tables are scanned for the mathematical operation codes. Each different code is recorded once and a complete list made of all the operation codes referred to in the expressions for one problem. This list of codes is incorporated in an index of information for library subprograms. A location is assigned to each required subprogram and this assigned location is stored in the index. When the location assignments are finished, this index is punched on cards. These cards are used during computing time by a relocation program to pack the specified library subprograms in working storage. The information in this index is also used to insert the addresses for the machine instructions which are dependent on the actual location of each of the library subprograms.

This index is now discarded and full attention is directed toward each sequence table and the preparation of the corresponding machine instructions for that expression. The actual locations of the input data and of the result data for each operation code are taken from the sequence table. These are stored as addresses for certain machine instructions of the library subprogram for that operation code. These machine instructions containing the references to data and to subprograms are packed adjacent to similar machine instructions for the previous operation code in the same table. This procedure of storing data locations as addresses of instructions and then packing the instructions continues for each operation code of a sequence table. The complete set of machine instructions for the one expression is punched on cards in binary form to be used at computing time. The entire process is repeated for each subsequent table. When the punching for the last expression takes place, the automatic coding for the problem is finished.

Computing Procedure

The master tape is used during computing time as it retains the bulk of the system instructions. The coded binary cards containing the instructions for a problem are used repeatedly with varied values of input data. The instruction cards are fed to the machine together with heading cards for identifying the results. At the start of computing time, the working storage is filled with those portions of the system needed to prepare the machine for computing. The library subprograms are packed adjacent to one another in working storage. An area originally set to zero is reserved for values of input data. The input data is read as decimal numbers and stored in the area reserved for the corresponding symbols. Only one value is saved for any one symbol at a time. A new value merely replaces the old value for the same symbol. After these preliminary preparations are finished, a continuous cycle of machine action takes place. The computing always starts with the first expression and ends with the last expression. Normally, the expressions are executed consecutively but a TRN or USE symbol alters this normal routine. Loops for iteration or integration can be included between the first and last expression by means of the logical control symbols. All computing is performed in floating point arithmetic. The results are available for each case after executing the last expression for that case. Whenever an intermediate result of computing is needed, it must be written as a separate expression. This cycle of reading input data, computing results, and printing or storing result data is broken when the problem is finished or when some interruption of machine action occurs.

Computing Controls

The following computing controls are necessary to increase the over-all usefulness and flexibility of a computing system.

1. The choice of an expression to execute due to the result of a comparison.

2. The systematic modification of input data when it varies by regular intervals.

3. The selective printing of input data and computed results.

4. The selective punching of input data and computed results.

5. The printing of comments with pertinent information which describes errors or points out violations in usage of the library subprograms, the input data, the expressions, or other machine instructions.

6. The interruption of computing due to an emergency and the later restoration of data for continued computing from the point of interruption.

7. A combination of the last two features; i.e., the printing of comments, the interruption of the computing, and the later restoration of the data for continued computing.

An explanation and description of these controls clarifies the benefits which they add to an otherwise incomplete system.

Comparison and Selection

It is assumed that the expressions are written in the correct sequence for computing, and that this same physical sequence is maintained throughout the problem. In other words, a reference to expression number 1 is always the first expression of the written set; and a reference to expression number 2 is always the second expression of the written set.

Often, a comparison of two values or a selection of a specific expression is desirable at some definite point in the computing which upsets this normal sequence. Suppose that the following condition is necessary, "when the value for A is greater than the value for B, use equation number 6 to compute the value for C; otherwise, use the next equation to compute the value for C." This selection is written as follows:

WHN A GRT B USE 6

A comparison of this type is one of the logical controls which can be handled by the system in the same manner as a mathematical equation. Therefore, insert this logical control in its proper sequence with the set of expressions. Each of the values to be compared can be computed prior to the comparison in the same expression. For example,

WHN $[A+R-T]$ GRT [SIN $[X+Y]\cdot W]$ USE 10

Modification of Data

Some mathematical problems are of the type similar to example 1, $Y=e^{-x^2}$ sin CX, where Y is evaluated for all values of X from -0.99 to $+1.00$ in intervals of 0.01. Similar situations frequently arise and it seems appropriate for the machine to prepare its next new value of input whenever possible. The symbolic expression for this data preparation is

MOD $X+K1$ LIM $K2$

The value for the increment $K1$ is added to the value for X and the sum replaces X. This new value for X is compared with the limiting value for $K2$. If X is less than $K2$, the input data reading routine is by-passed at the beginning of the next case. If X is greater than $K2$, cards for input data are read by the machine at the beginning of the next case. The arithmetic operation attached to the increment can be $+$, $-$, $\cdot$, or $/$. The data values can be positive or negative since the signs are tested to insure modification in the indicated direction.

Selective Printing

The BACAIC system did not initially include selective printing. This shortcoming was immediately realized when unnecessary printing of all results for a problem took place. This complete printing of intermediate results was confusing and difficult to explain to inexperienced personnel. It was also a needless waste of valuable machine time. The symbol chosen for selective printing is TAB. A reference to any data symbol is allowed with a maximum of six references per TAB. Each TAB symbol is written as a separate expression. The TAB cards follow the equation and control cards. When the computing is finished for one case of input data, the indicated values are selected and stored on a magnetic tape. For the following expression

TAB A $K5$ 7 R 19 42

the values for A, $K5$, result 7, R, result 19 and result 42 are selected and stored. The system then by-passes all printing routines at that time and continues to compute the next case of data. At the end of computing for a problem, the selected and stored data are listed. Multiple TAB expressions are permitted, but the printing for the first TAB is completed before any printing for the second TAB takes place.

Selective Punching

Selective punching satisfies the need for a form of output which can be used as direct input to another machine program. In the BACAIC system, the data values are selected and stored the same as for selective printing. However, unlike TAB, this is an additional function of the system and does not replace another function. When the computing is finished for a problem, the selected and stored data values are punched on cards as decimal numbers. A maximum of six data reference symbols is allowed per PCH code. The PCH expression cards follow all other expression cards.

Diagnostic Assistance

There are two legitimate types of machine stops when BACAIC is controlling the machine:

1. A STOP when the operation of the machine can be continued.
2. A STOP when the operation of the machine cannot be continued.

Each of these stops can be caused by keypunching errors, computing difficulties or machine malfunction. In order to distinguish which error caused the machine to stop, a "machine trail" is printed. This "machine trail" includes a pertinent comment to state the reason for stopping and to indicate corrective measures. It also includes the exact location in the memory unit of this unexecuted instruction, the number of the expression it was examining or computing, and the next instruction to execute after the corrective measures are accomplished. This last-mentioned transfer instruction is impor-

```
SAMPLE   03 EXPRESSIONS  SAMPLE PROBLEM FOR BACAIC

1   MOD  X  +K1  LIM  K2
2   EXP  $K3 - X.X , . SIN $C . X.  * Y
3   TAB  X  Y

         X VALUE  Y VALUE


THE ORIGINAL INPUT DATA FOR CASE NUMBER  1

+  C    5.0          X   -1.1          K1 0.2           K2  1.0
1

 THE SELECTED RESULTS ARE LISTED AS  FOLLOWS.

     X VALUE          Y VALUE
    .90000000-  8-   43486215   8-
     70000000-  8-   21489906   8-
     50000000-  8-   46609057-  8-
     30000000-  8-   91164176-  8-
    100000000-  9-   47465517-  8-
    100000000   9-   47465517   8-
     30000000   8-   91164176   8-
     50000000   8-   46609057   8-
     70000000   8-   21489906-  8-
     90000000   8-   43486215-  8-
    110000000   8-   21039020-  8-

THE COMPUTING IS COMPLETED FOR ALL CASES OF DATA ENTERED IN THE MACHINE.
```

Fig. 1. A solution of a problem by the BACAIC system

tant if the computing can be continued from that point. All of these are aids to locating an error or discrepancy and to provide a written record for future reference or study of the expressions and the program. An example of a "machine trail" is:

```
54      TWO ADJACENT OPERATION CODES IN AN EXPRESSION
        DECIMAL NUMBERS                    OCTAL LOCATIONS
CONTROL PROG.    EXP. NUMBER        STOPPED AT      TRANSFER TO
     13               4                3752            1654
```

Interruption of Computing

Occasionally, the computation for multiple cases of data must be interrupted before the computing is finished for all the cases. The computing can be carried on at a later time if the values for the parameters, the constants, and the results in the memory can be restored to the identical values at the time of the interruption. When an interruption is necessary, a SENSE switch is turned ON while the machine is computing. The results for the current case of data are computed and printed or stored. In those instances where selected values are stored for later printing or punching, this printing or punching of the accumulated data also takes place. The afore-mentioned pertinent data are punched in binary cards. These same data are printed to be used as a reference for the purpose of cross-checking the data and results. When the time arrives to continue the computation, these binary cards are fed to the machine prior to the decimal input data for the unfinished cases. A console SENSE switch is turned ON which controls the reading of binary data cards and storing them in the memory unit prior to computing the first expression. The contents of the memory unit are hereby restored to the identical values at the time of the interruption. The computing is then carried on as if no break had occurred.

Interruption after Diagnostic Assistance

Sometimes a violation of a computing rule for a library subprogram is caused by an incorrect value of data. When this value is an incorrect input value, it probably is sufficient to note the error in the data, to print the current results at that computing point, and to start computing for the next case of data. Zero values are stored for that case of result values in problems which include a TAB or PCH expression. This prevents the possibility of printing or punching erroneous results unwittingly.

When the incorrect value encountered is one which was developed in the computing, an attempt to determine the cause of the error is recommended. The interruption control is activated by turning on a console SENSE switch. This causes all values in the reserved area to be printed as decimal numbers and to be punched as binary data. All data previously stored for the TAB or PCH are handled the same as for the end of computing. It is assumed that the cause of the error can be detected after examining these printer decimal data. The binary values for these data are fed to the machine at a later time so that computing continues from the beginning of the next case. An example of an error which causes the machine to stop is shown in the following:

```
11  FOO7  THE ARGUMENT IS TOO LARGE.  ADD MORE VALUES TO THE TABLE.
        DECIMAL NUMBERS                    OCTAL LOCATIONS
CONTROL PROG.    EXP. NUMBER        STOPPED AT      TRANSFER TO
     31               5                3222            5136
13 CASE RESULTS WRONG.  PUSH START FOR NEXT CASE, OR SENSE 1 FOR INTERRUPT
        DECIMAL NUMBERS                    OCTAL LOCATIONS
CONTROL PROG.    EXP. NUMBER        STOPPED AT      TRANSFER TO
     31               5                6362            5136
```

Computing Features

The addition of various extra features contributes to the operating smoothness of any computing system. The ability directly to include empirical or other functions resulting from test result correlation minimizes mathematical curve fitting and hence elapsed setup time. Complete machine identification of results saves clerical time and reduces errors resulting from misinterpretation of unidentified data. The originator of a problem needs assurance that the machine interprets his problem correctly. This is accomplished by the system comparing machine results with the results anticipated by him. Other features can be added as the need arises to expand a system.

Table Look-Up and Interpolation

A table look-up routine is needed to satisfy all those conditions of data which cannot be easily expressed by equations. In many instances, this set of table data is prepared more quickly than one equation or multiple equations for the data. Tables can be altered from one computing time to the next when table data are part of the input data rather than part of the expressions.

In cases where empirical data are used for parts of the computing, it is often advisable to resort to a table look-up and interpolation routine. At the present time, only linear interpolation is available in the BACAIC system.

The values for the table look-up routine enter the machine in a manner similar to entering values of regular input data, except that the pairs of table data are stored consecutively. The first item of each pair of values must be in consecutive ascending or descending order. The maximum size for each table is arbitrarily limited to 400 half words or 100 pairs of values in the BACAIC system. The tables are stored as consecutive records on a magnetic drum whose limit of 4096 half words is also the limit of half words for all tables. The system prepares an index for locating each table whenever it is needed.

Suppose that the first table on the drum is an X, Y, table such as:

X	Y
2.0	20.
3.0	30.
5.0	50.

Find the corresponding value for Y when $X=3.72$ and $K3=1.0$. The expression for BACAIC is as follows:

ARG X TBL $K3*Y$

The value stored in $K3$ is the number of the table used to find Y. This expression is interpreted to read "look up the argument X in the first table and substitute this value for Y." Incidentally, the argument can be computed prior to the lookup routine in the same expression. For example,

ARG[$X+Y-$SRT[SQR SIN $A+$SQR COS A]]TBL $K1*Y$

Identification of Result Values

Up to this time, little attention has been paid to identifying the quantities to be computed for each problem. During the

```
JOB 7959 38 EXPRESSIONS  DATA REDUCTION - FR 205  FOR WIND TUNNEL

          1   $K13 - K17,/K18
          2   K13/$K13 - K17,
          3   $K13 + K17,/$K18 . $K13 - K17,,
          4   $K17 + 1 . SQR E, PWR 2
          5   F/4
          6   K3 . $M - N,
          7   WHN 6 GRT K16 USE 9
          8   K16 - 6 * 6
          9   $J + K2 . $M - N,, . $K1 + 6,
         10   9 + $N - 5, . K9 + $M - 5, . K8 + $P - 5, . K7
         11   SRT$$$H/1, PWR$K17/2, - K17,/1,
         12   K17 + 1 . SQR 11
         13   12 PWR 3
         14   SRT$G + K14,
         15   SRT$K13 . K11/K12,
         16   K4 . K5 . 15 . H . 11/$14 . 13,
         17   K17 + 1 . SQR E
         18   17/12
         19   18 PWR 3
         20   K4 . K5/K6 . H/F . 11/E . 19
         21   $K18/$K13 + K17,, PWR 3
         22   K10 - K7
         23   21 . 15 . 22 . L/16/14
         24   K22/$23.K21,
         25   SQR24.SQR$SQR24, + K19.SQR$SQR24, + K20.SQR24 - K21.23.24 + K22 * Q
         26   WHN 25 GRT K16 USE 28
         27   K16 - Q  * Q
         28   WHN Q  LES K26 USE 31
         29   24 - 25/$K23.SQR$SQR24,.24 + K24.24.SQR24 + K25.24 - K21.23, * 24
         30   TRN 25
         31   K17 + 1 . SQR 24
         32   31 PWR 2
         33   L/32
         34   K17 + K13 . SQR 24
         35   K13 . SQR E . 20 . K6
         36   K13/K18 . 5 . SQR E
01266    37    10  + 33.34.22 - 5.$22 + 35,
         38   37/$36.K6,

         1        2        3        4        PO       6
         TEST     8        9        FG       MN       12
         13       14       15       16       17       18
         19       AO/AL    21       A2 - ACB 23       24
         25       26       27       28       29       30
         31       32       P2E      34       35       QO
         D INLET  CD INLET

         THE ORIGINAL INPUT DATA FOR CASE NUMBER  1

           K1 .04           K2 23.39         K3 0.0           K4 0.983
           K5 6.1575        K6 4.891         K7 0.3849        K8 1.3902
           K9  2.9732       K10  6.1575      K11  32.174     K12 53.345
           K13   1.4        K14  459.0       K15   0.2        K16   0.0
           K17   1.0        K18   2.0        K19   15.0       K20   75.0
           K21  216.0       K22  125.0       K23    6.0       K24   60.0
           K25  150.0       K26   1.0    -4  A    114.0       B     20.0
           C  1.0           D  7025.0        E  1.99          F  14.76
           G  110.          H  8.05          I  6.00          J  -288.0
         + L  9.99          M  6.80          N  3.30          P  7.41

           CASE NUMBER    1.       THE COMPUTED RESULTS

         1                2                3                4                PO               6
01266     20000000  8-     35000000  7-     30000000  7-     77037089  7-     19159602  7-

         TEST             8                9                FG               MN               12
01266          9                             82454000-  7-     47740751  7-     66182211  8-  108760170  8-

01266    13               14               15               16               17               18
01266     12864995  7-     23853721  6-     91890304  8-     96560392  8-     17920200  7-     16476804  7-

         19               AO/AL            21               A2 - ACB         23               24
01266     44732052  7-    100409966  8-     57870380  8-     57726000  7-    133139777  8-     50446148  8-

         25               26               27               28               29               30
01266                              28                                31        50446148  8-           25

         31               32               P2E              34               35               QO
01266    105089628  8-     11897598  7-     83966528  7-     13562739  7-     27227557  6-     53111758  7-

         D INLET          CD INLET
01266     72863860  7-     28049417  8-

         THE COMPUTING IS COMPLETED FOR ALL CASES OF DATA ENTERED IN THE MACHINE.
```

Fig. 2 Actual data reduction problem and results

computing of a problem, the only printed information other than the given expressions and the input data is the computed results. When each result is printed, it is identified by a corresponding result column heading.

The quantities to be computed are not the same for all problems, therefore, the result column headings are not the same for all problems. These headings must be introduced individually for each problem. They are separate from the expressions which they identify so that they do not interfere with the mathematical symbols and abbreviations and are not interpreted. These heading cards are fed to the machine in back of the expression cards. The headings are stored in the memory unit during the computing and each is available for printing whenever the corresponding value or column of values is printed.

Checking of Sample Data

In many instances, it is desirable to check the accuracy of the machine-coded instructions before computing multiple cases of data. A satisfactory check of the accuracy is a comparison of a set of anticipated (hand-calculated) results with a set of machine computed results. This comparison of results checks the accuracy of the coded instructions for:

1. The interpretation of the mathematical symbols.
2. The machine-coding of the operations.
3. Comprehensiveness of the library subprograms.

A set of sample data includes a value for each data reference symbol in the expressions for the problem. It also includes a value for the anticipated result to each expression. These anticipated results are fed to the machine in the same manner as input data.

The machine computed results for the algebraic equations are self-explanatory. The machine computed results for the logical controls of the BACAIC system are indicated as follows:

Expression	The Machine Computed Result
WHN *A* GRT *B* USE 6	....6 or the number of the next consecutive expression
WHN *A* LES *B* USE 8	.....8 or the number of the next consecutive expression
TRN 12	 12
ARG *M* TBL *K*1 * *Y*	*Y* the interpolated value from the table to be substituted for *Y*
MOD *H*+*X* LIM *Y*.	.[*H*+*X*] the incremented value for *H*

The computing is started in the normal manner. The computed result for expression number 1 is compared with its anticipated result. If these two values are the same (slide-rule accuracy), expression number 2 is computed and its two results are compared, then number 3, etc. When the computing is finished for all the expressions the value for the machine computed result for each expression is printed.

If the two results (anticipated and computed) for a comparison are not the same for an expression, both result values are immediately printed for that expression. Accompanying these values are appropriate comments and sufficient data to analyze the difference. These data include the values for the parameters, the values for the constants, and the values for the computed results stored in the memory unit at that time. Usually this information is sufficient to isolate or to indicate where the discrepancy occurred.

In order to locate more than one error during each check-out period, the computing is continued without interruption. The value for the anticipated result for the questionable expression is substituted for the computed result for that expression, and the computing is continued for the next expression. This test and substitution is made so that an error in one expression at the beginning of a check cannot be reflected throughout the computation. If this substitution were not made, it is possible the results of the succeeding expressions might not compare with the anticipated hand-calculated results.

If the difference (as explained) is the result of an incorrectly keypunched expression card, a new program must be coded by the machine. After the expression cards are corrected as indicated, the coding phase is repeated. If there are only small differences between the anticipated results and the computed results, the computer accepts the coding for the expressions.

When the reason for the difference between the expected and computed results is not obvious after examining the data print-out for the sample computation, the coding of the expression or expressions must be repeated. The reason for the error may be discovered if the 3-address Sequence Table is printed. This Sequence Table indicates the order of machine execution for each operation in the expression. The table is printed as follows:

Sequence Number	Address for the Left-hand Quantity	Operation Code	Address for the Right-hand Quantity	Address for the Result
1				
2				
3				

The exact sequence of the machine operations can be examined and the reason for the error determined. A possible misplaced front or back bracket symbol can alter the correct sequence.

There will be occasions when even this sequence table is not sufficient. In these cases the coding is repeated and the table of character codes for the expression is printed. This, however, is of very little use without an explanation of the character codes and is used only in extreme cases.

When the results compare favorably for each expression, the instruction cards for the program are accepted as correct.

Economic Aspects

The BACAIC system is a completely automatic system for which the only required information for a new problem is:

1. The algebraic expressions with the result column headings.
2. The decimal input data for each case.

The operation of the machine is entirely dependent on the instructions contained in the system. A few of these are controlled by the ON or OFF position of some external switches on the console panel. However, the machine operator (not the originator of the problem) is responsible for the position of these switches.

This system encourages the programmer to direct more attention toward the mathematical preparation of the expressions. This is a field in which he probably is better trained and more experienced than in the field of machine coding. More time can be spent concentrating on the phases of the problem which require human judgment and decision and less time on the tedious task of coding. Also, a minor detail which is quickly apparent after the first attempt with BACAIC, is the noticeable lack of careless errors. When a comment for a careless error is machine printed on the result sheet, a greater effort is made by the programmer to eliminate such errors before using the machine The standardizing of the procedure for the machine operator helps to avoid confusion which often results from vague or poorly written individual operating procedures.

A vital point of interest to most computer users concerns the amount of elapsed time from the outline of a problem to the time when the first production results are ready. Service to outside departments based on overnight or 24-hour planning is probably the best that any installation can reasonably strive for. When using the BACAIC system for solution of a problem, this goal is within reason. An estimate of the time required to solve a special problem can be based on the approximate time required to perform the various steps as follows:

Number of Expressions	10 Expr.	30 Expr.	50 Expr.
1. Write the expressions and prepare the data	1 hr.	2 hrs	4 hrs.
2. Key punch the expressions and data	1/2 hr.	1 hr.	1 1/2 hrs.
3. Machine code the expressions	2 min.	3 min.	5 min.
4. Machine compute one set of results	10 sec.	40 sec.	1 min.

The time allotments for the various steps are generous and can be decreased by improving the pieces of the system and also by increasing the experience of the programmers for writing the expressions. Often, a different approach to the same problem results in fewer expressions and shorter machine time.

It required approximately 18 man-months to outline the plan, develop the ideas, establish the rules, write the instructions, and "debug" these instructions for the BACAIC system. These 18 man-months were spread through an elapsed time of 1 year. At the end of the first 10 months, most of the system was in working order. The usual elusive errors and unreasonable reasoning had to be located and deleted. The majority of the subprograms were already available, and only had to be altered to include error comments rather than error stops.

The writing of the instructions for this system is proving to be a never-ending process and will only cease when more advanced ideas are not forthcoming. It could easily be in a continuous state of change if all the ideas for improvement are accepted and included.

The Adaptability of a Natural System

A 1-to 2-hour informal discussion describing the general outline and operation of BACAIC is our method of introducing it to various engineering groups. In each case, this discussion is followed by the distribution of a written digest of the BACAIC system. This digest quickly reviews the preparatory steps for writing both the expressions and data and out-

lines the operating steps for the machine procedure. Many of the rules for writing the expressions are merely a review of the fundamentals taught to a beginning student of algebra. These rules must be strictly adhered to so that both the programmer and the machine interpret each situation in the same manner. The talk accompanied by the digest proves to be sufficient information for a beginner. Naturally, there are individual questions at a later time. These questions usually concern the inclusion of additional features to improve the system. All of these new ideas are welcome and whenever possible they are included immediately.

It is worth mentioning that most improvements and additions are readily included. The BACAIC system was originally planned with that desired flexibility in mind. It is fairly easy to include an additional mnemonic symbol but, if the symbol is to refer to a library subprogram for its operation, the library subprogram must also be available.

The specific information for each problem; i.e., the expressions and the input data, can be relayed by means of the telephone from another department to the computing facility. This service is possible because of the standard procedure of the BACAIC system after the expressions are formulated. The availability of telephone service tends to decrease the elapsed time for solving a problem, and also tends to increase the correctness of the written expressions. Individual pride and reputation play an important part in reducing careless errors. Usually, the "debugging" of the expressions is possible in a few minutes prior to any machine operation. This fact is of tremendous advantage economically since without an automatic system considerable machine time is spent on check-out for each problem.

Conclusions

A natural computation language eliminates the machine coding details currently responsible for the expenditure of large amounts of man and machine time. This language can include complete machine operating directions as well as the mathematical problem statements. The time now spent in digital computer problem preparation can be reduced by as much as 90 per cent through the use of machine self-coding systems. A fundamental computation language makes evolutionary machine changes possible without extensive personnel retraining. The man-hours required to construct an autocoding system are no more than those formerly spent on a subprogram library. The basic interpretive principles for an algebraic computing system are applicable to most present-day stored program digital computers. These principles can be incorporated in the hardware of future machines.

Lincoln Laboratory Utility Program System

H. D. BENNINGTON C. H. GAUDETTE

THIS paper discusses a utility program system to assist the coding, check-out, maintenance, and documentation of large-scale control programs. A typical program contains 50,000 instructions, 1,000,000 bits of data storage, and is prepared by a staff of 20 to 40 programmers, many relatively inexperienced. The utility system requires 25,000 registers.

H. D. BENNINGTON and C. H. GAUDETTE are with the Lincoln Laboratory of the Massachusetts Institute of Technology, Lexington, Mass.

An Automatic Supervisor for the IBM 702

BRUSE MONCREIFF

VERY little experience has been accumulated in the operation of a large commercial data-processing center. However, reflection on the subject has led to the conclusion that, in the large-scale operation of such a system, there will be a different emphasis from the one usually present in the operation of a large-scale computing installation. The general administrative problem in both cases is, of course, to keep both staff and equipment operating efficiently. In the latter case, however, the emphasis is on new problem preparation, while in the case of the business application the emphasis must be on the efficient day-after-day operation of the same routines. The automatic supervisory routine described here is an attempt to solve those operating and programming problems peculiar to this "routine-dominated" situation.

The excuse for solving these problems with a machine program, rather than by instructions to the operator, is twofold:

1. The human operator cannot compete in speed with the machine in making routine decisions and in controlling the processing operations.

BRUSE MONCREIFF is with The Rand Corporation, Santa Monica, Calif.

2. The human operator is more likely to make mistakes in carrying out routine instructions.

The purpose of a supervisory routine is, therefore, to keep the machine running efficiently in spite of the slowness and fallibility of the human operator.

The various aspects of the proposed supervisory routine will be approached by looking at these problems of both operator and programmer which the routine helps solve. A summary description of the way the routine works will follow.

Operating Problems

On a machine such as the *702*, the major operating problem is the efficient handling of tape reels. Every few minutes, during a long run, either an input tape will have to be selected and mounted, or an output tape will have to be labeled and stored away. The malfunctioning of tape units will probably result in the assignment of units to the various input-output functions being changed from day to day. Trying to keep the machine running under these conditions, without a good system of operation including adequate checks, could be a difficult job, with a high probability of error. The component parts of this tape-handling problem will be examined one at a time.

1. Going from one job to another, aside from the internal problem of replacing the old program with a new one, involves a new input-output arrangement of the tape units. If this arrangement were the same every time a particular job is run, a table of assignments for each job could be preprinted and used day after day. Because of the malfunctioning problem mentioned, this is not feasible. Also, whenever there are sufficient tape units operable, the procedure of alternating tape units should be used. While one tape of a certain type is running, the next tape of that same type can be mounted on another tape unit. The problem of getting the machine to deal first with one tape unit and then the other is a programming problem to be dealt with in the following. However, the day-to-day changes in the input-output setup for each job, to take advantage of all the operable tape units, is a problem for the operating staff. The solution proposed here is to get the supervisory routine to calculate the optimum input-output arrangement before each job, having been given a list of the presently operable tape units. The routine should assign alternate tape units as far as possible, starting with those tape types which have the greatest number of tapes per run. Also, assignments of alternate pairs should be made to physically adjacent tape units, as far as this is possible. The machine will print out a table of assignments which the operator will use as a guide in mounting tapes.

2. Once the tapes have been mounted, the supervisory routine will check to see that this has been done properly. Each input tape will be checked for proper type, proper cycle number (e.g., that it was the output of yesterday's run), and proper sequence number within type and cycle. The first record of each tape in the system will be a label containing the information required for this check. The supervisory routine will consult a table of cycle increments, one for each input type by job, and will keep track of the sequence of tapes mounted during each job. The routine will also check each tape mounted for the purpose of receiving output to guarantee that the information on this tape is no longer needed. It will do this by comparing the cycle number of the tape with the current-run cycle number minus an increment (file protection period) which may be different for each tape type.

To summarize the foregoing paragraph: Each tape type will have associated with it both an input increment and an output increment; the former will guarantee the accuracy of input-tape mounting, and the latter will protect information against accidental erasure until it is no longer needed.

3. An identification label will be recorded by the supervisory routine on all output tapes as well as printed out on the typewriter for mounting on the reel can. This will provide positive visual identification for each can of tape in the system.

There is another operating problem, not connected with tape handling, which a supervisory routine can help solve. This is the problem of setting alteration switches before the running of a job. Two points are worth noting in this connection. First, the number of physical alteration switches provided on the *702* will occasionally be inadequate. So provision must be made for logical or programmed alteration switches. Secondly, there are what might be called the active and the passive types of switches. In the active type, the operator has a present need to manipulate a switch in order to alter the machine activity. The passive type of switch is one which he has been instructed to set in a certain way at a certain point in the processing cycle. While the operator is not likely to make mistakes with active switch settings, the probability of error increases in the case of the passive switches. It is recommended that the physical switches be reserved for the active use, while a system of logical, or programmed switches, to be set by a punched card, be used for the passive, job-setup type. The supervisory routine will establish whether or not an alteration switch card is needed for each particular program, and if so, will wait until a card of the proper type is provided. This card will be stored in a standard drum section. This system relieves the operator of the responsibility of remembering to make the switch settings before each job. A file of prepunched cards can be maintained which will cover the majority of setups. Most, if not all, occasions of the use of program parameters can be handled by this control card.

As a further aid to the operator, especially for the purpose of retracing the steps of a process that went wrong, the supervisory routine will automatically keep a log. Entries will include job numbers, tape setups, switch settings, information entered manually or by card, and error stops. One solution to the problem of the multiple use of the typewriter is the creation of a carbon copy. The tape labels can be cut out of the original, with the carbon left intact as a log.

Programming Problems

The problems which a supervisory routine can help solve, in the programming area, are those of co-ordination among programmers and those of co-operation between programmer and operator. In the first place, the employment of a supervisory routine can prevent the duplication of effort which arises where each program must contain the same general control and housekeeping routines. Each programmer must, in each of his routines, provide for tape-unit assignment, tape-unit alternation, tape-mounting instructions and checking, and a labeling system. He must also provide for loading his routines and for alteration switches of the program variety. All of these matters can be taken care of once and for all by means of incorporation in a supervisory routine.

In addition to eliminating most of the duplication of effort among programmers, the supervisory routine can be an unobjectionable means of enforcing the necessary standard practices. This is especially important in the matter of tape identification and tape-mounting checks. Unnecessary confusion would arise if every programmer were left to design his own system of output-tape identification. Then there would have to be designed a

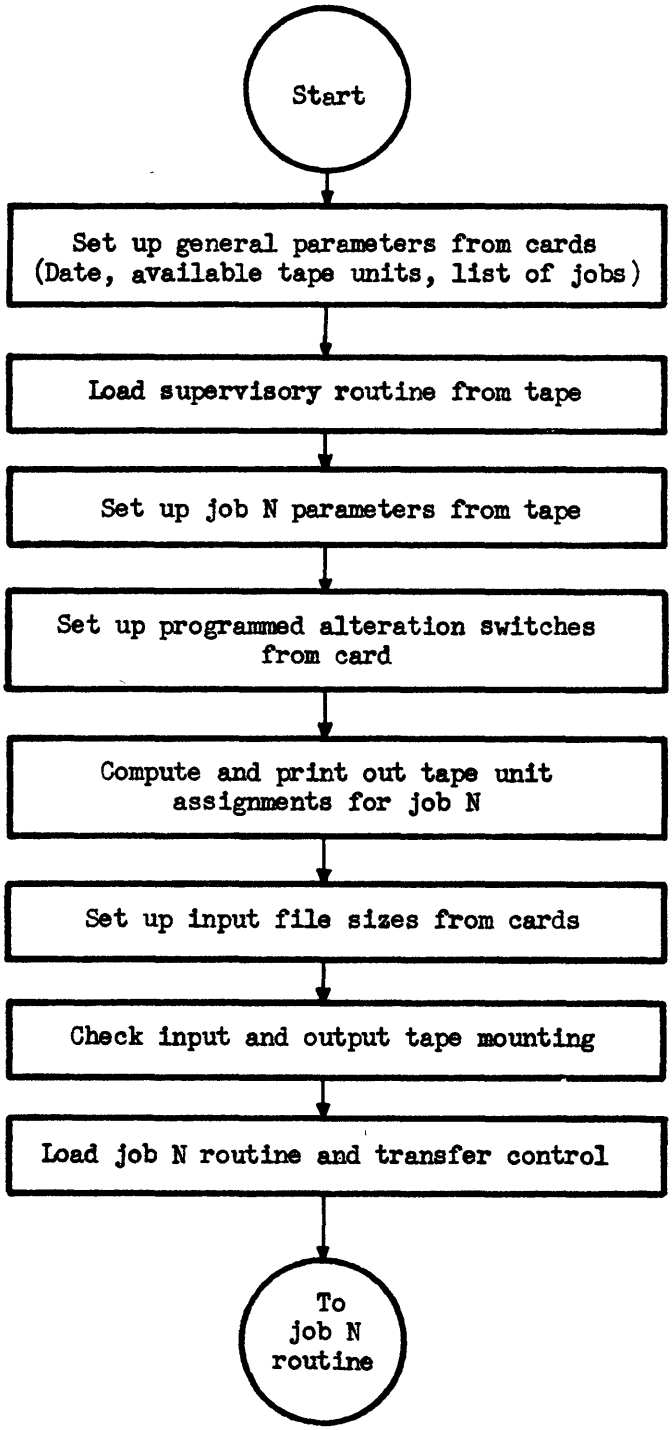


Fig. 1. Supervisory routine: initial operations

different input check for each type of input for every job.

The elimination of confusion is also the primary problem in the relations between the programmer and the operator. In systems where the operator has to run many different programs each day, he has many points of contact, and must understand and deal with many ways of doing things. In a sense, the supervisory routine provides a single point of contact between the operator and the programs. Once the characteristics of this supervisory routine are learned by the operator, his job is made easier. Changes in the specific job routines, once "debugged," are of little concern to him. Only changes in the supervisory routine directly affect the operator, and it is expected that most of these changes will grow directly out of operational needs. The previously discussed method of program alteration, the uniform method of tape identification, and the universal checking of tape mounting, will all but guarantee that the plans of the programming staff will be correctly carried out by the operating staff.

Operation of the Supervisory Routine

The supervisory routine will now be approached from the viewpoint of the machine operator (Fig. 1). The processing will be started each day by entering, via punched card, the day's date, cycle number, list of available tape units, a (standard) list of jobs with a starting point, and select and read orders for bringing in the first section of the supervisory routine from tape. This information, all but the select and read orders and the list of jobs, will be typed out as the day's first log entry. The list of jobs will control the processing sequence, and may be altered manually at any time. The programs for the various jobs will be automatically called up from tape by reference to this table. When the initial setup has been completed, the supervisory routine will be loaded from tape, and will search for the first job. The first section of each specific program contains the necessary program parameters, principally tape-unit requirements, to be used by the supervisory routine in accomplishing the setup. The supervisor will check first to see if there is need for a new setting of the programmed alteration switches. If so, the card will be read and the settings made. The complete tape-unit setup for this particular run will be computed and printed out. The machine will stop, waiting for the operator to signify, by pushing the start button, that the tapes have been mounted.

As soon as the tapes are mounted, each input tape and then each output tape will be checked as described. Any incorrect tape mounting will result in a printout describing the trouble, followed by an error stop. When the error is corrected, a recheck by the supervisor will be made of the whole setup, starting back with the first input tape. As the output tapes are checked, the identification record is written on the tape. This record is also printed out as a label when the tape is filled.

Upon satisfactory completion of the tape-mounting check, the supervisor will load the job program. After the specific program is loaded and arranged properly in memory and drum, a transfer of control to its starting point will take place. A transfer of control back to the supervisory routine, a small fragment of which is left stored in memory, will take place under any of the following conditions (Fig. 2):

1. An output tape has been filled. The output label will be printed, including any "closing" information left by the specific program in a standard location. If the

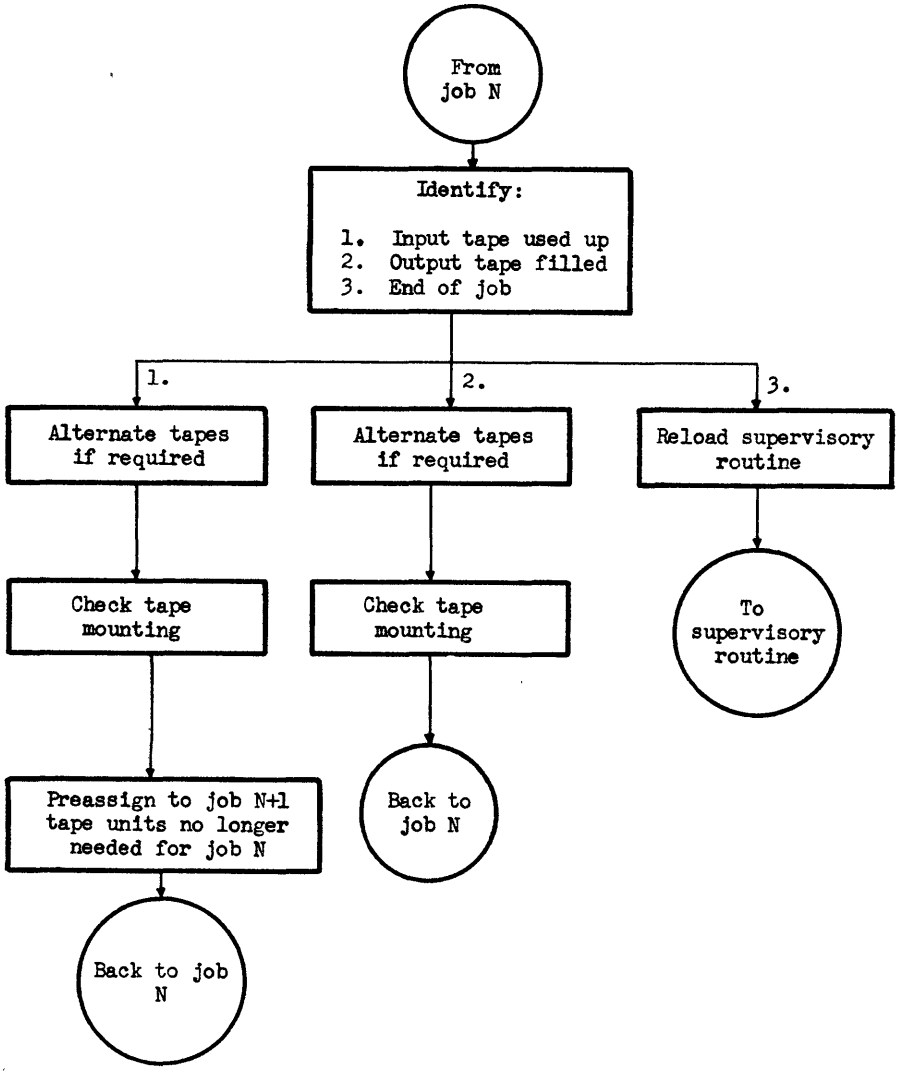


Fig. 2. Supervisory routine: between-tape operations

tape is one of those being alternated, controls will be set for the alternate tape unit. The routine will proceed immediately to the routine check of the tape mounted on the alternate unit. If this particular tape is not being alternated, the machine will pause until the new tape is mounted, before proceeding with the check, and the writing of the identification record. Control will then be transferred back to the specific routine.

2. An input tape has been used up. The same procedure for handling alternating tape units is followed, where necessary, before the new input tape is checked. The cycle number and serial number are checked and the serial number is stepped. Control is then returned to the specific program.

3. End of job. The supervisory routine is called up, and the setup for the next job is started. This consists of picking out the next job number and going through a setup procedure (Fig. 3) similar to that described.

Following the last legitimate job number of a cycle, there will be a pseudo job number which will call up a cycle closing routine. This routine will step the cycle number and then test an alteration switch to see if the operator desires a shutdown or a new processing cycle.

Preassignment of Tape Units

Certain items appearing in the block diagrams have not been explained. These have to do with the preassignment of tape units. This subject will be dealt with separately here, since some background discussion is required.

The delays caused by tape mounting have been reduced as much as possible by having the supervisory routine alternate tape units whenever they are available. But this "hides" only the tape mounting time which is internal to a job. If the operator must wait until a job is completed before starting to mount tapes for the next job, this whole setup time will constitute a delay. To ameliorate this situation, the supervisory routine has been designed to make preassignments whenever possible.

For output functions there seems to be no simple way of determining when tape units are no longer needed. On the other hand, the number of tapes in each input file is known in advance. This information, if available to the supervisory routine, can be used to trigger a preassignment of the no longer needed tape unit to the next job. In order to accomplish this preassignment, the routine needs to have available the job setup information for the next job, as well as the current one. The block diagram of the initial operations (Fig. 1) should be amended to include the setting up of job N + 1 parameters. The block diagram of the between-job operations (Fig. 3) should also be amended at the beginning of this routine to show that job $N + 1$ becomes the current job, job N, at this point. Then the first block again reads correctly, since it is the next job, job $N + 1$, parameters that are actually being set up at this time. These complications of the diagrams are not self-explanatory, hence they were not included.

The normal end-of-file system is not adequate, since in the case of alternating tape units, the first preassignment takes place when the next to the last tape has been used up. The method adopted for satisfying this need was to provide a card for each output file, punched automatically by the supervisory routine at the completion of a job. The card, in addition to identifying information, carries the number of tapes produced during this run. During a later cycle when the output file becomes input, the card is used to set up the controls needed for the preassignment function. Similar cards can be created manually for input files created outside the system.

When a job is completed, the remainder of the input-output setup for the next job is computed with the preassignments being conserved. The routine makes the best choice of physical location for alternate tape units, just as it did for the first job setup. But where preassignment is involved, the choice of the first of a pair of units has already been made. These second choices are made before any other assignments are made.

Programming the Supervisory Routine

Two general points of programming philosophy were adopted early in the detailed job of realizing the supervisory routine in a code for the *702*. The first was that each requirement for informa-

tion storage would be satisfied by the lowest possible level in the *702* hierarchy of storage devices. The second was that time would be used wherever possible as a trade-off for storage. The result of this approach was: (*a*) a complication of the logic and (*b*) a requirement for only 700 characters of permanent high-speed storage.

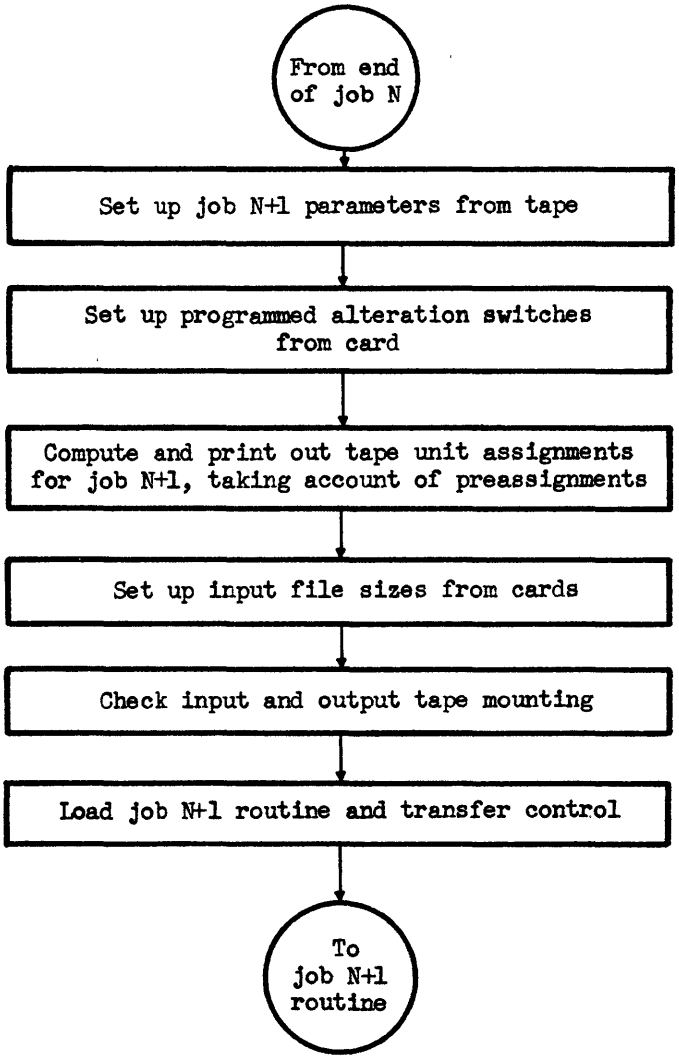
The supervisory routine is designed for storage on tape. The part which operates between jobs can fill the memory if need be (it does not, of course). This part is wiped out by the loading of the specific job program. The part of the routine which operates between tapes is stored on 15 200-character drum sections. For execution, these sections are loaded one at a time as needed into a common area in high-speed storage. Five additional drum sections are required for storing the tables of information used in the problem. When referred to, these are brought up into a second common area.

The part of the supervisory routine executed between jobs is reloaded from tape when needed. To save tape searching time, this program can be recorded again and again, between specific job programs. If this is done, and if the specific jobs are done in the order in which they appear on the program tape, there will be no tape searching required. However, full searching facilities have been built into the routine, for purposes of greater flexibility.

A single sheet (22-inch by 30-inch) flow chart was constructed initially, and then subjected to many changes. In a problem of this type, where logical complications are mostly those of relationships, a single sheet chart can be a great help. If carefully constructed, changes can become only a minor annoyance.

The routine was coded in IBM *702* symbolic, taking approximately 1,200 lines. This includes all entries required for the complete assembly of the problem for the *702*.

Fig. 3. Supervisory routine: between-job operations



Conclusion

The activities of a computer operator can be classified as (1) those things which he does when the machine is working well, and (2) those things which he does when the machine is working poorly. The supervisory routine described is intended to be an aid to and substitute for the activities of only the former class. If considered as a research problem, this approach may be realistic, but it does not constitute a complete practical solution which could be adopted as it stands. In fact, the aspect dealt with is most certainly the easiest part of the total problem of keeping the equipment occupied with useful work.

The purpose of the research was to gain a feeling for the complexity of a problem which as far as is known, has not heretofore been extensively investigated. The conclusion is that the construction of such a routine is not as complex as it originally seemed to be. The operating practicality is as yet not established.

Magnetic Recording Head Design

A. S. HOAGLAND
ASSOCIATE MEMBER AIEE

AN analysis of the process of magnetic recording of digital data is presented from which qualitative magnetic head design concepts are developed and their usefulness demonstrated, both in the evaluation of magnetic head structures and in design for high-density storage.

This work was begun in connection with the development of the International Business Machines Corporation (IBM) *305* magnetic disk storage file, where rather stringent requirements necessitated a program for improving the performance of the existing magnetic recording unit. The external constraints imposed on this work essentially limited the design parameters to those concerned with the magnetic head, excluding the gap dimension. The investigation of this subject led to the development of an analytical formulation of the process of magnetic recording which is directly applicable to data recording. The insight derived from the theory and its relative simplicity in application will be demonstrated through its successful employment in guiding head design development in connection with the IBM *305* magnetic disk project.

The paper will first briefly present a general discussion of magnetic data recording, stressing the form of the readback wave form and its relation to bit density. Then the basic theoretical development will be presented and the significance of these expressions to magnetic head design will be stressed. The remainder of the paper will present experimental results and their correlation with the theory, followed by a description of the selected magnetic head for the magnetic disk file and its performance.

Data Recording

Magnetic recording of data involves the storage and processing of binary information, utilizing two states which are capable of being differentiated. For convenience consider NRZ (nonreturn-to-zero) recording in which the storage surface is continuously saturated during writing, in one direction for a "1" and in the opposite sense for a "0." A particular input is then composed of a succession of alternating step changes in writing current. Reading involves a derivative-type action as indicated schematically in the diagram below, illustrating the overall transfer process from input to output. The surface co-ordinate is x, and $\bar{x}$ represents the variable indicating the position of the specified surface magnetization relative to the magnetic head.

$$i(t) \text{---} M(x) \text{---} \phi_h(\bar{x}) \text{---} vNd\phi_h/d\bar{x} = e(\bar{x}) = e(vt)$$

where:

$M(x)$ = distribution of magnetization set up in the storage surface
$\phi_h(\bar{x})$ = reading coil flux as a function of surface position
e = open circuit readback voltage
N = number of turns on the reading coil
v = surface velocity; t = time

Hence $e(vt)$ is a pulse-like signal for a step change in $i(t)$[1] and the surface velocity appears as a scaling factor in both time and amplitude for the output signal. An output voltage signal is then associated with each change in the direction of saturation or reversal in writing current.

Consider the output signals for arbitrary input patterns consisting of sequences of alternating step changes in writing current. On readback the magnetic field existing, that due to the magnetization of the surface, is extremely weak and hence the magnetic head will behave very nearly as a linear element. Further, writing definition or the width of a saturation transition region is much less than the corresponding reading resolution, due to the nonlinear surface saturation characteristic. Thus, generally, reading will limit density before the point is reached at which adjacent changes in surface state modify one another. These considerations imply that the principle of superposition may be applied to the overall input-output transfer process, using the characteristic step function output response. The width and wave form of this characteristic response are thus extremely important and such responses will be shown as functions of x, or distance, as only in this manner are they meaningful. As indicated, the velocity enters only as a scaling factor between distance and time and only spatial relations are significant in this recording process. It is evident that bit density considerations will follow directly.

Consider a symmetrical voltage pulse response with the base width of λ. The maximum bit density permissible where no mutual interference is to be allowed, i.e., each saturation change is to be resolved independently, is then equal to $1/\lambda$. On the other hand if the requirement is only that no pattern modulation occurs, implying no signal peak variation, then the bit density has a limit of $2/\lambda$.

Now a typical response signal is shown in Fig. 1. The base-line character (or wave form) is quite important, as well as pulse widths at various levels above the base-line, in density considerations.

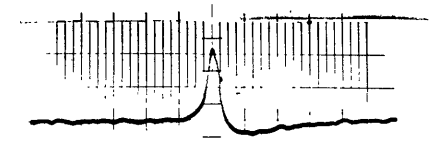


Fig. 1. Typical response signal

Scale: 3.5 mils per division

The "overshoots" may be broader than the main pulse. Thus no single criterion can be used in comparing responses.

A general objective for high-density design is to attempt to obtain a spike-like characteristic output signal, i.e., to eliminate base-line signal and maximize the rate of approach of the signal to the base-line. Considering undesired signal as noise, the overshoots tend to limit the signal-to-noise ratio at lower densities and first cause overlapping of signals as the bit density is increased. However, even with no base-line signal, as the bit density is increased and correspondingly a spacing reduction between adjacent sampling strobes errors will arise with an isolated pulse signal due to the gating of strobes on either side of the strobe which is actually associated with this output indication. This limitation is accentuated by pulse responses with gradual base-line approaches and can severely limit performance.

Theory

In addition to the principle of superposition the principle of reciprocity will be used, these forming the basic tools in this analysis. The applicability of the principle of reciprocity to readback follows as an imediate extension from the

A. S. HOAGLAND is with the University of California, Berkeley, Calif.

justification for using the principle of superposition. As employed here the concept of reciprocity states that the magnetic field, H, set up by energizing the reading coil with a small current (assuring low field levels appropriate to readback and likewise the same linear behavior) gives a measure at each space point of the degree of coupling existing between a weak magnetic field source at that point and the reading coil. The space points of concern are those that constitute the magnetic sources of readback, i.e., the storage surface, primarily the track passing under the transducer but also including nearby adjacent tracks.

A reasonable simplification for the objectives desired, and in view of the nature of the problem, is to neglect variations with surface layer depth. This is equivalent to considering the surface as a relatively thin film which is often the case. The depth factor will be treated later. Then functions defined along the surface will depend only on one variable. The centerline of the magnetic head gap (ring structures) will henceforth be chosen as the fixed reference point, *i.e.*, $\bar{x} = 0$.
Then

$$\phi_h(\bar{x}) = \phi_{hx}(\bar{x}) + \phi_{hy}(\bar{x}) \tag{1}$$

where the subscripts denote the reading coil flux contributions arising from each of the two possible components of surface magnetization, M_x and M_y, using a conventional ring-type transducer. The y co-ordinate is normal to the surface, defined positive in the direction of the magnetic head. First only a single track will be considered. The fringing field set up along the storage surface track when the reading coil is energized in the manner described for the reciprocity relation will have the vector components $H_x\bar{x}$ and $H_y\bar{x}$. Then, according to the principle of reciprocity, H_x is a function giving the sensitivity of the reading coil to horizontal surface magnetization and H_y gives the reading coil sensitivity to vertical magnetization.

It can be shown that

$$\phi_h(\bar{x}) = K\int_{-\infty}^{+\infty} M(x-\bar{x})H(x)dx \tag{2}$$

where M and H are vectors combined by a scalar or dot product operation.
Then

$$\phi_{hx}(\bar{x}) = K\int_{-\infty}^{+\infty} M_x(x-\bar{x})H_x(x)dx \tag{3}$$

and

$$\phi_{hy}(\bar{x}) = K\int_{-\infty}^{+\infty} M_y(x-\bar{x})H_y(x)dx \tag{4}$$

where the limits of integration are in practice only for the range over which H_x and H_y are significant. K is a constant proportional to the surface thickness, assuming uniform saturation throughout the surface layer.

Now to clarify the import of these formulas consider an idealized case where $M_y = 0$ and there is a step change

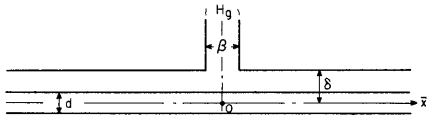


Fig. 2. Idealized magnetic head model

The pole faces extend to infinity.
β = gap size
δ = spacing factor
d = surface thickness
H_g = gap field

in horizontal magnetization M_x from $-M_s$ to $+M_s$. Then

$$\phi_{hx} = 2KM_s\int_{\bar{x}}^{\infty} H_x(x)dx \tag{5}$$

and as

$$e(\bar{x}) = vNd\phi_{hx}/d\bar{x} = KvN\int_{-\infty}^{+\infty} H_x(x)\frac{\partial M_x(x-\bar{x})}{\partial\bar{x}}dx \tag{6}$$

then,

$$e(\bar{x}) = e(vt) \propto vH_x \tag{7}$$

This relation states that the output voltage or pulse response wave form will resemble in time the static field distribution function H_x, i.e., the signal amplitude for a given position of the step change in magnetization relative to the magnetic head gap centerline will be proportional to H_x, a weighting function, at that point. Similarly, the output signal arising from a recorded step function change in M_y would be

$$e_y \propto H_y(\bar{x}) \tag{8}$$

where the subscript on the output voltage indicates that this contribution is due to M_y. Where both components of magnetization exist the two output signal components may be algebraically added to give the output voltage wave form, for $e = e_x + e_y$.

Determination of H

A rapid way to get estimates for the weighting functions H_x and H_y is to make rough field plots for the gap fringing field. Here there is no concern about local saturation effects since these maps are to apply for very weak fields. Qualitative information is generally adequate and, normally, fine detail would not be justified. It will be seen however, that in general a factor such as reading coil location is extremely important. The estimates for these fields from such mapping indicates their important qualitative features and these have been confirmed by a powder pattern technique of field observation.

Figs. 2 and 3 show ring-type structures and qualitative determinations of weighting functions, illustrating the importance of coil location and pole-tip shape. The magnetic potential lines shown in Fig. 2 do not, of course, actually apply within the coil or current source region. The orientation of the fringing field vector along the surface as indicated in Fig. 3, shows the influence of narrowed pole tips in reducing the base-line width (neglecting "overshoots") of H_x.

It is apparent that the information concerning H could be presented in terms of magnitude and angle rather than in the form of the scalar components H_x and H_y. Fig. 3 gives an indication of the angular variation of $H(\bar{x})$ and attention will be given to this

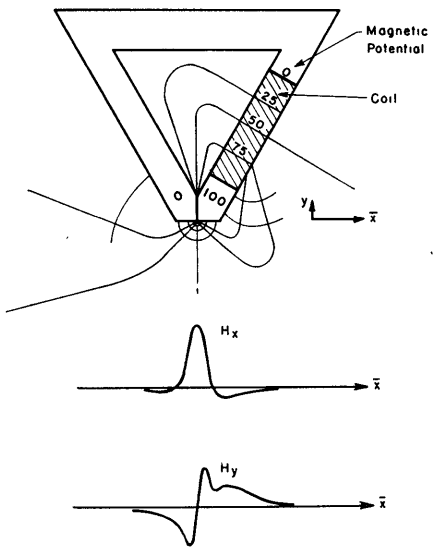


Fig. 3. Field plot showing influence of coil location on H_x and H_y

aspect of the field H later. It will be noted that with a ring-type structure, giving longitudinal recording, the principal field component will be H_x. The writing process will be considered later but it should be clear that if only horizontal magnetization occurred when writing then the only response function of interest would be H_x and conversely if $M = M_y$ then only H_y is of concern.

Voltage and Weighting Function Integrals

Certain statements may be made about the output voltage-time pulse area, other than its independence of

v, in terms of the weighting functions H_x and H_y. Consider a step change in M_x from $-M_s$ to $+M_s$. Then:

$$\phi_h(-\infty) = KM_s \int_{-\infty}^{+\infty} H_x(x)dx \tag{9}$$

and

$$\phi_h(+\infty) = -KM_s \int_{-\infty}^{+\infty} H_x(x)dx \tag{10}$$

Thus if

$$\Delta\phi_h = \phi_h(-\infty) - \phi_h(+\infty) \tag{11}$$

$$\Delta\phi_h = 2KM_s \int_{-\infty}^{+\infty} H_x(x)dx$$

but of course

$$N\Delta\phi_h = \int_{-\infty}^{+\infty} e(t)dt \tag{12}$$

Hence a relation is seen between the output single voltage-time area and the area under the corresponding weighting function. This immediately gives some insight into the possible degree of overshoot area to be expected. Further it reinforces the arguments for the importance of reading coil location since if the coil is so oriented that a uniform horizontal flux field tends to give zero flux linkages then the negative overshoot area must as a consequence be of the same size as the main pulse area; this applies here to both the weighting function and the output signal which would be proportional. These same considerations can be as simply extended to an M_y magnetization component, involving a consideration of the H_y function.

Adjacent Track Pickup

The sensitivity of the magnetic head to adjacent track magnetization can be considered by use of the reciprocity principle in the same manner as described, both with and without intertrack shielding. Let z be the transverse co-ordinate and let $z = 0$ at the given track and $z = z_1$ at an adjacent track. Note that since $H_x(z) = H_x(-z)$ the worst case for two adjacent tracks, one on either side of the magnetic head, would arise when both were similarly recorded. Now e.g., $H_x(\bar{x})$ at $z = z_1$ is not only considerably reduced in amplitude over that at $z = 0$ but its relative extent is considerably greater. Since the changes in surface saturation along a track must be alternating, then as the density of these saturation changes increases, there will tend to be a greater and greater effective cancellation of the individual signals. Thus due to the low resolution of a magnetic head with respect to neighboring tracks the lower density patterns are the most troublesome.

Finite Region For Surface Magnetization Change

An approach to an estimate for the significance of neglecting the actual surface magnetization and using an idealized step function change will be indicated by assuming $M = M_x$ and that the change in saturation from $-M_s$ to $+M_s$ occurs over the distance x_1 in a uniform manner. Let ρ equal the effective width of H_x. ρ then gives a measure for reading resolution and is equivalent to the factor λ defined earlier for a characteristic output voltage response. Now

$$M_x = -M_s + 2M_s x/x_1 \qquad 0 \leqslant x \leqslant x_1 \tag{13}$$

$$M_x = -M_s \quad x \leqslant 0; \quad M_x = +M_s \quad x \geqslant x_1$$

then

$$\partial M_x(x-\bar{x})/\partial\bar{x} = -2M_s(x-\bar{x})/x_1 \tag{14}$$

$$\text{for } 0 \leqslant (x-\bar{x}) \leqslant x_1 \text{ or } \bar{x} \leqslant x \leqslant \bar{x}+x_1$$

and

$$= 0 \text{ otherwise}$$

Now substituting these conditions into equation 6 the following expression is obtained for e $(\bar{x})$:

$$e(\bar{x}) \propto \frac{\int_{\bar{x}}^{\bar{x}+x_1} H_x(x)dx}{x_1} \tag{15}$$

which clearly agrees with the earlier expression in the limiting case of a step function magnetization change. The influence of x_1 on the output signal can be quickly estimated. For any value of x_1, e$(\bar{x})$ can be obtained by averaging $H_x(\bar{x})$ over the interval x_1 for a set of values of $\bar{x}$. This procedure is readily done by graphical means.

The principal influence of a finite x_1 is of course to reduce the signal peak and increase the base-line pulse width. It is evident that for an x_1 only several times less than ρ the approximation of the saturation change by a step function is very good. Experimental attempts to determine a magnitude for such a factor on oxide surfaces indicate that the actual situation is even more favorable to the foregoing approximation. This is in accord with earlier statements regarding the importance of reading resolution as compared to writing definition, due to the surface saturation characteristic, since x_1 is a measure of the latter while ρ gives a measure of reading resolution.

Gap Size, Spacing, and Surface Thickness

The inclusion of these parameters in the theory will be outlined by consideration of the simplified head model shown in Fig. 2 and the assumption of a recorded step change in longitudinal magnetization. The extension of this development to other cases is obvious and it particularly lends itself to a ready description of the manner in which gap size, spacing, and surface thickness enter into the determination of the characteristic response. Here, only H_x need be considered.

First neglect the surface layer depth. Then the following expression may be written

$$H_x(0) = g(\delta/\beta)H_g \tag{16}$$

where $g(\delta/\beta)$ is a monotonically decreasing function and $g(0)$ is somewhat less than 1.0. The function g gives the gap fringing field attenuation. The gap size and spacing distance enter in the manner shown for this particular model for here parameter variation actually only involves scaling, a single fringing field plot sufficing. Thus if both β and δ are changed in the same ratio this merely amounts to a magnification or reduction of the entire field plot and the relative field attenuation given by g would remain the same. For an actual head structure the fringing field dependency upon these parameters can again be estimated fairly quickly by mapping. Now almost the entire magnetomotive force will appear across the gap when the reading coil is energized. Thus, $H_g\beta = Ni$. Then

$$H_x(0) \propto g(\delta/\beta)/\beta \tag{17}$$

This relation indicates the dependence of the output signal amplitude, for a step function change in M_x, upon the gap size and spacing. If both these factors are halved then g remains the same and the relative magnitude of the peak value of the weighting function, and hence the output signal, is doubled. It is clear that for this idealized head the above reductions are equivalent to scaling the field plot by a factor of 1/2 in each direction and therefore the relative pulse width will be reduced to 1/2 its former value. Thus the weighting function integral value is preserved as would be expected from the previous discussion on this subject.

The surface thickness may be included in the following manner. Let $H_x(\bar{x})$ be replaced by $\overline{H_x}$, the average value of H_x over the surface depth. Now it was pointed out that the constant K in the expressions for flux is proportional to the surface thickness, for uniform magnetization throughout the surface layer, since the total magnetic field source coupling with the reading coil for any value of $\bar{x}$ is proportional to d. Like-

wise K would be proportional to the head width with the two dimensional type of problem here. Then

$$e(0) \propto d\bar{H}_x(0) \tag{18}$$

When $H_x(0)$ is a good approximation for the average value of H_x over the surface layer cross section in the plane $\bar{x} = 0$ then the following expression, indicating the signal amplitude dependence upon gap size, spacing, and surface thickness, may be written

$$e(0) \propto g(\delta/\beta)d/\beta \tag{19}$$

Note that δ is measured to the center of the storage surface layer.

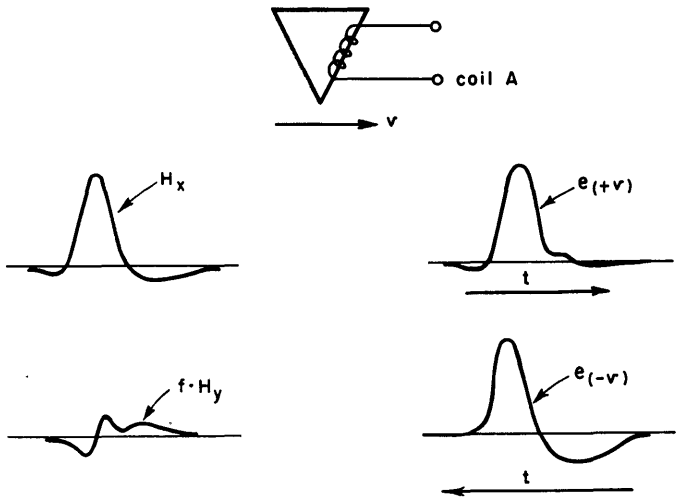


Fig. 5. Effect of oriented surface magnetization

$e_{(+v)} \propto H_x + f\,H_y$
$e_{(-v)} \propto H_x - f\,H_y$
where $f = M_y/M_x$

Orientation of Magnetization

Referring to Fig. 3 it is seen that with a ring-type magnetic head structure a qualitative classification of response functions would indicate H_x as an even function and H_y as essentially an odd function, insofar as such a division is concerned. Now consider a step change in saturation magnetization where now the direction of saturation is at some angle with respect to the horizontal, less than 90 degrees. Let M_x be the principal component, assumed a positive step change, and first consider the case where M_y is also a positive step change. Now since the actual change in surface magnetization is composed of these two independent terms the output response is the sum $e_x + e_y$, each term of which is proportional to its corresponding weighting function, adjusted by the actual magnitude of the associated step change in magnetization. As is clear from an inspection of Fig. 5 where this composition of signals is shown, the resultant output signal, $e(+v)$, is unbalanced relative to H_x. Now regard $e(-v)$ in Fig. 5 showing the same composition with only the sign of M_y changed. It is evident that this is a different characteristic output signal possessing another wave form unbalance. The base-line wave form of the output signal is considerably different in these two cases. The effect is easily visualized in terms of the angular variation of $H(\bar{x})$. It follows as a direct extension from the earlier theory that for an oriented step change in magnetization the output signal for each value of $\bar{x}$ is proportional to the scaler product of $H(\bar{x})$ and a unit vector in the direction of M or, in other words, proportional to the magnitude of the component of H in this direction. Now regard in Fig. 4 the region giving rise to the larger H_x "overshoot," this region being located on the coil side of the magnetic head. It is seen that H is oriented along an axis extending into the fourth quadrant, obtained by a clockwise rotation of the horizontal axis. Then for M oriented along an axis traversing the first quadrant, corresponding to the case giving rise to $e(+v)$ in which both M_x and M_y have the same sign, it is evident that the scalar product occurs between two vectors at nearly ninety degrees with respect to one another. The change in sign of M_y, used for obtaining $e(-v)$, rotates M by ninety degrees and therefore orients it along an axis running into the fourth quadrant and much more closely parallel to the axis giving the orientation of H in the region of concern. Thus, it would be expected that the "overshoot" signal response of $e(-v)$ would be considerably more pronounced.

The significance of these two characteristic wave forms will become apparent if the writing process is briefly considered. The storage surface passes through a 2-dimensional recording field when writing. The important portion of this field as far as writing is concerned is the field that the surface passes through on leaving the gap region of the magnetic head. Here a surface cross section horizontally saturated in the gap itself is subject to a not inappreciable vertical magnetizing component as it passes out of the vicinity of the head. Then there will be a tendency for an orientation of the saturated surface magnetization away from the horizontal. Further this orientation will depend on the direction of motion of the surface during writing. The two possibilities are indicated in Fig. 6. It can be seen that the two characteristic output voltage wave forms might be expected, the particular one depending upon the direction of surface motion during writing, except for an entirely symmetrical recording structure. A reversal of the direction of relative motion on reading adds no additional wave forms but merely produces a mirror image of the wave form previously obtained. Fig. 7 shows actual characteristic readback voltage wave forms showing the influence of the direction of surface motion. A delta-type head structure was used with a coil on one leg as indicated in Fig. 5. The leading and trailing leg designations indicate that the coil leg is the one first passed by a surface point and vice versa respectively. Note that the type of oriented

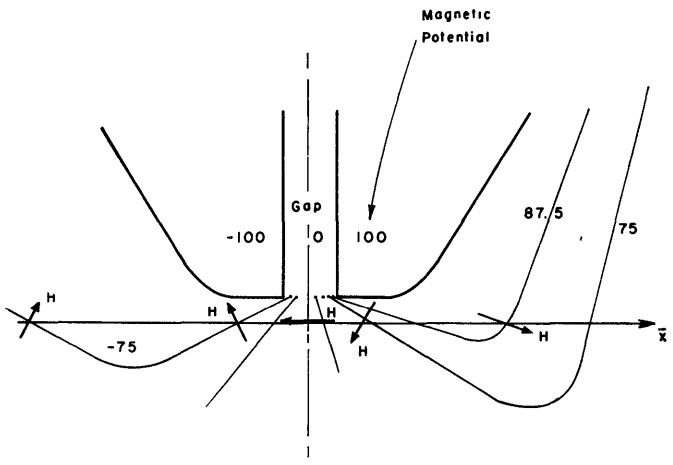


Fig. 4. Influence of narrow pole tips on senstivity function H

magnetization expected is in agreement with that necessary to account for these wave forms in this theory.

Theory as Applied in Design

A close interaction existed between experimental work and the refinement and application of the theory. The concepts and relations described readily lend themselves to the interpretation of existing experimental results and further, from them performance predictions can be made for contemplated structures. The concept of an oriented saturation magnetization was essentially developed as outlined to explain early experimental results for which an assumption of a longitudinal step change in magnetization was inadequate. While this orientation factor could not be experimentally determined, this hypothesis of an oriented saturation magnetization led to a definite pattern of behavior which could be examined. All the tests indicated consistent agreement with this picture. The procedure for head design was to use only the step function response for

Fig. 6

evaluation. Final tests were carried out with several structures using a recording unit and the magnetic head performance under operational conditions correlated excellently with the step function response results.

On the basis of the proportionality between the output signal and the weighting functions H_x and H_y the importance of reading coil location and pole-tip shape is readily apparent. Further, with oriented magnetization, even with a symmetrical core structure, the leg upon which a reading coil may be wound is of paramount importance. The prior discussion and the sketches of Fig. 5 indicate that a favorable location for a leg-mounted reading coil would be the "trailing" leg with respect to the surface motion, since an overshoot compensation is possible.

This design theory indicates quite obviously the desirability of reducing gap size, spacing between head and surface, and surface thickness from the point of view of resolution. In the experimental results presented these parameters were fixed and hence these results show an optimization within this framework.

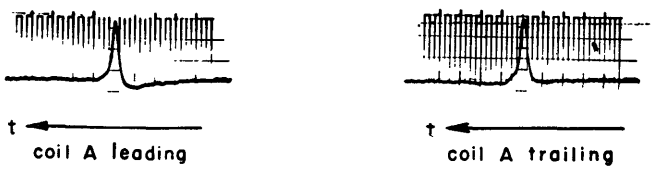
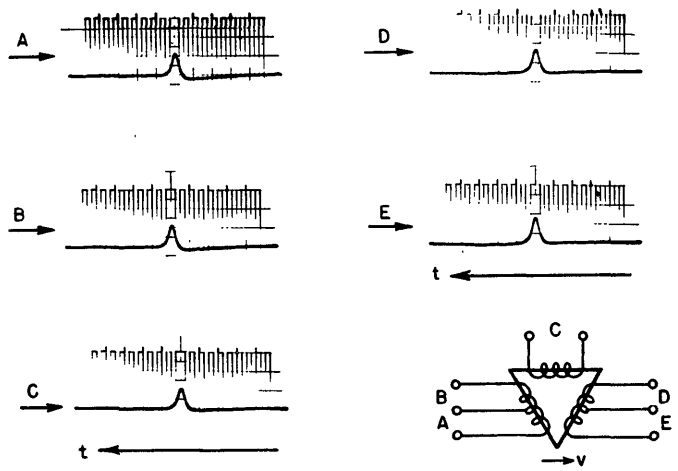


Fig. 7. Actual wave forms showing effect of direction of surface motion

Scale: 3.5 mils per division

Fig. 8. Delta-type head

Scale: 3.5 mils per division



Experimental Results

The following wave forms were obtained in the course of the head design program for the IBM *305*. Some results are from structures constructed primarily to test the usefulness of the conceptual approach described here. Some results then merely illustrate certain points. Following discussions of these results the selected head design and its performance will be indicated.

Fig. 8 shows a delta-type head and the signals obtained on readback from coils located at various positions around the delta, for the same step function change in saturation. The relative influence of coil position on overshoot can be noted. An indication of the compensation effect for the trailing leg coil position on overshoot can be seen. The leading and trailing leg identifications apply to the head orientation with respect to surface motion during writing. The signals obtained when a completely different ring-type head structure was used for writing with, however, the same gap size, indicated that the recorded magnetization is primarily dependent on the gap field and not greatly influenced by pole piece shape.

Fig. 9 shows the characteristic output signal for two delta structures, both with the favorable trailing-leg coil location but one (*S-25*) possessing a pointed-type pole piece design compared to the other (*S-29*) whose pole tip edges, adjacent to the surface, are withdrawn much more gradually. The compensation of overshoot is such that insofar as overshoot is concerned the responses are nearly equivalent. However, the greatly increased pulse steepness in base-line approach and consequent reduction in the base line pulse width is noticeable. This result agrees with that expected from the estimated form of H_x. In one sense this pointed-type head is a tailored design for the problem at hand. As would be expected, if these magnetic heads were deliberately spaced much closer the compensation action may not be nearly as satisfactory. This was the case, the relative magnitude of the overshoot increasing. This effect was more pronounced the narrower the tips.

Design Results

The final selected design was a head with a 2-mil radius tip and the coil assembly located on the trailing leg, as indicated in Fig. 10. The operational tests at increasing bit densities clearly established the importance of the sharp base-line approaches for the characteristic pulse response, for other structures while lacking this feature did eliminate the overshoot as well. The performance of this head under the nominal operational conditions of a 1-mil spacing, 2-mil gap, and a 1-mil surface layer resulted in a conservative estimate of about a 13-mil base line pulse width for the characteristic response and a peak signal to peak overshoot ratio of greater than 20. Nominally, no pattern modulation, or readback amplitude attenuation, would be expected below 150 bits per inch (bpi) storage density. Fig. 10 shows a pattern readback at 115 bpi, a figure which represented an equipment limita-

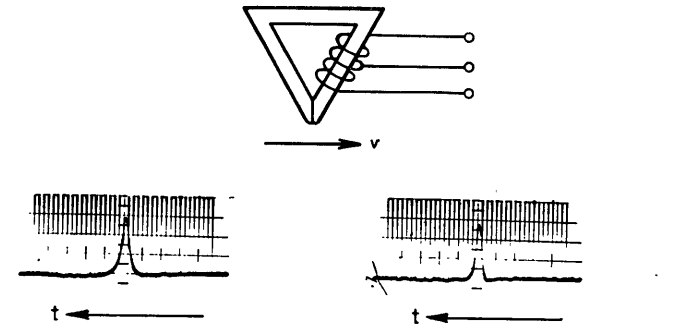


Fig. 9

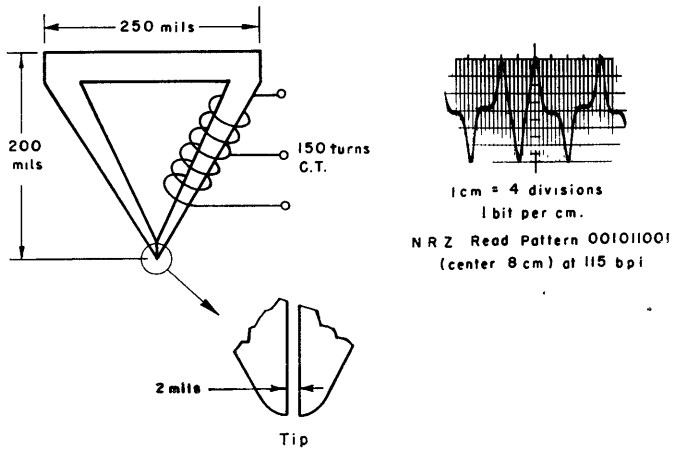


Fig. 10

tion at the time. The specified maximum operating density was 100 bpi.

Summary

A method for magnetic head design has been presented which stresses a conceptual point of view regarding the recording process which gives considerable insight into the problem. Qualitative principles are made available as guides in design and allow a ready approach to evaluation and interpretation. Further, these methods can be usefully applied from crude estimates to any desired degree of refinement, and they lend themselves directly to a study of recording structures of a more radical nature.

References

1. Magnetic Drum Recording of Digital Data, **A. S. Hoagland.** *AIEE Transactions*, vol. 73, Sept. 1954, pp. 381–85.

2. Field Measurements on Magnetic Recording Heads, **D. L. Clark, L. L. Merrill.** *Proceedings, Institute of Radio Engineers*, New York, N. Y., vol. 35, Dec. 1947, pp. 1575–79.

3. Studies on Magnetic Recording (Part II), **W. K. Westmijze.** *Philips Research Reports*, Eindhoven, Netherlands, vol. 8, June 1953, pp. 161–83.

A Terminal for Data Transmission Over Telephone Circuits

ENOCH B. FERRELL

THE Bell Telephone Laboratories has for some time been interested in digital transmission as a means of communication between automatic switching systems.

In a recent experiment, a simple terminal for data transmission has been demonstrated. Between two such terminals it would be possible to send data back and forth over ordinary telephone channels at the rate of 750 bits per second, or 1,000 words per minute.

The demonstration equipment involves magnetic tape to magnetic tape transmission using amplitude modulation of a 1,200-cycle carrier. It employs a 7-bit self-checking code.

For a long time the Bell System has been interested in sending numbers from one place to another over ordinary telephone lines. When you place a call to someone on the other side of town, your central office must pass that party's number to his central office as part of the instructions for completing the call. This has been done over the same trunk that is used a few seconds later for the actual conversation. It has been done at speeds of less than ten bits per second—speeds comparable to that of the telephone dial. This has been extended to cover calls between cities, over a large part of our long-distance network. The speed has been doubled by the use of multifrequency signaling, which sends two out of five frequencies in the voice band.

Some time ago a study was made of various methods of transmitting digital data at considerably higher speeds. Part of this study is reported in a paper "Transmission of Digital Information over Telephone Circuits"[1] by Horton and Vaughan. With the coming of age of electronic digital computers and data processing techniques, it is apparent that this high-speed transmission may serve needs other than those of the control of telephone switching.

Bell research people have been considering what sorts of data transmission could occur over the great variety of transmission facilities that exist in the telephone system. Teletypewriter which is already in considerable use might be mentioned. For the most part a teletypewriter channel uses a narrow frequency band of something like 150 cycles and transmits digital information at the

Enoch B. Ferrell is with the Bell Telephone Laboratories, Inc., Murray Hill, N. J.

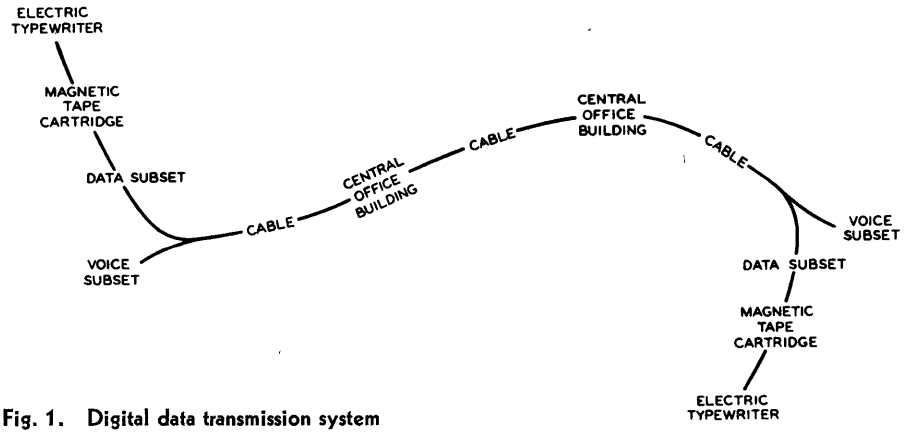


Fig. 1. Digital data transmission system

rather low speed of 50 or 75 bits per second. If the full width of a voice channel is used, a speed in the order of 1,000 bits per second can be reached. This can be done over existing telephone message circuits, the same kind of connection that people all over the country use every day to talk to each other. The use of this kind of data transmission in connection with centralized accounting schemes, centralized inventory control, the rapid handling of mail order business or insurance policy changes, and the like is easy to imagine.

For some of these purposes, or for others that may not have been thought of yet, still higher speeds may be needed. Specially treated lines may be necessary such as are used for certain data transmission jobs already undertaken.

In some of the carrier systems, several speech channels are modulated up in frequency and placed side by side to form what is called a group. This group then passes through cables, amplifiers, volume controllers, and equalizers as a single unit. The bandwidth of such a group is a dozen times that of a single speech channel, and its data transmission speed is proportionately higher. Perhaps these groups should be made directly available for some special data transmission jobs.

In the coaxial cable and microwave radio relay systems, these groups are combined into supergroups, and these supergroups into systems. One can imagine that someday whole batteries of high-speed computers might be scattered across the country, connected together by wave guides.

In the meantime some experimenting along these lines has been done. This is a report of a research group, and, therefore, does not describe any available service offering. If such a service becomes available it is possible, even probable, that the form will be different from that of the experiment.

There are many problems to be solved. As the business of data transmission is approached there appear not only the obvious problems of transmission, but also the problems of technical compatibility between the business machine and the telephone network, and the problems of methods compatibility between office or computing procedures and telephone operating practices. It is highly desirable for the designer of business machines and the designer of business methods to work with the telephone man in solving these problems. This kind of co-operation has been going on for a long time. Currently a technical committee originated within the joint computer committee is quite active on just such problems.

Recent experimental work involves a system something like that of Fig. 1. A magnetic tape with the digital information on it is prepared on a machine that will be called the business machine. This tape is removed from the business machine and placed in a data subset, which is connected by an ordinary telephone circuit which is established through ordinary switching exchanges to a similar subset at a distant location.

The data set at one end reads the tape and transmits the information at high speed to the other end. The data set at the other end receives this information, checks it for plausibility, confirms reception or requests repetition as needed, and writes the received information on another magnetic tape. When this operation is completed, the tape can be taken from the data set, put in the business machine, and a hard copy can be printed out at the receiving end.

One of the important reasons for use of magnetic tape comes from the need for a speed translation. The normal telephone channel is capable of transmitting data much faster than a human being can operate a keyboard. But it lacks a great deal of being able to transmit data at the tempo of our modern digital computers.

It appears, therefore, that in most applications, some such speed translation will be desirable. Magnetic tape provides an excellent means for writing at one speed and reading at a different speed. Magnetic tape can be easily stored and its associated equipment is quiet in operation.

This was the general plan. These are a few of the details. In the experiment an electric typewriter was used. Normally, it will prepare hard copy and paper tape from keyboard operation, or will print out from paper tape. It has been modified so that it will prepare hard copy, paper tape, and magnetic tape from the keyboard, and so that it will print out from either paper tape or magnetic tape.

On the magnetic tape a 7-bit serial code is used. The 7-bit code is an odd-parity check code. Actually this is restricted to a 3-or-5-out-of-7 code. At the beginning of each line the modification of the typewriter automatically inserts a 2-character START code on the tape. At the end of the line, or when the carriage return key is struck, it writes the carriage return character plus a 2-character STOP code. The normal typewriter mechanism performs the actual carriage return.

In the data set, the tape signals are used to produce simple amplitude modulated pulses of 1,200-cycle carrier. See Fig. 2. With a useful band of 1,200 ±500 cycles, the theoretical limit on speed is a bit rate of 1,000 bits per second. When a factor of safety is put in, and when time

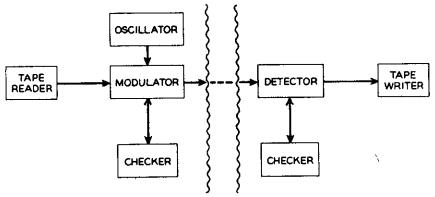


Fig. 2. Functions of the data subset

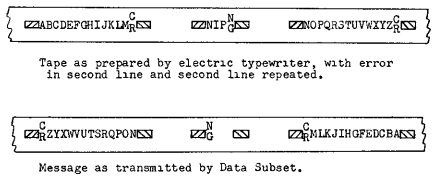
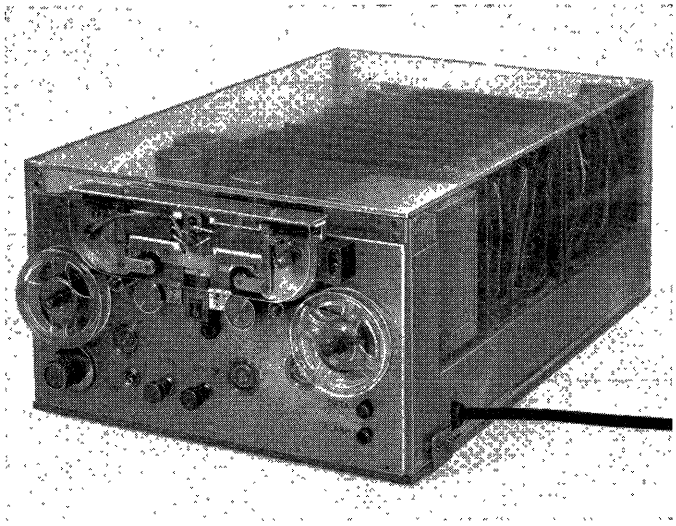


Fig. 3. Illustration of method of handling errors

is taken for start signals, stop signals and confirmations, there is an effective speed of a little less than 600 bits per second. This, on the 7-bit code, is a little less than 800 words per minute. With future developments, it may be possible to increase this speed.

The 2-character start signal is



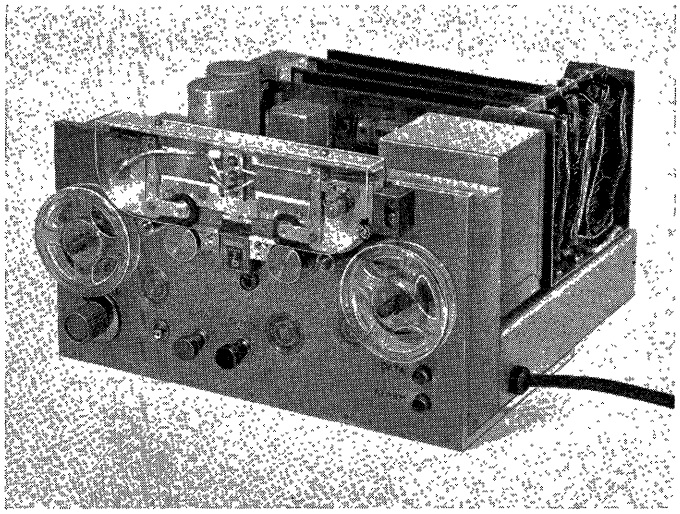


Fig. 4 (above). Modified electric typewriter

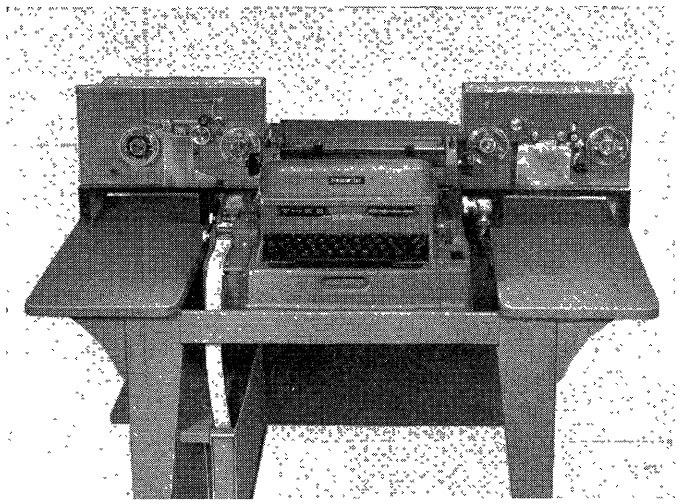


Fig. 5 (above right). Experimental transmitter-receiver unit

Fig. 6. Experimental data subset, complete with checker

11111111111001. On the long string of eleven 1's volume control is established, and synchronization on the final 1. A local clock oscillator in the receiver is kept in step after that by the bit structure of the message. With all characters in the age restricted to a 3-or-5-out-of-7 code, the longest possible string of zeros is eight. Such a string must be both preceded and followed by strings of at least three 1's. This permits satisfactory synchronization of the clock oscillator, and satisfactory volume control on the amplifier.

The magnetic tape is always reversed between writing and reading. That is, the tape is always read backwards. This avoids rewinding of the tape. Since there is one reversal at the transmitting end and one at the receiving end, a total of two which is an even number, the final copy comes out right end to. These tape reversals also permit a rather interesting method of handling errors.

If the typist makes an error and recognizes it she strikes the "Error" key. This prints an ERROR code and the STOP code on the tape, and operates the typewriter carriage return. Then she types the line again, this time presumably complete and correct. The magnetic tape now looks, symbolically, like the tape in Fig. 3.

The STOP code is the exact reverse of the START code. The transmitter, in handling this block, first finds a START code, then a block of the message, and then a STOP code. It transmits these signals, waits for confirmation that the receiver has made a satisfactory odd-parity check on each character, and then, after receiving that confirmation, starts again. It reads the START code, and then finds the ERROR code. This means that the block now coming up contains errors. So the transmitter simply stops and waits for that false block to get out of the road. This is quite all right, of course, because the corrected block, although written later by the typist, has already been sent and acknowledged over the line.

In a similar manner, if a transmission error, as from noise on the line, produces an even-out-of-7 character at the receiver, the receiver remembers this till the end of the block. Then it writes a PLEASE IGNORE code at the end of this block, and asks the transmitter to back up and repeat that block. The typewriter, in printing out, now prints the good block, ignores the bad block, and thus produces only correct copy.

Fig. 4 shows the electric typewriter with the tape-handling equipment and circuitry used in its modification.

Fig. 5 shows the experimental transmitter-receiver without the error detection circuits. Fig. 6 shows the combination transmitter-receiver and checker.

The circuitry used here is similar to that now common in many digital computers. It is based on the use of transistors, diodes, and magnetic cores. A general idea of its size can be obtained from the fact that in the complete transmitter-receiver checker a total of about 180 transistors are used.

In constructing and testing the data set mentioned here, a good many people have contributed in many ways. A few are H. W. Bode, T. L. Dimond, and J. R. Pierce who suggested the project, and W. D. Lewis and other members of the Switching Research Department of the Bell Laboratories who gave it advice and support. W. A. Malthaner has served as project engineer. J. E. Schwenker and G. P. Darwin have been responsible for much of the circuitry. J. F. Muller has been responsible for the tape-handling equipment. It is hoped that some of them will prepare more detailed reports on their work in the not too distant future.

Reference

1. Transmission of Digital Information over Telephone Circuits, **A. W. Horton, Jr., H. E. Vaughan.** *Bell System Technical Journal*, New York, N. Y., May 1955.

The Use of the Charactron with ERA 1103

BEN FERBER

THE Charactron[1,2] tube was invented by Joseph T. McNaney and developed by Convair since 1950. The main purpose for installing a Charactron on Convair's ERA *1103* computer was for real time simulation. However, other valuable uses for the Charactron on the *1103* have been found.

Physical Characteristics

This Charactron, with its cathode-ray display tube, type *C7A*, can display alphanumeric characters at a rate of 10,000 characters per second. The equipment which includes a cathode-ray tube with 7-inch-diameter screen can be used with either one of two cameras, easily interchangeable in a matter of a few minutes. Fig. 1 shows the Charactron with a Beattie camera using 35-millimeter film in a magazine. It is possible to remove the exposed film without removing or exposing the unexposed film. Fig. 2 shows the type of construction of the main body of the equipment. Fig. 3 shows the tube mounted vertically and viewed by the camera using a mirror mounted at 45 degrees to the camera lens and to the tube screen. The screen may be viewed through a filter during operation without impairing the results. The four drawers contain the power supplies.

The second camera, the Kenyon camera shown in Fig. 4, is a camera and photo laboratory combined. All operations, exposing, developing, fixing, and projecting, are performed in parallel. While the computer is calculating and displaying one page of answers, the camera is fixing and developing the previous pages. This process takes about 2 seconds, and if the calculations take more than 2 seconds a page, then the fixing and developing does not hold up the process at all. The finished film is extruded and can immediately be viewed on a film reader such as a Recordak. Editing can be done at this point to determine which frames are to be enlarged and printed. Fig. 5 shows the Charactron, with Kenyon camera attached, connected to the ERA *1103*. A test generator is also included in this unit which enables alignment adjustments to be made without the use of the computer.

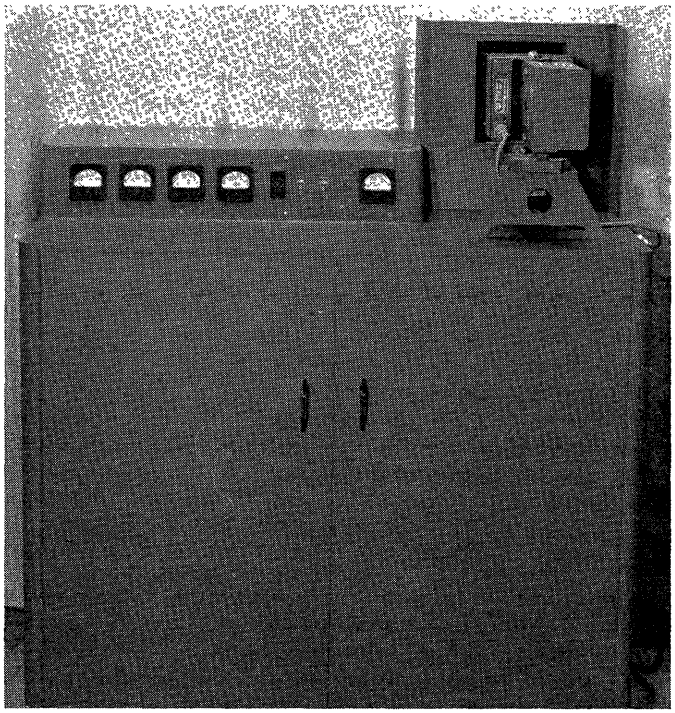
BEN FERBER is with Convair, a division of General Dynamics, San Diego, Calif.

A 6-bit code is used to select the proper alphanumerical character from a matrix in the Charactron tube. These characters are in a 6-by-6-array. Three bits are used for horizontal selection and 3 for vertical. A 20-bit code is used to position the characters on the face of the tube. Ten of these are used for horizontal selection and ten for vertical. Thus, a total of 26 bits defines the alphanumerical character and its position on the face of the tube. The *IOB* buffer on the *1103* has a capacity of 36 bits. The characters we chose to display are the numbers 0 through 9 and the alphabet not including the letters O or I. (The numbers zero and one do double duty.) A decimal point and a minus sign complete the list of characters.

Applications

As an aid in debugging, it can display the contents of memory, Fig. 6. Another common technique used in debugging a floating point program is a trace or automonitor, Fig. 7. This is an example of a concurrent trace giving the address, the command, and the result of each command. The trace operates at a rate of better than ten lines per second. The Charactron can also be used to edit input data. In this case, while the computer is calculating the results, the input data are plotted. Cases that show up with points obviously

Fig. 1



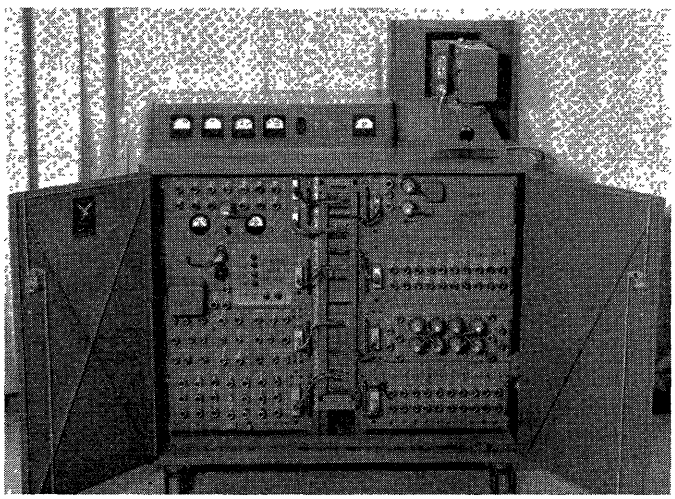


Fig. 2

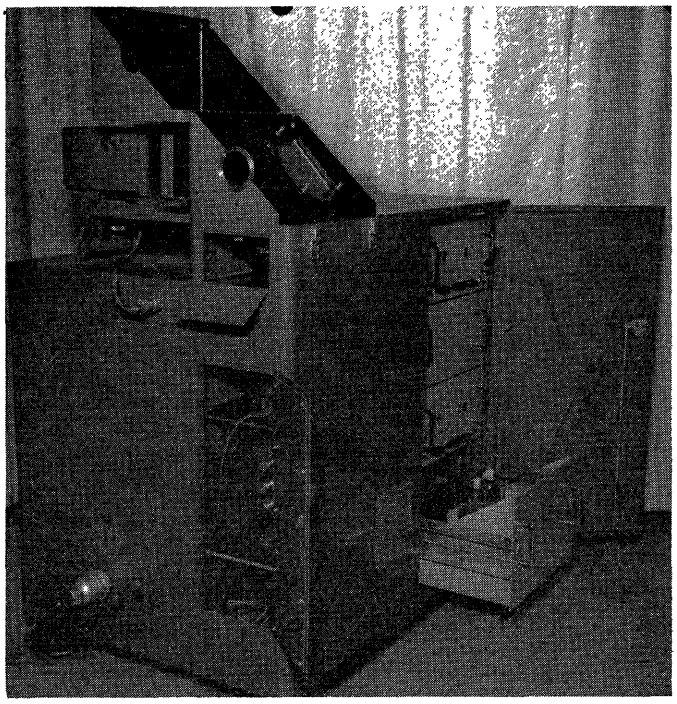


Fig. 3

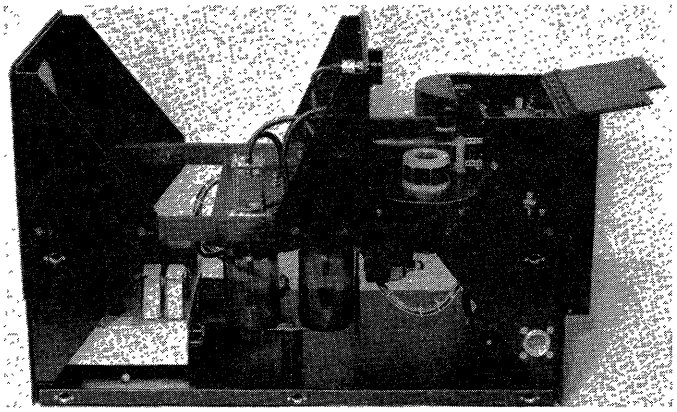


Fig. 4

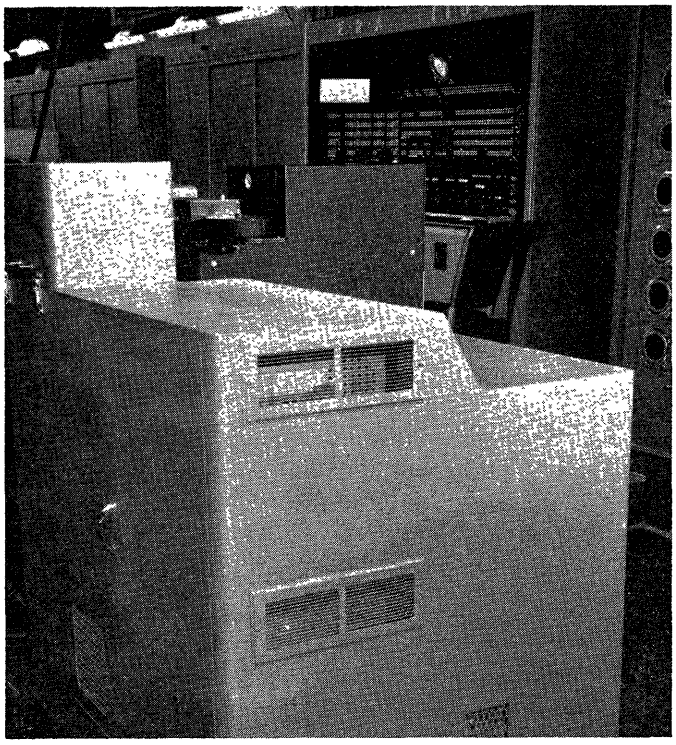


Fig. 5

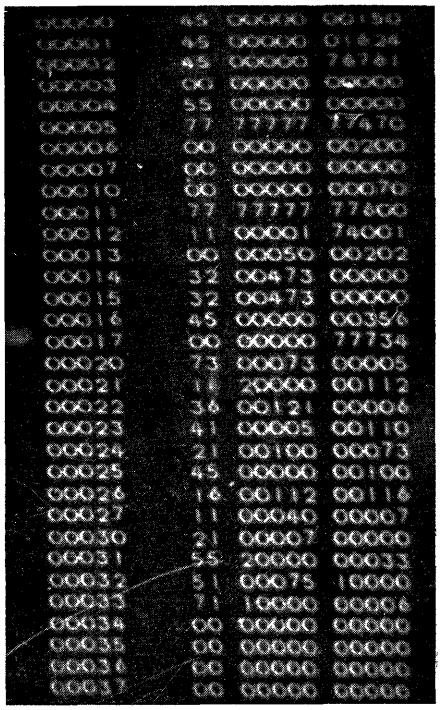


Fig. 6

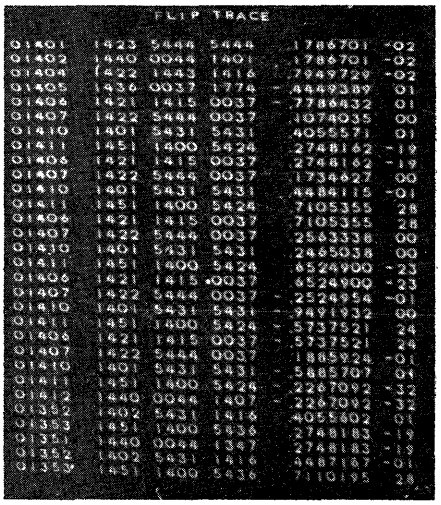


Fig. 7

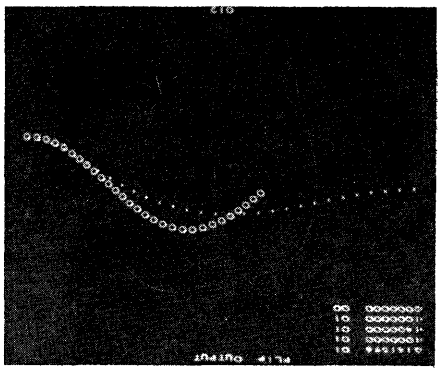


Fig. 8

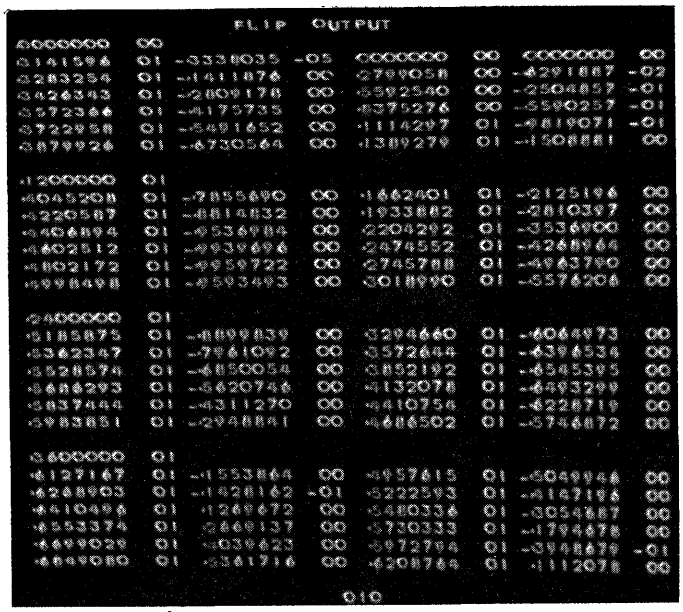


Fig. 9

out of line can be corrected and rerun.

Any of the characters can be used to plot graphs. Multiple graphs can be plotted on the same frame, Fig. 8. The input parameters can also be displayed on the same frame. These graphs represent the solution to two simultaneous differential equations. A table of the values plotted could be separate, Fig. 9.

A question often asked is, "How many characters can be displayed on one horizontal line?" With this installation 50 characters can be displayed horizontally. A Charactron with a 7-inch tube has been built which can display 100 characters per line with very fine definition and sufficient intensity for photographing at a rate better than 20,000 characters per second. It should be noted that the format is not limited with a Charactron. Answers can be printed in vertical columns. Each column, for example, could represent all variables at one time interval of integration.

Early this year, Convair expects to tie together a very large analogue computer with the ERA *1103*. Between the two will be the conversion equipment—analogue to digital and digital to analogue. This setup is for a real time simulation problem. Of course, the output plotters on the analogue will be used, but the output of calculation from the ERA *1103* are also needed. This output must be very fast because of the real time simulation. Since the magnetic tapes have inertia start and stop times which make them too slow, and the drum may not be large enough to store all the answers before the problem is finished, it is felt that the Charactron with its extremely high speed may answer this challenge.

References

1. The Charactron, **Joseph T. McNaney.** *Proceedings*, Institute of Radio Engineers, New York, N. Y. March 1952.

2. The Type C19K Charactron Tube and its Application to Air Surveillance Systems, **Joseph T. McNaney.** *Ibid.*, March 1955.

A New Tape Handler for Computer Applications

ROBERT BRUMBAUGH

THE rapid advances made in digital computer design within the past few years have, unfortunately, not been accompanied by a corresponding advance in the design of input-output equipment of comparable performance. The increasing scope of computer applications has further intensified the limitations imposed by available input-output equipment.

Magnetic recording tape, as a storage medium for digital information, is assuming a role of ever-increasing importance, and is now unsurpassed as an input-output medium for the rapid transfer of information. In addition, magnetic tape equipment has become an important element in automatic data-reduction systems.

Many types of magnetic tape handlers have been designed in the past, the great majority to meet a more or less specific application. As a result, extensive modification has often been necessary to adapt these units for other applications. In recent years, more versatile designs have been evolved to meet the increasingly diversified requirements for digital recording equipment. Although the new equipments represent a step in the right direction, these pioneering efforts were at times overly complex, and consequently caused a sacrifice both of reliability and economy.

For many years Ampex has concurrently pioneered in the recording of analogue signals on magnetic tape. Many of the problems in this field are very similar in nature, if not in degree, to those in the computer field. In addition to their useful analogue function, standard instrumentation recorders have many times been modified for application to computer systems; this approach is obviously not the answer to the increasingly stringent and refined requirements of the computer industry.

Believing that its extensive past experience could be applied to solve many of the increasingly difficult problems in the application of input-output equipment, Ampex Corporation initiated a program to develop a magnetic tape transport for computer use, providing versatility and reliability equal to that of the instrumentation recorder. Briefly, the most desired requirements of a tape transport for computer use are as follows:

Robert Brumbaugh is with the Ampex Corporation, Redwood City, Calif.

1. A wide range of tape speeds, track arrangements, and types of recording/reproducing heads must be possible with little or no change in the basic design.

2. The head assembly and associated tape tracking system must be of a precision and accuracy such that tapes will be completely interchangeable from machine to machine.

3. Reliability and simplicity must be achieved to reduce maintenance and costly "down time."

4. The start-stop time and distance must be kept low, and, perhaps even more important, consistency or start-stop characteristics must be maintained.

5. Price must be low in order to minimize computer accessory cost.

6. Operator skill and training must be minimized; this requires extreme simplicity in tape threading and controls.

7. A high-speed rewind function should be included.

8. A simple, functional appearance is required for integration of the tape handler into existing and future computer systems, many of which are intended for office rather than laboratory use.

A tape transport mechanism incorporating all of these objectives has been developed by the Ampex Corporation, and is shown in Fig. 1.

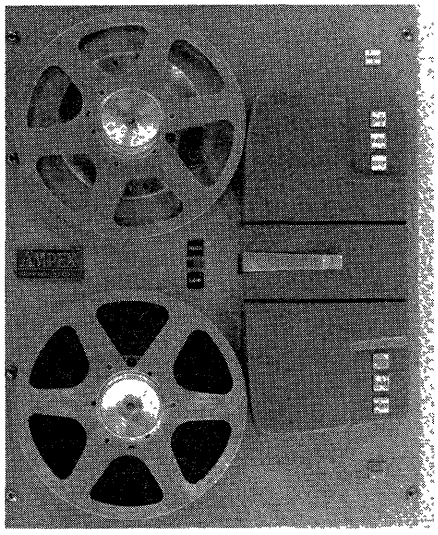


Fig. 1. New Ampex tape handler

Tape Feed System

One of the major problems in accomplishing a rapid start and stop of tape is the development of a reliable, smooth-responding, constant-tension tape feed. The tape feed system isolates the relatively large inertia of the reels from the inertia of the reels from the section of tape to be accelerated and driven past the record or playback head.

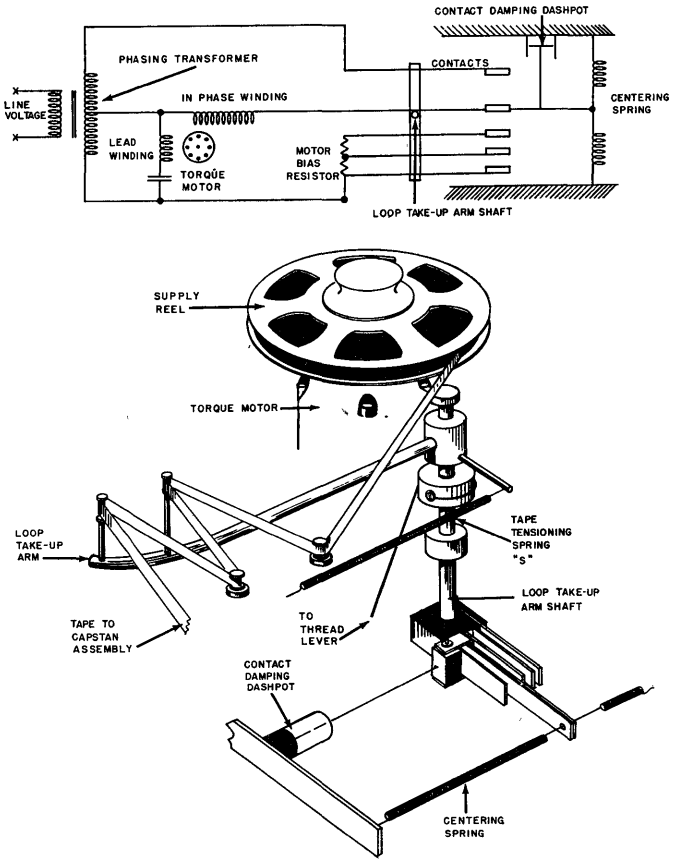
Before discussing the details of the system, it is well to review the development. An electronic-controlled servo-system offers the distinct advantage that the response can be controlled and adjusted quite easily. However, such systems are relatively complex and costly; as a result, it was decided to abandon such a system.

A step or "on-off" type of controller offers maximum reliability and simplicity, but for the most part the response leaves much to be desired. The system finally devised combines the best features of both: simplicity, reliability, and smooth response. The performance closely approximates a true proportional system with error-rate damping. A rigorous mathematical analysis of the system (see the appendix) yielded results closely following the actual performance.

Fig. 2 shows the essential details of the servo system. Only the left follower arm is shown for simplicity. The action of the system is as follows.

Quiescent or static tape tension is established by the tensioning spring *S*. The choice of the spring and the position of the spring lever arm were selected so that the tape tension would be almost constant over the full displacement of the loop take-up arm. Quiescent tape tension (3 ounces) is exactly balanced by a motor torque which is determined by the motor bias resistor *R*. The actual system has two such bias torque levels, and enough purposely introduced friction to eliminate hunting or contact bounce over the entire range of reel load.

Fig. 2. Tape feed system



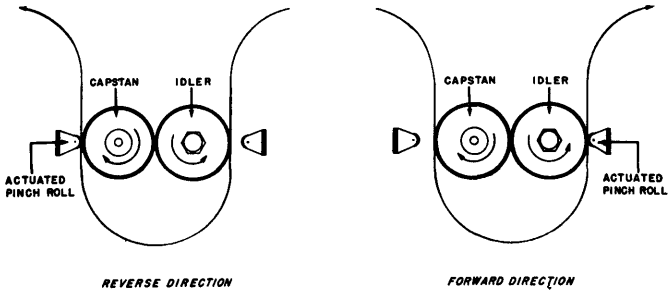
The take-up arm neutral position is established by the control contactor centering springs, and hence the position of the control contactor relative to the take-up arm. An input signal, of either a supply or demand of tape, from the capstan section, displaces the loop take-up arm and contactor. The contactor, in turn, switches the motor in-phase winding to the appropriate side of the servo phasing transformer. Full-power motor torque drives the reel in a direction which restores the loop take-up arm and contactor to its center or neutral position.

This system also offers a very convenient method for accomplishing the high-speed or rewind mode of tape motion. If, for example, we should disable or remove all power from the right hand servo, the right hand take-up arm will drift to the right, coming to rest against its stop. Under this condition the tape on this side is no longer under tension. The differential of tape tension between the two sides facilitates rewind from right to left. Stop, during rewind, is accomplished merely by restoring control to the disabled servo. Total rewind time of a standard 2,400-foot 10-inch reel is somewhat less than 2 minutes.

Table I. Specifications of Ampex Standard 1/2-Inch 7-Channel Digital Record Reproduce Head

Mechanical Specifications	
No. channels	7
Track width	0.032 inch
Track center-to-center distance	0.070 inch
Gap width	0.0005 inch
Gap vertical alignment tolerance	0.0001 inch
Vertical or azimuth tolerance	±1 minute of arc
Electrical Specifications	
Inductance (typical)	Outer tracks, 10 mh Inner tracks, 12 mh
Turns	400/leg, common center taps
Write current (tape saturation)	10 ma (zero to peak)
Read output volts	8 mv (peak-to-peak) at cps, 30 inches per second

Fig. 3. Tape drive



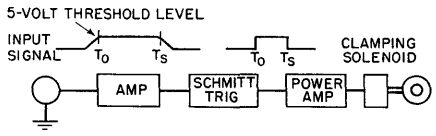


Fig. 4 (right). Control circuit block diagram

Are contacts really reliable and permanent when subjected to many millions of make-break cycles? First, the problems of metal transfer, pitting, and contact sticking are greatly reduced when the circuits are alternating current. Second, and possibly more important, when tape is demanded at rates of from 10–60 bursts per second, the servo integrates this motion into resultant or average tape speed. Under such conditions the contacts may break and make only several times per second. In actual life tests, under full power to an inductive load, the contacts have made upwards of 60 million cycles with more than 50 per cent material remaining for useful service.

Head Assembly and Tape Tracking System

One of the greatest problems facing the user of tape transports is interchangeability of tapes recorded on different machines. The problem basically is one of interchannel timing error, which, in turn, is caused by both static and dynamic factors. The static factors are almost exclusively in the province of head design—lack of consistency in gap alignment and azimuth. Any deviation in these quantities exacts a cost in effective pulse-packing density and interchannel time relationships. The Ampex Corporation has recently introduced a new technique for the manufacture of multichannel head assemblies, a technique which has reduced alignment and azimuth errors to values approaching instrument errors. Table I lists the new head specifications. The close tolerances to which the heads are manufactured eliminate any necessity for azimuth adjustment, and the attendant difficulty of insuring tape interchangeability.

The gap and azimuth tolerances, transformed into time at 30 inches per second between outer channels, would result in a maximum of 30 microseconds error introduced by the gap, and 5 microseconds error introduced by azimuth. The interchannel time displacement error between tape handlers would then be a maximum of 16 microseconds.

The dynamic interchannel time displacement error is also another important factor affecting pulse packing. The tape guides, head mount, and tape drive system exercise the most effect on this parameter. Once again, only design and precision of component manufacture can minimize the error. To take advantage of the excellent head tolerances, the guides and head must be mounted integrally on a plain surface which is held to the same or better dimensional tolerances. The capstan clamping idler must make accurate line-to-line contact with the tape, thus eliminating shear in the plane of the tape. Actual measurements show an outer-channel peak jitter of better than ±3 microseconds.

Tape Drive System

Fast start-stop tape motion is accomplished by clamping the tape to either one of two counter rotating capstans (see Fig. 3). The capstans are covered with 0.062-inch-thick rubber tires. The rubber eliminates tape marking and damage which might otherwise result from impact of the pinch roller. The pinch rollers are driven by two solenoids, which in turn are controlled by two power amplifiers. To eliminate the effect of variations in control signal amplitude and rise time, the solenoid power amplifiers are driven by a Schmitt Trigger circuit. Fig. 4 shows this relationship.

A d-c type of control was selected so that it could be driven directly by a gate or flip-flop.

Conclusion

The myriad applications of digital computation are just beginning. Many steps must be taken before the full

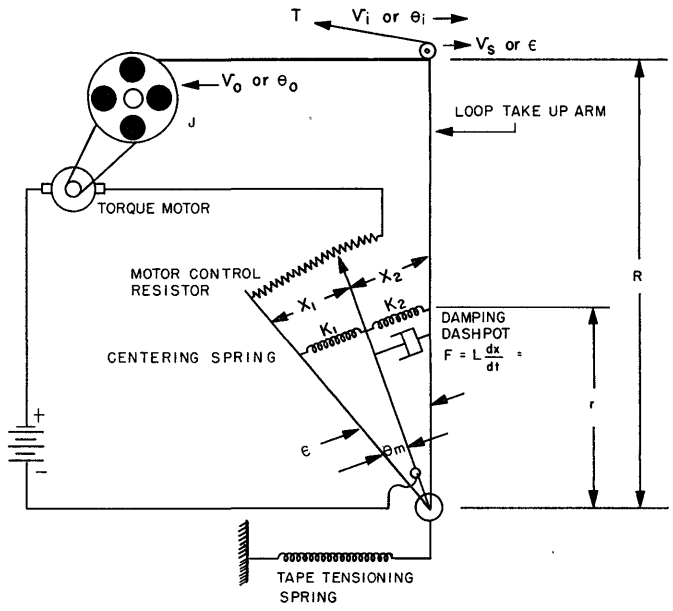


Fig. 5. Unidirectional system

utilization of modern digital computers can be realized. The design of this new tape handler, and future extensions of this design, will materially aid in the attainment of the goal sought.

Appendix

For purposes of analysis a unidirectional system is shown in Fig. 5. The contacts have been replaced by a continuously variable resistor, the static tape tension, friction, and the small component inertias have been neglected.

The following forces are acting on the control resistor slider:

$$K_1X = K_2X_2 + L\frac{dx_2}{dt} \tag{1}$$

$$\Theta_m = \left(\frac{K_2}{K_1+K_2}\right)\epsilon + \left(\frac{L}{K_1+K_2}\right)\frac{d\epsilon}{dt} \tag{2}$$

$$\epsilon - Kr(\Theta_i - \Theta_0) \text{ and } \frac{d\epsilon}{dt} = Kr(W_i - W_0) \tag{3}$$

Substituting equations 3 into equation 2

$$\Theta_m = K(\Theta_i - \Theta_0) + K(W_i - W_0) \tag{4}$$

From equation 4, it is apparent that the motor control angle, and hence motor torque is proportional to both input displacement and rate of change of displacement.

Finally, the dynamic forces acting on the motor and tape system are as follows:

$$J_0\frac{d^2\Theta_0}{dt^2} = K\Theta_m = K\left[(\Theta_i - \Theta_0)K_2 + \left(\frac{d\Theta_i}{dt} - \frac{d\Theta_0}{dt}\right)L\right] \tag{5}$$

From

$$\Theta_0 = \Theta_i - \epsilon$$

$$J_0\frac{d^2\Theta_i}{dt^2} = J_0\frac{d^2\epsilon}{dt^2} + L\frac{d\epsilon}{dt} + K\epsilon \tag{6}$$

Equation 6 is typical of a "closed loop" system with "error rate" damping only. In terms of the error E, the solution gives:

$$\epsilon = \frac{wi}{\sqrt{\frac{K}{J}}}\,\text{SIN}\left(\sqrt{\frac{K}{J_0}}\right)t \tag{7}$$

Or in the critically damped case:

$$\epsilon = w_i t e^{-\frac{L}{2J_0}t} \tag{8}$$

The actual system response may be made to approximate closely either of these cases by adjusting the dashpot damping coefficient L.

Nomenclature

K_1K_2 = centering spring constant
J_0 = Reel and motor inertia
L = Dashpot damping coefficient
$\theta_i\theta_0$ = Input and output tape displacements as referenced to reel radius
ϵ = Take-up arm displacement
θ_m = Motor control resistor angle

Requirements for a Rapid Access Data File

GEORGE EISLER

GENERAL-purpose business-data-processing machines now on the market are limited in their performance mostly by the electronic file systems associated with them. The only file medium that has proved itself acceptable from the standpoint of both cost and speed is magnetic tape. In the quest for better, i.e., faster, cheaper, etc., general-purpose machines, a "random access" file has been held out as perhaps the key to the next step of progress. "Random access," however, is a misleading term if it is used to describe what an ideal file should be, rather than adopted. The considerations which, taken together, point to the requirements for a rapid access data file in a general-purpose data processor will be presented here.

The desirable factors having the greatest influence on system utility are listed as follows:

Speed

High operating speed when large groups of data are to be processed at one time.

Adequate look-up speed when single records are required.

Addressing

Items locatable by the same identifications as in a manual file.

Items locatable by various categories other than normal identifications.

Capacity

Adequate capacity, probably implying both expandability and high utilization of file space.

Interchangeable storage medium.

Nonvolatility

Ability to retain information in the absence of electric power for periods of weeks or months.

Furthermore, these desirable characteristics must be implemented by devices of reasonable cost which, in addition, are within the over-all processor system requirements of reliability, size, weight, and other factors.

The implications of the features listed are considered in detail in the following section; the order, however, in which they are taken up does not necessarily reflect their relative importance since most of them are fairly interrelated.

GEORGE EISLER is with The National Cash Register Company, Hawthorne, Calif.

Speed

One of the most important considerations bearing on the performance of a data-processing system is the speed with which records can be made available by the file for the processing operations.

Data-processing operations calling for some modification of the file can be carried out in two general ways, depending on the particular application. If the time lag is permissible, data for a number of similar operations may be collected and processed at one time as a group; or, if the time lag is not permissible for some reason, individual record modification may be carried out as soon as a new piece of information arrives and the machine is available. The choice of the procedure actually used, however, not only depends on the demands of the processing situation but the characteristics of the file as well.

By way of illustration, the processing of a group of checks which a bank receives from a clearing house may be performed as a single continuous operation at a convenient time in the scheduled daily program. It so happens that with a magnetic tape file in the processor, this is preferable, since it is far faster to process data in a group when both the new and the stored information are in a matched sequence and the tape need be scanned only once. There is no reason, of course, why this operation could not be done by treating the checks separately and in no particular order, if there were a file where any record

might be retrieved in a sufficiently short time. The choice in this case is based on file characteristics.

In the same bank, however, the balance in depositors' accounts must be made available to tellers about to cash large checks over the counter. There is no possibility here of group operation; each inquiry must be answered as it arrives. The example is fairly representative of most applications; that is, the bulk of the processing may be handled in groups with some delay, but a certain amount of immediate action is often necessary.

The more immediate problem in data processor design is to provide reasonable access speed to randomly chosen records. Improvements in the speed of group processing, while certainly to be desired, are not so pressing. On the other hand, the access time requirement alone is relatively modest in the cases where individual record selection is a must.

To illustrate, in a particular bank during an 8-hour workday, there might be something like 1,000 inquiries from both tellers and customers requesting the balance in particular accounts. If further, rather arbitrarily, it is desirable not to devote more than 10 per cent of the processor's time during working hours to the task of retrieving records to answer individual inquiries, and no time sharing is possible, the time allowed within the processor for each inquiry is $\frac{8\times3{,}600\times0.1}{1{,}000}$ =2.88 seconds. While this is a long time in electronic terms, it must be contrasted with the several minutes that is required to scan a single reel of file tape as specified in most electronic data-processing systems. These figures indicate that the current magnetic-tape files cannot be used directly to answer random record requests in the situation cited, and that, in an installation requiring such operation, either a specialized mechanical arrangement must be made or the information summarized in anticipation of the requests.

On the other hand, the performance of magnetic tape file systems while operating on groups of records is another matter. Considering a representative file, for instance, an average of about 30 records a second will be provided to the processor when alternate records are examined in the processing run (records of 15 words' length are assumed here). This is roughly equivalent to an access time of 35 milliseconds per record; when one out of four records are examined, the access time on this basis is 55 milliseconds. Even at a one-out-of-ten processing density, the equivalent access time is 115 milliseconds.

For a typical application of a general-purpose data processor, the situation is roughly as in the foregoing illustration. The speed of file operation for group processing of data is much more important than the requirements for individual record retrieval simply because most of the processing can be done on a group basis, and for the present this is the most efficient way. This will continue to be the case until files are produced which will provide processing speeds at least approximating those achieved by magnetic tapes, as well as providing individual record retrieval at the somewhat more modest speeds called for.

A discussion of file access time must also include consideration of the sorting question. The group processing speed of the magnetic tape file is based on both the file and modifying data being in the same sequence, so that the tape may be scanned in a single, unidirectional pass. Depending on circumstances and the machine, it has been estimated that about 100 milliseconds per record are required for sorting the new information into sequence before the actual file modification. On the other side of the ledger, however, many business procedures require data in alphabetical or numerical order so that very often storage in sequence is desired apart from file operating needs. In particular where a sorted output such as a stock list is required many times from, essentially, the same file data, sorting at each readout is a wasteful procedure; in these cases, an ordered file storage would be much more efficient. This being the case, it seems that the sorting time should not be considered chargeable only to a magnetic tape file, or for that matter, to any pure scanning-type system. This, especially, since those files which use at least a partial scanning scheme for record location also benefit from ordered record storage.

In assessing the prospects of faster access to records, consider first a scanning-type file. There appear to be two possibilities for increased speed:

1. Scan faster.
2. Scan only the parts of the records which are significant for identification.

However, along with the second possibility, the ability to scan any selected segment of records is desirable in order to maintain the file operating efficiency where, for instance, all items belonging to a given category must be selected.

There are, however, very definite limits to the speed with which scanning can be accomplished. Assuming, strictly for the sake of illustration, that the file contains 10^8 binary digits and that a maximum of one second is allowed for each randomly chosen access, the binary digit rate must be 100 megacycles when the worst case is considered. By way of comparison, digital circuitry is made to work only with great difficulty and expense even at a 10-megacycle rate, so that a completely sequentially scanned file cannot be expected to function very fast for individual location of items unless new techniques in electronic circuitry are developed concurrently.

Scanning only the address portions of records is more promising, but still not clearly feasible since the assumption of file capacity was not realistic but on the low side.

In a discrete (or multiple) entry type of file, the access time to a particular record is reduced by entering the file directly at, or at least near, the desired record's location. The additional requirement for doing this is the knowledge of the record's location in file, so that the proper entry point can be chosen. The greater the number of entry points, the faster the access, but also greater is the amount of information required to choose the proper entry point in the first place.

Addressing

The procedure used in finding records in the data file of a business processor also has great influence on the ultimate performance of the system. The generally desirable features are next considered. These are:

1. A great part of the data handled in the course of business is now filed by classification according to names, stock symbols, nonconsecutive account numbers, etc., and it is greatly desirable that the processor data file be designed to locate information by using these commonly accepted appellations.
2. The amount of information to be filed under a particular identification symbol (often called the "natural name") varies greatly, as, for example, the number of transactions in a particular bank account in a given month; the file should be able, therefore, to handle efficiently records of widely varying lengths.
3. In addition to the ability to select records by natural name, there should also be provision for the selection of file data by types or categories, as would be useful, for instance, in searching the accounts receivable file for overdue accounts.

To make available the desirable features listed, suppose that the approach taken in the main store was utilized and a discrete space in the data file allotted to each possible record identification, i.e., to each possible combination of letters in

names, each possible permutation of a stock number, etc. The natural name would then be used directly by the processor as a file address. The result would be a requirement for a file of huge capacity, of which only a relatively small part would be used at any one time, since there are a great number of combinations of letters which are not used by anyone for names; and of those that are, not all are customers of a particular company. About the same situation arises with regard to stock numbers, car registration numbers, and other commonly used identifications, most of which have large amounts of redundancy inherent in them. From the machine design standpoint, at least the file could of course be greatly simplified if business concerns would adopt consecutive numbering systems in their operation. The difficulties encountered in changing to consecutive identification schemes undoubtedly varies greatly from one firm to another. Its advantages for machine processing should, however, be weighed for each application, so that in the future more efficient use of file storage space may be made. In the meanwhile, general-purpose machine design based on the continued use of redundant natural names seems necessary.

In order then to reduce the amount of wasted space in the data file, it is necessary to be able to assign empty locations to records as the particular situation demands. That is, the available file spaces should be filled by the automatic action of the processor as the need arises. The two approaches to this are: (*a*) to store records, in or out of ordered sequence, without reference to location and then scan the file to recover a particular one, or (*b*) to assign a specific file space to the given record, remember this assignment, and use it to locate the record when desired.

Considering the scanning system first, it is noted that the records must contain as part of the entry their own identification symbols, and that a specific record is found by examining, one after another, all of the records on file. However, as previously discussed, with present-day mechanizations, this type of file tends to be slow, particularly when only a single record is to be found. In a group processing operation, the speed is improved by a large margin, but still remains the factor limiting over-all operating performance.

A scanning system handles records of variable size naturally, there being no particular need to have the records be of predetermined, or of equal, length. On the other hand, the closing out of records necessitates the eventual rewriting of the file in a densifying operation, in order to preserve a high space utilization efficiency. Whether rewriting is necessary when new records are to be added depends on whether the records are filed in sequence or not; but, since it is mostly likely that sequential ordering would be used in any completely scanned file, complete file rewriting must be considered a part of the scanning-type file operations. This, of course, is the description of a magnetic tape file. To find ways to overcome some of its shortcomings in the first place, it is best to consider some alternatives.

Selective scanning, that is, scanning only portions of records, as a means of speeding up file access is atractive; the disadvantage here is that most selective scan procedures require that information words be spaced at certain regular intervals so that, for example, every tenth word may be examined for content. With records of varying lengths, this results in wasted space, which can be minimized by the proper choice of intervals, but not completely eliminated. Selection by category is also made correspondingly harder.

Considering a pure discrete entry system next, it is seen that while the access time to a record might be very small the choice of entry point implies prior knowledge of the record location. This prior knowledge can only be stored in a additional memory system if, at the same time, the use of natural names and a reasonable storage utilization is to be retained.

This second memory system, usually referred to as an index, is by no means a small auxiliary. It has the job of picking out a file location (out of possibly 10^5 choices) that contains the particular record having the input natural name. This input natural name itself can be one configuration out of an astronomically large number of possibilities representing the permutations of possibly ten alphanumeric characters. Making provisions for record selection by category complicates the index even further. The magnitude of this indexing operation is such as probably to make the pure discrete entry file impractical.

Between the pure scanning system, which is found to be slow, and the pure discrete entry, which requires too much equipment, might lie a combination system which would provide a suitable compromise. In general, this type of file would be scanned for actual record location, but the scanning would begin at any one of a number of entry points. Minimum average access time would be obtained when approximately equal amounts of information are stored between access points, so that the amount of scanning would be minimized. The indexing mechanism in this arrangement would choose an entry point for a given natural name instead of a specific location, thereby reducing the number of possibilities from which the choice has to be made. Within this framework, a number of alternatives are possible, allowing varying amounts of flexibility to the file system.

One possibility is to divide the entire pertinent range of natural names into intervals corresponding to intervals between file entry points. The actual entry point selection would be done by a matrix which would use little time for operation but would require resoldering, or at best, reconnecting on a plug board, for a change in the interval. An alternate approach would allow the computer to rearrange the intervals, changing the indexing unit appropriately, thus eliminating the waste of memory space in the fixed interval approach because it is necessary to allow spare memory capacity between intervals to account for uneven interval loading. Additional flexibility, however, requires that more and lengthier file housekeeping steps be performed.

It is difficult to classify and examine systematically the various modes of indexing possible; it can be seen, however, that the index, active or passive, is essentially a storage device, and is subject to much the same organizational problems as the data file proper. It is likely then that a successful mechanization will also represent some compromise within the cost-capacity-speed triangle.

Capacity

A major distinguishing feature of business problems is the vast amount of file data that must be handled. A further distinguishing feature seems to be that the file data are growing vaster all the time. Inquiry into the needs of potential users of business data processors indicates that a capacity of 10^8 characters seems to suit the majority of applications. The value of 10^8 characters is then chosen as a reference figure, with the understanding that it is certain that some applications will require a much larger capacity. This additional capacity might be supplied by duplicate equipments, or, if the design and processing flow permit, by additional storage media which are detachable from the file mechanism.

The quoted reference figure, however, represents only the number of characters of information to be filed by the process-

ing system. To arrive at the actual capacity required in a particular file, the amount of wasted space inherent in the design must also be accounted for. As noted previously, unused file space may result from the deletion of records or from the erection of discrete intervals into which the information does not fit precisely. This latter condition arises when the design includes fixed intervals between entry points, fixed record lengths, or, for that matter, fixed word lengths. The inefficiency arising from these factors is unavoidable to a certain degree, and in any case may be preferable to increased access time or additional housekeeping operations, but must be accounted for with additional capacity nevertheless.

High utilization of storage capacity may also be enhanced by designing the file so that it may expand as necessary. As an example, when a magnetic tape file is completely filled, only a new reel of tape is required for expansion. The cost of additional capacity is then only the cost of the medium itself, i.e., the tape, and the additional processing and access time that is now required. This expandability is a particularly valuable characteristic of a general-purpose data file whose future applications cannot all be foreseen in advance. It is especially useful where this feature can be used to insert new records between already existing ones in a sequentially ordered file without requiring that at least the remainder of the store be rewritten to make room for the new insert. Admittedly, file expandability in this sense would not be practical in many of the high-speed file systems currently conceived; on the other hand, it might be possible to achieve at least some expansion if its advantages are clearly recognized and kept in mind.

Connected with file expansion, the ability to interchange the storage media used in a file system contributes to its flexibility. As the total contents of files get larger, there is usually less need to be able to refer to the entire file at one time. Considerable equipment duplication can be avoided in these cases if the storage medium itself is detachable from the associated hardware.

Nonvolatility

A most important requirement for any business data file is that it be able to retain accurately the information stored in it for long periods of time. In general, this means that the stored information should remain unaltered by lack of electric power, preferably permanently, but at least for periods of weeks or months. The need for essentially permanent machine-accessable storage arises from the relatively slow operation of input-output equipment; at the current input-output speeds, it is not feasible to store the data file on more durable paper records, to be read into the processor whenever required. Permanent storage is, of course, the most desirable; however, storage durations of weeks or months are probably acceptable at the additional penalty of requiring complete file restoration at appropriate intervals of time. In any case, the longer the available safe storage period, the more suitable the medium.

General Requirements

It might be pointed out that the techniques to meet the requirements listed in the previous sections are all presently available individually, but their use together results in files of unacceptably high costs. In a sense then, the problem can be considered as one of searching for new means to accomplish the job at reasonable prices, at the same time maintaining the high degree of reliability necessary for any successful processing operation. In this same sense, lower production costs, increased freedom from environmental changes, lower power requirements, etc., can all be considered as part of the goals of new data file development programs.

Engineering Design of a Magnetic-Disk Random-Access Memory

T. NOYES W. E. DICKINSON

THE International Business Machines magnetic-disk random-access memory is a large-capacity storage device with relatively rapid access to any record. Fig. 1 shows the unit.

The information is stored, magnetically, on 50 rotating disks. These disks are mounted, so as to rotate about a vertical axis, with spacing between disks of 0.3 inch. This spacing permits magnetic heads to be positioned to any of the 100 concentric tracks which are available on each side of each disk. Each of these tracks contains 500 alphanumeric characters. Thus, the total storage capacity is 5,000,000 characters.

The reading and writing is accomplished with two magnetic-recording heads. These heads are mounted in a pair of arms which can be moved in a radial direction to straddle a selected disk. The arms are positioned to the selected disk and then moved into the desired track by means of a feedback-control system. A unique arrangement permits one set of magnetic-powder clutches to provide the drive force for both positioning tasks.

The time to position the heads from one track to another depends upon the distance separating the tracks. The average access time, however, is about 0.5 second.

Disks were chosen for the recording surface because they have a good volumetric efficiency for surface storage. In addition, they permit multiple access possibilities to any record. Magnetic recording was chosen because of both its permanence and its ease of modification while still allowing good storage density.

Disk Array

The 50 disks, which make up the disk array, are the key to the layout and to the size of the memory device. A vertical shaft was chosen as this more readily permits multiple access mecha-

T. NOYES and W. E. DICKINSON are with the International Business Machines Corporation, San Jose, Calif.

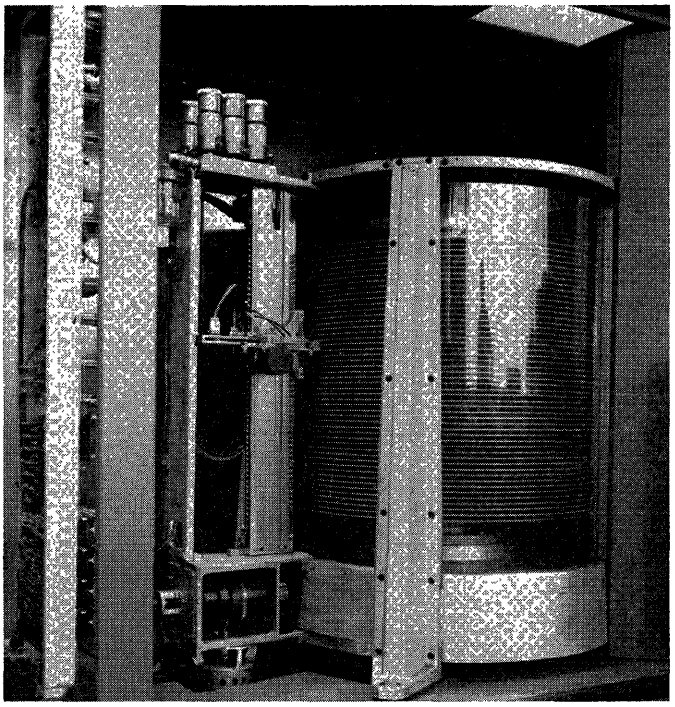


Fig. 1. Random access memory unit

nisms to be used. Theoretically, 20 access mechanisms could be installed. Up to the present time no more than three access mechanisms have been used. Mounting pads are machined on the upper and lower end castings of the array so that the access mechanisms may be bolted in place and be easily removed for periodic servicing.

The disk shaft runs at 1,200 rpm on precision tapered roller bearings mounted on a heavy stationary vertical spindle or axle. The radial runout limitations are less severe than with a magnetic drum because of the read-write design so that 0.001-inch runout is acceptable. The driving power is supplied by a conventional capacitor-start induction-run motor. The large inertia of the disks requires a $1^1/_2$-horsepower motor to accelerate them to running speed in less than a minute but the running power is less than a horsepower. Voltage variations of ± 10 per cent give less than 1 per cent variation in speed.

The iron-oxide-coated aluminum disk is 24 inches in diameter and 0.1 inch thick. The thickness is required to maintain flatness during assembly and dynamic stability in use. The flatness of the disk is indicated by the requirement that there be no more than 0.0015 inch out-of-flatness in any 2-inch distance on the surface. However, as much as 0.030-inch runout can be tolerated in total axial runout. The uniformity of the magnetic coating is indicated by the tolerance of ±10 per cent on the 60-millivolt peak-to-peak signal at the outside track.

The disk-to-disk spacing is established by the disk thickness and the clearance required for the magnetic heads to go between them. For practical reasons, on this machine, the spacing was set at 0.3 inch space between 0.10-inch-thick disks. Thus, the over-all height of the 50 disks is 20 inches.

A modified non-return-to-zero type of magnetic recording is used. The density of the recording varies inversely with the radius of the track. The density on the inside track is about 100 bits to the inch while on the outside track this drops to about 55 bits to the inch. (The outer-track radius to the inner-track was chosen as near to the optimum ratio of two as was mechanically feasible.) This recording density permits 500 characters composed of eight bits each, to be recorded on each track. With 100 tracks on each of the 100 sides, a total of 5,000,000 characters can be stored. An idea of the quantity of the storage can be conveyed by a few comparisons. This amount of storage is about the same as 60,000 80-column punched cards, or 2,000 feet of magnetic-recording tape recorded at 200 characters to the inch with no space between records or 940 single-spaced typewritten pages. An equivalent storage capacity on a magnetic drum would require a drum 13 inches in diameter and 42 feet long. Such a drum would have about seven times the volume of the disk array.

Magnetic Heads

Closely associated with the disks are the magnetic heads used to record on them. Unlike drum heads, these heads must be of minimum height. Refinements in design and techniques have brought this height down to 0.2 inch. Binocular microscopes must be used in the manufacture of these assemblies which are potted in an epoxy resin after assembly to give durability and shock resistance.

The magnetic element consists of two distinct magnetic circuits. One circuit with its coil erases only, and the other circuit reads and writes. The erase gap erases a wider track than the following write gap records. This design allows reduced precision in radially positioning the heads because there are no magnetically disturbed track edges to contribute noise to the newly recorded track which might not precisely coincide with the track previously written by a different access mechanism.

The spacing of the heads from the disk is maintained by an air bearing obtained from minute air jets in an annular manifold surrounding the magnetic elements. The 0.001-inch spacing is held despite the axial runout in the disk so that there is never physical contact between the heads and the magnetic coating. The head is horizontally constrained in a gimbal socket in the arm.

The use of compressed air in the magnetic heads, and the access positioning detents, requires a small compressor. This unit operates constantly, supplying air to a surge tank or by-passing it to the atmosphere as necessary. Approximately 0.6 cubic feet per minute raised to 50 pounds per square inch is used per access.

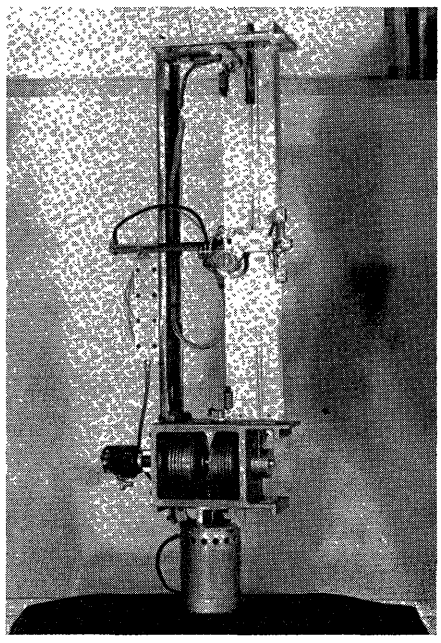


Fig. 2. Access mechanism assembly

Access Mechanism

The access mechanism shown in Fig. 2 is used to position the pair of heads to any track on the disk array. The heads face each other in a pair of arms. The arms move inward to straddle a selected disk when reading or writing. The arms, in turn, are carried on carriage for vertical motion to the desired disk.

The arms are guided, for radial motion, in bearings on the carriage. Within these

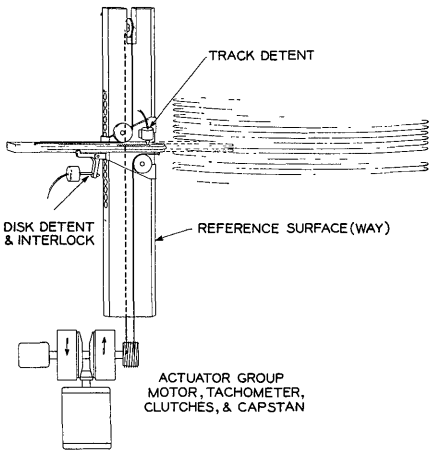


Fig. 3. Mechanical functional schematic

bearings the arms are capable of about 6 inches of radial motion. The inner 5 inches position the heads over the disk recording area. When the arms are in their outermost position, the arms are completely outside the disks. This is the position the arms are in during vertical-drive motion.

The carriage, during vertical motion, slides on a vertical "way." See Fig. 3, a functional schematic. At each of the 50 disk positions a detent hole is provided in the way. A pneumatic-detent piston is energized upon arrival at the desired disk. This detent, by means of a mechanical linkage, controls an interlock which frees the carriage and locks the arms for vertical drive and frees the arms and locks the carriage for radial drive. The arms are capable of being freed only when the carriage is positioned properly at a disk and the carriage is capable of being freed only when the arms are completely outside of the disks. Thus, a safe interlock is provided to prevent mechanical damage to the disk array.

The driving force is provided by a pair of magnetic-powder, motor-driven counter-rotating clutches. These clutches have a common output shaft on which is located a drive capstan. A small, steel cable connects the drive capstan to the arms through a system of three pulleys. When the arms are locked, the carriage is free and the clutch torques result in vertical-drive motion. Similarly, when the carriage is detented the arms are free and the same clutches control radial motion. In addition to the detent for locking the carriage, a detent is provided to position the arms accurately to the selected track. These detents greatly relieve the positioning requirements of the position-feedback controller.

The clutches are controlled by a null-seeking feedback-control system. Position signals for the radial and the vertical drives are obtained from potentiometers on the carriage and on the way, respectively. Fig. 4 shows a functional schematic which is intended to illustrate either radial or vertical positioning. An electrically floating voltage supply feeds the potentiometer element. Taps are located uniformly along the potentiometer element. The desired address is established by grounding one of these taps with an address-relay tree. An error voltage is seen by the potentiometer wiper unless the wiper is positioned at the selected

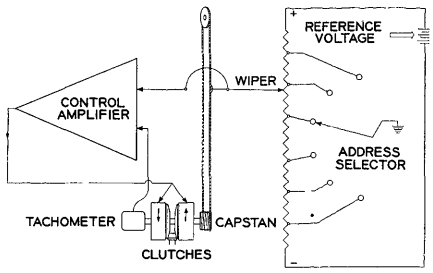


Fig. 4. Electrical functional schematic

tap. Through the control amplifier the clutches are controlled to position the wiper so as to make the error voltage zero. The tachometer is used to stabilize the feedback loop. A d-c control system is used with relatively large voltage on the potentiometer to eliminate any serious drift problem.

A group of relays are used in a logic network to guide the access mechanism in positioning. The logical decisions are based primarily upon the condition of the carriage detent, whether the carriage is at the correct disk and whether the arms are at the correct track.

Read and Write Amplifier

The read and write amplifier circuitry is quite conventional. The only variation from normal is the inclusion of an automatic-gain control in the read amplifier. This feature is included to correct for input variations caused by variations in heads, disks, and surface speeds.

Self-Clocking System

To relieve the accuracy required in positioning the head along the track, a self-clocking system has been incorporated. The tolerance with this system is such that limited movement of the head along the track can occur while reading or writing. Reading or writing with different access mechanisms is also facilitated by this system.

The clocking system is composed of two oscillators. Their operation is such that one oscillator is on while the other is off. In writing, one oscillator runs continuously for timing the entire record. In reading, the oscillators are controlled by the recorded bits. Each bit read switches one oscillator off and the other on, thus resynchronizing on each bit. When an oscillator is turned on it always comes on with a certain phase relationship. The combined output of the two oscillators feeds the bit ring used to determine bit position within the character.

In a system such as this the frequency of the oscillators relative to the disk speed is important. With the present arrangement a tolerance in excess of 1 per cent is permissible. This tolerance is easily met with the large inertia of the disk array, the motor voltage-speed characteristics, and the oscillator stability.

Print I—A Proposed System for the IBM Type 705

R. W. BEMER

Purpose of the Print I System

THE PRINT I (PRe-edited INTerpretive) system has been designed to meet the engineering and scientific computing needs of those Type *705* installations where such work is a secondary computing requirement. It is specifically tailored for this purpose and in general has no similarities to Type *705* systems designed for business and commercial applications. Although provision is made to move freely from abstraction to *705* commands and back, it is not recommended that PRINT I be tied to any business system for the reason that the restrictions of one type of usage will too often hamper the other.

The basic consideration in the planning of this system has been ease of learning and continued operation by personnel with either very modest programming experience or none at all. Pseudo-instructions are simple and straightforward; basic principles may be assimilated quickly. There are no restrictions in the use or combination of pseudo-instructions, although minor increases in operating speeds have been introduced to effect this. Basic logical errors in the written program will be detected automatically and detailed analysis of such errors will be typed out to the operator.

Choice of System Characteristics

Components

Since some Type *705* installations have ordered configurations which do not include a magnetic drum, the only components specified for this system are magnetic core memory and sufficient magnetic tape units to handle expected problem size.

Abstraction Type

The first decision to be made was between the compiling and interpretive modes. The fastest method of computer operation is with a compilation of basic machine commands operating in linear progression without modification. The first drawback to this method is that there are no computers in production with the nearly infinite memory required for this long thin line of instructions. The second is that the time advantage gained is not worth enough to compensate for cost of additional memory if it were available. A more practical variation of this is the type of compiler which still forms machine commands but uses modifying commands to reduce the volume of instructions, just as a conscientious programmer would. The serious drawback to this method is that it requires that extreme complexity be built into the compiler in order for it to function as efficiently as the clever programmer. In addition, the program produced is often much larger than that produced by a compact interpretive system.

An interpretive mode offers quick construction and minimum storage, obtained by the complicated weaving and interlocking that the coder may accomplish. Its drawback, an extremely serious one, is that the command is normally re-interpreted every time it comes up for execution in actual operation. Thus the variable "fetch" and "operate" commands are fabricated a multiplicity of times, whereas a compiler fabricates them only once.

This has led to rather determined stands by the opposing compiling and interpreting adherents. Fortunately there is a compromise available, as demonstrated in the PRINT I system. A pre-editing routine performs a compiling and assembly function to make the commands numerically palatable to the executive routine which is in the interpretive mode. In addition, a repeat command is furnished in the list of pseudo-instructions. This enables certain commands to be executed n times in succession, once with interpretation and command fabrication, and $n-1$ times in the exact fashion that an ultra-efficient compiler would generate the program. Multiaddress commands are used, for although single-address commands are more efficient when the desired characteristics are inherent in the machine language command, multiaddress commands are more efficient in an interpretive abstraction, just as buying more groceries at a time minimizes the cost of trips to the store.

Arithmetic and Format

Floating decimal arithmetic was chosen as most generally acceptable to all scientific users. Since the specification of a mantissa length must, to our way of thinking, specify a corresponding set of subroutines to that accuracy, PRINT I will be packaged initially in 8- and 10-digit versions. A 20-digit version is to be produced after completion of the 8- and 10-digit systems. If enough demand should exist, a 5-digit version may well be produced. Each system will be complete in itself for all operation. There will be complete freedom of interchange between the floating point abstraction and *705* language.

For faster internal operation and convenience, floating point numbers are stored internally as $xxx\ldots\ldots xx\overset{\pm}{P}\overset{\pm}{P}$. The x's represent a mantissa which is a proper decimal fraction with a non-zero leading digit. PP is the power of 10 multiplying this fractional number. Externally, the format is $\pm\ \overset{\pm}{P}\overset{\pm}{P}\pm xxx\ldots$, so that trailing zeros need not be written by the programmer; the pre-edit routine will take care of this automatically.

As far as possible, all arithmetic operations and subroutines will be performed with rounding. A legitimate zero in this system has both power and mantissa equal to zero. To facilitate and accelerate operation with zero operands, the mantissas will be tested for zero in all arithmetic operations. Messages are provided to be typed out in case of error during operation, such as:

Division by zero.
Square root of a negative number.
Power overflow (exceeding +99).
Logarithm of zero or a negative number.
Sine or cosine of angle greater than a prescribed limit.

This list of error messages will be completed as the various elements of the system are finished.

All addressing of operands and locations of pseudo-instructions for this system are in regional form, consisting of one leading alphabetic character and, normally, three numeric characters. Inserts may be made by an additional numeric character on the right, as *G*125 and *G*1251.

Tracing and Diagnostic Routine

There will be a single type of diagnostic routine associated with this system, since the programmer will want to see

R. W. BEMER is with the International Business Machines Corporation, New York, N Y.

results corresponding to his pseudo-instructions, and since compound indexing introduces time-wise complications, a trace routine of an insert transfer to trace control type will be used. This routine will allow high-speed operation to various points of interest, at which time tracing of a specified nature will occur. This may mean tracing of every pseudo-instruction, stores only, transfers only, or any combination of these. This will be under the control of alteration switches. When under trace control, tracing will proceed indiscriminately through both PRINT I and *705* commands.

Interpretive Section

PRINT I will always be in core memory during operation of a program prepared for this system. Work done so far indicates that the arithmetic operations and standard subroutines should be contained within 4,000 character positions. Provision is made for a floating subroutine to be called from tape storage as required. This is described in detail in the section on the pre-edit routine.

The interpretive routine does not use check indicators in any way as decision elements. They are reserved for stops while operating in *705* language, and switches may therefore be set to automatic stop during the operation of PRINT I. Any entry to PRINT I sets up the auxiliary storage units as required for its operation; therefore, all auxiliary storage units are available while operating in *705* language.

Indexing

A system of 4-dimensional indexing is simulated within PRINT I. This indexing is incremental and transfer is dependent upon the contents of the registers exceeding a specified limit. There are three index registers, referred to as *R*1, *R*2, and *R*3. They are four digits in length and the contents may be either positive or negative. Since these contents are used only for address modification, they are unsigned, and negative quantities are stored as 40,000 complements. Any regional address may be augmented by the contents of any of these registers or the arithmetic sum of any two or all three. This alteration takes place in a fixed memory position before fabricating the necessary machine language commands from the address portions of the pseudo-instructions. The original command in operating sequence is never altered. Loops formed by transfer on index commands will therefore require reindexing from the original addresses. Proper use of indexing features will greatly lower the number of PRINT commands in any program. It is for this reason that the interpretive routine of up to 4,000 characters can be tolerated as a permanent part of core memory.

System Entry

PRINT I will be loaded into core memory from a deck of cards furnished for a specific mantissa length. Tape unit *1* is normally assigned to contain, for purposes of re-entry, the PRINT system after initial loading. It will also contain the pre-edit routine, nonstandard subroutines and both the original and edited progams. On entry of the system from either cards or tape, a general check will be performed to indicate proper loading. If errors occur, typed-out alarms will indicate their nature; a second loading may in some instances be entirely satisfactory.

Input and Output

Two types of input are required, one for instructions to the pre-edit routine and the other for data necessary for execution. Data input routines of three types may be provided, as follows:

Serial loading, based on a single specified address per card.
Random loading, address and contents specified for each word.
Special 3- or 4-digit volume data loading, a single floating power being assigned to all input and scaled in the program.

Output to printer will be by type-wheel image in memory, consisting of several words appropriately spaced for printing. These positions will be loaded by pseudo-store-for-print commands specifying data position number. Printing will be of the write-erase 00 type for flexible format. Spacing will be under program control with write-erase 01 specifying a group mark in the highest memory position.

Punching, when necessary, will be accomplished by a similar method, utilizing a card image position in memory.

Pre-edit Routine

The programmer will normally code mnemonic commands which specify, in a variable field controlled by commas, the regional addresses and indexing. A typical command is:

SUB F123, 1, F286, 2, F174, 1+3

These commands and associated comments are punched one per card for pre-edit and assembly. Card columns are punched consecutively, the first blank column indicating the end of the command and the start of the comment field. For each mnemonic command, the pre-edit routine produces a corresponding numeric command especially tailored for the fabrication of machine commands from its components. This numeric command will be of varying length according to the operation specified. Matched sets of mnemonic and numeric commands, together with the comments, may be printed for inspection at pre-edit time at the option of the operator. If the program is known to be correct, the edited form may also be punched out on cards for later re-entry in more condensed form, also avoiding repetition of the pre-edit process. Both of these options are under alteration switch control. For convenience in correcting programs, both the original mnemonic program and the edited routine will be stored on the library tape. Temporary patching of the program, by transfer-out and return, may be made by reading in command cards to replace those in error (this group may contain inserts). After the program is running correctly, these patch cards may be inserted in their proper sequence in the program and the entire program re-edited properly. Tracing will be printed out in conformity with the pseudo-instructions as far as possible, with respect to the contents of the various addresses and index registers.

A regional address notation was chosen to insure early completion of this system. Symbolic addresses of the type where the address is either descriptive of the contents or literally the contents themselves would greatly complicate the pre-edit routine. Complete freedom of interchange between *705* commands and PRINT I commands is automatically assured by the pre-edit routine. Upon discovery of variance from, or return to, the 5-digit *705* command, the pre-edit inserts links and transfers as required. For this reason, any address in a non-alphabetic region is a fixed *705* character address. The *Z*-region is reserved for PRINT I.

Two types of regional addresses are recognized by the pre-edit routine. The first is for data, where the address represents a fixed number of characters according to the mantissa length of the system used. Data should normally be allocated by the programmer to the highest portion of memory. This is done by specifying to the pre-edit routine that a certain region starts at a specific character address, the extent and positioning of the region being dependent on (number of addresses in region) times (word length). The basic address of a data word is that of its highest memory position. The pre-

edit allots memory in $(m+2)$ modules, where m is the mantissa length, and computes the basic address from the numeric portion of the address.

The second type of regional address is for the pseudo-instructions, which are of various lengths. They are normally allocated to the portion of memory just above the PRINT I system, and are normally obeyed in ascending order. The basic address of an instruction is that of its lowest memory position and equals (the basic address of the previous instruction)+(length of previous instruction). In this case, the pre-edit must construct a table to correlate the regional address of each instruction with its basic address. If the program should start to infringe upon memory allocated to data, the pre-edit types an alarm indicating the overlap position. Automatic treatment of this situation is not provided for; responsibility for the avoidance of or correction of such a condition is assigned to the programmer.

The programmer is not required to write redundant information in the variable field of the pseudo-instruction. The pre-edit automatically inserts the proper information as described under the command summary.

Certain alarms are set up to be typed out during pre-edit if the programmer has ignored the very few restrictions inherent in the PRINT system. A partial list of alarms contains:

Program running into data storage.
Calling for a pseudo-instruction as data, when not in PRINT, with the exception of the commands replacing first address.
Nonrepeatable command following a repeat command.
Nonalphabetic address ending other than 4 or 9.
Minus index limit on a clear index and set limit command, or converted limit greater than (memory capacity—10,000).

When a floating subroutine is coded, the pre-edit knows that such a mnemonic symbol has no assigned operation code in the table. It is assigned the floating subroutine operation code and the pre-edit automatically sets up *705* instructions to bring the proper subroutine (if it exists on the library tape) into the floating position in PRINT I. Such a call-out is set up on change of requirement only. Depending on detailed study of the various factors in frequency-time-storage balance, it may be desirable to set up two such floating subroutine positions. Another possibility is for the programmer to specify to the pre-edit routine the amount of memory he is willing to allocate to floating subroutines. Replacement would then be set up only if the desired subroutine exceeds in size the amount of available memory left.

Operation Summary

General

Operation codes are limited to 40 in number. This does not necessarily limit the number of operations to 40, for a single operation code may be tagged in the body of the command. Twenty plus codes, from 04(5)99 are allotted for nonindexable operations. Twenty minus codes, from $-99(5)-04$ are allotted for indexable operations. Proposed codes and corresponding mnemonic symbols are:

Nonindexable			Indexable		
04	CS1	54	$\overline{04}$	TMT, TAB	$\overline{54}$ SQR
09	CS2	59	$\overline{09}$	ADD	$\overline{59}$ SIN
14	CS3		$\overline{14}$	SUB	$\overline{64}$ COS
19	TRZ, TNZ, TR+, TR−		$\overline{19}$	MPY	$\overline{69}$ ART
24	TRU		$\overline{24}$	−MY	$\overline{74}$ LGD, LGE
29	RPT	79	$\overline{29}$	DIV	$\overline{79}$ EXD, EXE
34	RWR	84	$\overline{34}$	−DV	$\overline{84}$ TRC
39	TX1, TN1	89	$\overline{39}$	MAD	$\overline{89}$ TRE
44	TX2, TN2	94	$\overline{44}$	PMA	$\overline{94}$
49	TX3, TN3	99	$\overline{49}$	FSR	$\overline{99}$

Other operations (such as ELO, NOR, various store-for-printing operations, and placing the TRx tally in an index register) are not fixed at this time and are not shown in the table. If desired, the user may insert his own subroutines by placing the proper entry in the operation switch position; this may involve replacement of some of the standard subroutines.

This manual will contain an appendix with enough detailed information to enable the user to make these and other alterations with relative ease.

Interpretation and indexing times vary with the indexing requirements as shown in table (time given in milliseconds):

Type	No Indexing	1 field	2 fields	3 fields
Test first for any	0.816	1.326	1.581	1.836
Test each in order	0.952	1.207	1.462	1.717

One of these two types will be chosen on the basis of lesser elapsed time in actual operations. Nonindexable commands require 0.697 millisecond (ms) for interpretation.

α, β, and γ may be any regional address, including α, β, and γ or the pseudo-accumulator PACC, which is a field in memory reserved for that function. PACC is not indexable, and a zero indicator is automatically inserted for it by the pre-edit routine. Indexable commands are usually written with the index register indications for each address following the address itself, separated by commas, as:

SUB G258, 2, G184, 2+3, G702, 1+3

If the result address is omitted in writing the pseudocommand, the pre-edit routine assumes that it is to be PACC. If no index information is written, indexing is denied by setting the tag to zero.

Indexable Operations

Class I-1, Arithmetic Operations. There are eight operations in this class. The operations are:

Written	Signifying
ADD $\alpha, R\alpha, \beta, R\beta, \gamma, R\gamma$	Place $(\alpha)+(\beta)$ in γ, PACC
SUB	Place $(\alpha)-(\beta)$ in γ, PACC
MPY	Place $(\alpha)(\beta)$ in γ, PACC
−MY	Place $-(\alpha)(\beta)$ in γ, PACC
DIV	Place $(\alpha)\div(\beta)$ in γ, PACC
−DV	Place $-(\alpha)\div(\beta)$ in γ, PACC
MAD	Place $(\alpha)(\beta)+(\text{PACC})$ in γ, PACC
PMA	Place $(\alpha)(\text{PACC})+(\beta)$ in γ, PACC

Two mnemonic symbols may require explanation; MAD stands for multiply-add and PMA stands for polynomial multiply-add, so-called because of its suitability to polynomial evaluation with α as the address of the argument. These two operations are also exceptions in that when modified by a RPT command the result is not stored in γ until the last repetition is complete.

Class I-2, Subroutine Operations. There are nine operations in this class. One of these is a floating subroutine which is at any time the function assigned to it by the pre-edit routine. At present, all

transcendental functions refer to radian arguments. The operations are:

Written		Signifying	
SQR	$\alpha, R\alpha, \beta, R\beta$	Place $\sqrt{(\alpha)}$	in β
SIN		Place sine (α)	in β
COS		Place cosine (α)	in β
ART		Place arctangent (α)	in β
LGD		Place $\log_{10}(\alpha)$	in β
LGE		Place natural log (α)	in β
EXD		Place $10^{(\alpha)}$	in β
EXE		Place $e^{(\alpha)}$	in β
FSR		Place required function of (α)	in β

Class I-3, Transmittal Operations. There are two operations in this class. Both place the contents of α in address β. When used with an RPT command, they will effect block transfer in a direct or inverse manner, or open up a series so that interpolated values may be interspersed. For the TAB (transmit absolute) operation, the number is guaranteed to be positive in the β position, remaining in the original form in α. Unless PACC is specified as the β address, it will be unaffected by either of these operations, which are:

Written		Signifying
TMT	$\alpha, R\alpha, \beta, R\beta$	Place (α) in address β
TAB	$\alpha, R\alpha, \beta\ R\beta$	Place the absolute value of (α) in address β

Class I-4, Comparison Transfer Operations. There are two operations in this class. They are:

Written		Signifying
TRC	$\alpha, \beta, R\beta, \gamma, R\gamma$	Transfer to command in α if (β) are greater than or equal to (γ)
TRE	$\alpha, \beta, R\beta, \gamma, R\gamma$	Transfer to command in α if $(\beta) = (\gamma)$

When either of these commands is modified by an RPT, a special addressable register (4-digit) is reset to zero and a 1 is tallied in it every time the transfer condition is not met. γ is normally not indexed by the RPT and is the item to which successive β's are compared. This allows for block table look-up and later modification to a finer interval of search. This may be done by *705* language modification of both the RPT and the TRx command to alter both the starting address and the search interval. A preferable method is to use a new pair of commands with the RPT already set to the finer interval and the TRx command indexed by a function of the tally. For such TLU operations the n of the RPT command is normally set to 99.

Class I-5, Floating Point Operations. There are two operations in this class. They will be used to convert fixed point numbers to floating point form and vice versa. Exact characteristics have not been specified yet. The operations are:

Written		Signifying
FLO	$\alpha, R\alpha, \beta, R\beta$	Convert the fixed point number in α to floating point form and place in β
NOR	$\alpha, R\alpha, \beta, R\beta$	Convert the floating point number in α to fixed point form and place in β

NONINDEXABLE OPERATIONS

Class N-1, Transfer Operations. There are five operations in this class. Although they in some respects duplicate *705* commands, they are nevertheless included in PRINT for convenience. They operate in less time than it would take to leave to *705* language, test, transfer, and re-enter PRINT. There is also a distinct saving in memory positions. The operations are:

Written		Signifying
TRZ	α, β	Transfer to command in α if $(\beta) = 0$
TNZ	α, β	Transfer to command in α if $(\beta) \neq 0$
TR+	α, β	Transfer to command in α if (β) are +
TR−	α, β	Transfer to command in α if (β) are −
TRU	α	Transfer to command in α unconditionally

Class N-2, Repeat Operations. There are two operations in this class. The operations are:

Written		Signifying
RPT	n, i, j, k	Perform the next command n times, using as operands the contents of $\alpha+(n-1)i$, $\beta+(n-1)j$ and $\gamma+(n-1)k$, as required.
RWR	n, i, j, k	Same as above, except reset PACC (pseudoaccumulator) to zero before proceeding.

These operations apply only to the next operation. By letting the interpretive routine know in advance, and indexing by given numbers rather than the contents of an address, the repetition operates at maximum speed, requiring neither interpretation or command fabrication. Indexing by repeat commands is secondary and subordinate to indexing by index registers, and the simultaneous use of both is possible. Operands are then the contents of:

α indexed $+(n-1)i$, etc.

n is a one or two digit number, always plus but carried negatively in memory. i, j, and k may be 1- or 2-digit numbers, both plus and minus. When minus, they are carried in memory as unsigned 40,000 complements.

All repeatable (indexable) instructions interrogate the n position, which serves as a tally, before storing a result. If this is nonzero the tally is reduced by one and the operation is automatically repeated with further indexing by i, j, and k. If this is zero, it signifies either that the command was not intended to be repeated or that the command has been performed for the nth time. In either case, the program proceeds to the next command in sequence. In the case of the MAD and PMA commands, the result is not stored in γ until the tally is zero. Intermediate results are stored in PACC.

Class N-3, Set Index Operations. There are three operations in this class, one for each of the index registers. Each resets the contents of the specified index register to zero and stores the quantity j from the command as a limit tally. j is always a positive quantity and when converted (by multiplying by data word length) must be less than or equal to (the memory capacity of the *705*) − (10,000). The operations are:

Written	Signifying
CS1	Reset R1 to zero and set limit tally to j
CS2	Reset R2 to zero and set limit tally to j
CS3	Reset R3 to zero and set limit tally to j

Class N-4, Non-Test Transfer Operations. There are three operations in this class, one for each index register. Each augments the contents of the specified index register and the corresponding limit tally by the quantity k, which may be both positive or negative, and transfers to the instruction in α. To increase operating speed, all possible combinations of sums of index registers are also carried along and the appropriate combinations are also augmented by the quantity k. This permits single-indexing for any address. When converted, k is stored in either true or 40,000 complement form for unsigned add-to-memory. The operations are:

Written		Signifying
TN1	α, k	Augment all registers using R1 by k and transfer to the command in α
TN2	α, k	Same as above, except refer to R2
TN3	α, k	Same as above, except refer to R3

There are two methods of setting an index register to an arbitrary value. One is to give a CS1 and then a TN1 with that arbitrary value in the k position. The other, used when the present contents of the register are known, is to give a TN1 with k as the difference between the known and desired contents. The trans-

fer may merely be to the next command in sequence.

Class N-5, Test Transfer Operations. There are three operations in this class, one for each index register. They function in the same manner that the TN1 operations do, except that the transfer to the command in α is nullified if the limit tally for that register (which has been carried in a negative form) becomes equal to or greater than zero. The program then proceeds to the next command in sequence, and the use of all functions of that register is denied. The operations are:

Written		Signifying
TX1	α, k	Augment all registers using R1 by k and transfer to the command in α unless the limit tally for R1 equals or is arithmetically greater than zero
TX2	α, k	Same as above, except refer to R2
TX3	α, k	Same as above, except refer to R3

Appendix I

Some typical programming for matrix operations is given here to illustrate the use of PRINT I instructions. An example is given for the multiplication of a 6 by 5 matrix by a 5 by 4 matrix to produce a 6 by 4 matrix. This operation is symbolized as:

$$(M1ij)(M2jk) \rightarrow (M3ik)$$

and for convenience in following the indexing the complete matrices are given as:

$$\begin{vmatrix} M111 & M112 & M113 & M114 & M115 \\ M121 & M122 & M123 & M124 & M125 \\ M131 & M132 & M133 & M134 & M135 \\ M141 & M142 & M143 & M144 & M145 \\ M151 & M152 & M153 & M154 & M155 \\ M161 & M162 & M163 & M164 & M165 \end{vmatrix} \times \begin{vmatrix} M211 & M212 & M213 & M214 \\ M221 & M222 & M223 & M224 \\ M231 & M232 & M233 & M234 \\ M241 & M242 & M243 & M244 \\ M251 & M252 & M253 & M254 \end{vmatrix} = \begin{vmatrix} M311 & M312 & M313 & M314 \\ M321 & M322 & M323 & M324 \\ M331 & M332 & M333 & M334 \\ M341 & M342 & M343 & M344 \\ M351 & M352 & M353 & M354 \\ M361 & M362 & M363 & M364 \end{vmatrix}$$

For simplicity in explanation the elements have been assigned regional addresses symbolic of their row-column identification in the matrices. In actual operation they are usually loaded row by row in sequential addresses, but the same indexing principles apply without exception.

The programmer would write the following set of instructions:

Address	Op	Variable Field	Comment
B001	CS2	60	Set R2 to 0, limit to 60
B002	CS3	4	Set R3 to 0, (or reset)
B003	RWR	5, 1, 10, 0	Row-column
B004	MAD	M111, 2, M211, 3, M311, 2+3	product summation
B005	TX3	B003, 1	Step up R3 by 1
B006	TX2	B002, 10	Step up R2 by 10

Note that the development was chosen so that the elements of the result matrix were computed row-wise. Additional instructions could be inserted to print the matrix row-by-row during computation.

It is an easy matter to abridge the foregoing program for matrix-vector multiplication. A good practice problem is the coding of:

$$(M1ij)\{V1j1\} \rightarrow \{V2i1\}$$

Appendix II

Fig. 1 demonstrates the effect of the RPT command on timing. Average operation

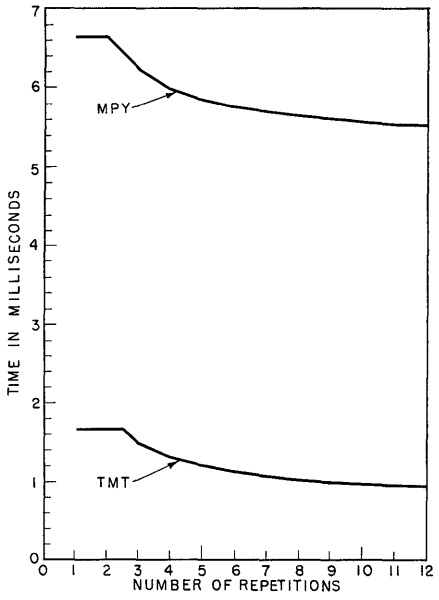


Fig. 1

times for a repeated series are shown versus number of repetitions.

MPY, when performed only once, takes 6.63 ms. When repeated, the first MPY takes 8.57 ms, the last takes only 4.72 ms, and all intermediates take 5.33 ms. The break-even point is at two successive MPYs.

TMT, when performed only once, takes 1.65 ms. When repeated, the first TMT takes 3.20 ms, the last takes only 0.46 ms, and all intermediates take 0.76 ms. The break-even point is between two and three successive TMTs.

Times for the first operation of a repeated command are high because they include the time for the RPT and its interpretation. It is obvious that a considerable time-saving may be effected by repetitive operations. They should be considered carefully not only for obvious operations such as matrix manipulation and block transfer, but also for linearly progressive calculation. Here it is advisable to group, by judicious choice of addresses, operations of the same type to take advantage of the reduced timing.

The IBM Type 705 Autocoder

ROY GOLDFINGER

THE *705* Autocoder is a system of automatic coding for use with the International Business Machines Corporation (IBM) EDPM* type *705*. However, before description of the system, the distinction between automatic programming and automatic coding will be emphasized. This will help to place the Autocoder (which, of course, is "automatic coder") in its proper perspective.

Programming is concerned with the general organization of a computer process. It encompasses such considerations as selecting the proper array of peripheral components, organizing the high-speed and auxiliary memories for the most economical access to information, deciding whether to process in one pass or two, and so on. These are programming problems which must be faced before any computer code can be written profitably.

Coding, on the other hand, is concerned mainly with a single command or an item of information. The job of the coder is to translate from the general pattern laid

ROY GOLDFINGER is with the International Business Machines Corporation, Poughkeepsie, N. Y.

* Electronic data-processing machine.

down by the programmer to the series of elemental machine-language commands necessary to operate the computer.

A human coder is capable of being both programmer and coder at once. That is, he may take the programmer's statement of an event to be performed, decide upon the best way of accomplishing it, choose among several alternatives, introduce considerations such as scaling, checking, detection of logical as well as machine errors, and simultaneously reduce the whole to the necessary series of machine-language entries. The machine coder, or "automatic coding system," generally falls far short of the ability of the human coder in the decision-making and optimizing area, while it outstrips him by far in speed, accuracy, and the production of legible results.

The finest of the so-called "automatic programming systems" are, in fact, excellent automatic coders. Those programs which have attempted to proceed further into the programming realm, intent on organization of storage and other decision-making functions, have had a significantly consistent tendency to fall into disuse. At the present state of knowledge on the effective use of large-scale general-purpose machines, it does not seem reasonable to attempt to formulate and develop into an automatic programming scheme a general theory of process-organization. The usual consequence of such an attempt is the crushing comment, "Yes, you have a clever system there, but unfortunately it won't fit the program I am working on now."

The *705* Autocoder is intended primarily to facilitate the coding of a particular machine, though it surely should be extensible to others of the same predominant general characteristic, that is, machines which are character-addressable rather than word-addressable. The Autocoder is intended as a tool to aid the human coder by assisting him in the ways in which an automatic coding scheme functions best, while leaving him on his own otherwise. Its principal features are these:

1. It allows the coder to define his constants and working data in terms of their familiar English-language names and their field lengths. He may address any item with which he is dealing by name and he may also refer to combinations of things, such as all the fields within a record under a single tag.

2. It extends the built-in *705* command code through the use of "macro-instructions," which are sequences of *705* instructions which experience indicates have a high-repetition rate.

3. It permits the incorporation of subroutines, which in turn may call in other subroutines, and so on. Since subroutines are written in Autocoder language it follows that successful programming becomes potential library material.

4. Since the Autocoder does not prescribe the organization of high-speed storage the programmer retains the freedom to establish any layout of instructions, constants, working and input-output space he desires.

Input to the Autocoder will be from punch cards generally, although tape input will be an alternative selection. Each input item, whether an instruction or the specification of a constant or datum field, will occupy a full card. This permits additional space for the coder's comment to be included with the required information. The Autocoder itself and its library of subroutines will be provided from a single tape.

The Autocoder assigns memory locations in order subject to the modification imposed by items in the input program which call for overlapping or skipping of selected areas of memory. In its processing of each input item it causes about 33 characters to be examined on a virtually character-for-character basis. This type of operation is feasible only because a character-addressable machine is available, and one which is quite fast in performing single-character operations.

The output from the Autocoder is a program in *705* actual, prepared in a form known as a load deck, that is, arranged in a form which permits rapid loading into the *705* memory, or onto drum sections or tape at the time it is desired to run the finished program. The Autocoder will produce such a load deck on cards or on tape alternatively. In addition there is produced a printed listing which displays in parallel the input program and the final result with actual memory locations.

The Autocoder requires three basic kinds of information to specify a program. These are:

1. *Instructions.* Commands to the *705* to perform certain logical operations on data or other instructions. These may be single *705*-type operations, or macro-instructions defining fixed sequences of *705* operations, or requests for closed-end subroutines.

2. *Constant definitions.* A listing of the required parameters, tables, and other constant information.

3. *Record definitions.* Detailed description and layout of input and output areas, and internally used storage areas.

A tag, consisting of a name or other descriptive designation, containing up to ten alphabetic or numeric characters is applied to any instruction, constant, or record field referred to by the program. Instructions which are not modified or transferred to need not be tagged. In addition to tagging individual constant or record fields, suitable combinations of them may be tagged also, as, for example, a PAY RECORD consisting of such fields as EMPLOYEE, SERIAL, GROSS PAY, and so on. Thus, it is possible to address whole records at a time with operations like READ, WRITE, or TRANSMIT (which is a unique *705* instruction for performing memory to memory transfers of any number of characters). A feature of the Autocoder is that it applies some logic to the operation itself to distinguish between operations involving fields within records, such as ADD and operations involving whole records, such as READ. The *705* requires that addresses be selected differently in these instances and the Autocoder adjusts accordingly.

Operations may have operands which are blank, to be supplied later by the program; operands which are actual, should it be desired to address specific memory locations; operands which are descriptive, as in addressing fields by their assigned tags; or operands which are literal. The use of literal operands is a novel feature of the Autocoder.

Literal addressing derives its name from the use of addresses which are the literal equivalents of the contents of the field being addressed. A literal address identifies a field in memory storage by specifying its contents. Other methods of addressing are "actual," "relative" and "symbolic," or "descriptive." In each of these systems a quantity stored in memory is addressed through some scheme of locating it, either relative to the entire memory, as in actual, or relative to some partition of memory, as in relative itself, or else by some unique label, as in symbolic. For example, the quantity +314159 (plus pi) may be addressed at its actual memory location, a relative location, or even at location PI as in SUBTRACT PI. However, the literal program employs SUBTRACT (+314159). The address is literally descriptive of the operand.

Whenever a programmer uses a scheme of addressing which is not literal he is saying, in effect: Operate on the quantity at the location designated by ————. In order to establish the proper location the programmer must maintain a record of location assignments which he consults whenever he refers to a constant or data field. The advantage of relative or symbolic addressing over actual is in the ease with which addresses may be assigned. However, the use of literal addressing obviates all look-up of address assignments. Thus computer coding is

brought a step nearer the spoken language which associates an action with its object.

By enclosing the literal quantity in parenthesis the coder tells the Autocoder that the address is literal. The Autocoder searches its list of already-stored literals, and unless the present one is a repeat assigns a new location for it. The number of characters within the parenthesis determines the length of the field established. Because the Autocoder does not allow the same literal to be repeated in memory it is possible to designate an area of working storage as well as a constant by this means. For, repeated references to any configuration of characters will result in the same actual memory location being addressed in the final program. Hence a variable may be stored and reused by addressing it in consistent fashion.

Areas of constants and records are described to the Autocoder by writing DEFINE CONSTANTS or DEFINE RECORD, using the proper codes, as operations, and then following with the fields comprising the area. Since the DEFINE line may itself bear a tag it becomes possible to refer to the whole as well as the parts. In the numeric column on the coding sheet is written the length of each field defined. On the DEFINE RECORD line itself may be written also the tag of some record previously defined. This instructs the Autocoder to overlap the present area on the one defined earlier. For example, on a single tape there may be two record types with different field arrangements. Both types will, of course, have been read into the same input area, but once having been distinguished, will require that their fields be addressed differently by the program. The ability to specify multiple overlays on the same actual memory area is the answer to this problem.

There is an advantage resulting from the descriptive tagging of record fields and from the fact that subroutines are written and stored in the same coding conventions as other programming. The expected situation in a given commercial data-processing installation is that there will exist a number of records which are common to many computer programs. For example, employee clock cards may be used by both the payroll process and some cost accounting procedures. Now, once the records are defined, the sequence of fields and their lengths established, it follows that each program and each subroutine within program may refer to the records by name. Hence it is possible for any number of coders to prepare library routines which refer directly to the common information of the installation. The actual filling in of the coding sheet and the subsequent key-punching has been made as minimal and as loose as possible. Nonsignificant zeros need seldom be written. Alphabetic fields are always written left-justified while columnar numeric fields are right-justified. This conforms to normal usage and should speed up training in the system while reducing clerical errors.

Although the *705* is itself rather intricately coded, requiring the specification of one of 40,000 characters of memory and a companion selection of one of 16 possible storage units in just four alphanumeric characters, the user of the Autocoder need not be concerned with the actual *705* addressing system. In fact, whenever he does refer directly to an actual *705* location he writes a 7-digit multiaddress which the Autocoder suitably encodes into the 4-character quantity recognized by the *705*. A projected system of macroinstructions should ease the task of address modification during the running of the program by providing the coder with an all-numeric, decimal logic with which to operate on an actual *705* address together with its binary superstructure.

The macroinstructions provide an exciting field for exploration. In a sense they allow the programmer to act as a creator of machine logic. The macroinstruction is like an open-ended subroutine in that it calls forth from a library of macrofunctions a sequence of *705* instructions which, under suitable modification, are inserted in its stead. However, the uses to which macro-instructions are being put and the concept of them seems quite new. They are regarded not so much as functions peculiar to specific classes of applications such as square root routines in mathematical applications, but rather as extensions of the built-in logic of the machine itself. For example, a common occurrence in running a machine is to type a message to the console operator. Ordinarily this requires the execution of a SELECT instruction to specify the output device, in this case the console typewriter, followed by a WRITE instruction, addressing some location at which a message has been stored. Instead of using these several instructions, the macroinstruction TYPE whose address is literally the message itself is introduced. The job is done in one line of coding instead of three, in this case. Even more powerful macroinstructions effecting input-output operations introduce tests for end-of-file, and error conditions. Should the latter occur, error-correcting routines are called automatically.

The two major categories of macroinstructions mentioned here so far are those dealing with address modification and input-output operations. In addition ways of causing the *705* to simulate other logics are being considered, such as floating decimal or fixed decimal with different scaling conventions than that built into the machine.

Also there is interest in the creation of additional commands which experience may indicate facilitate coding, such as a TRANSFER LOW, for example. Incidentally, this work should prove beneficial to the engineers who contemplate changes and new designs. The compound operations simulated most often are the ones the engineers would be encouraged to build into future equipment.

The *705* Autocoder is intended to be an aid to coding. It does not attempt to process more than an entry at a time, or to do more than the coder asks it to at any stage. The output program is under coder control throughout. Yet it is felt that this is a considerable asset, and it is felt that there are implications inherent in the Autocoder which have not yet been explored. An intriguing one is the possibility of direct and immediate transcription from flow diagram to computer input.

For example, in a process in payroll, on the flow diagram a box is found which directs the coder to SUBTRACT UNION DUES and the next to STORE GROSS PAY. But these are statements already in the language acceptable to the Autocoder. No further encoding is required. There may be subsequent boxes: WRITE CHECK RECORD and PRINT PAYROLL REGISTER LINE, all Autocoder language. The future possibilities for a thoroughly mechanical coding system are very favorable with a system like this for a starter.

However, it should not be inferred that a system as elementary as the *705* Autocoder is considered an end-all to machine usage problems. The preponderance of difficulty still resides in the systems-understanding and program-layout areas, especially for the commercial data-processing user. The technology and ingenuity in designing and running (including coding) of machines seems today well in advance of the understanding of what we are attempting to process with the machine. Automatic coding, of the kind exemplified by the *705* Autocoder, is a limited approach to a minor segment of a major problem.

Program Interrupt on the Univac Scientific Computer

JULES MERSEL

IN the use of high-speed digital computers the general situation has been to have the program or the programmer at the console in sole control of what the machine is doing. In recent years, however, a situation has often occurred where a program, so to speak, has only been lent the machine until a higher priority problem is ready to go on the machine. At these times the machine was stopped, the low priority problem was taken off the machine, and the high priority problem was allowed to run. Though such a situation is preferable to having a computer idle while it awaits the high priority problem, the time lost in getting the old problem off and the new problem on has been an unfortunate loss.

At the National Advisory Committee for Aeronautics' (NACA) Univac Scientific installation at Cleveland, Richard Turner foresaw that the foregoing situation would be the usual thing rather than merely an occasional nuisance. The way Mr. Turner intended to run his machine, and in fact does run it, the wind tunnel has first claim on the services of the Univac Scientific. It is important to NACA's operation that information from the wind tunnel reach the computer as soon as possible, that it be processed as soon as it reaches the computer, and that the results are returned to the wind tunnel immediately after processing.

The fast handling of wind tunnel information was achieved by running input-output lines directly from the wind tunnel to the Univac Scientific. Inasmuch as the Univac Scientific has the capability of having any kind of input-output, this was not a difficult task. The problem of quick access to the machine was solved by Mr. Turner's suggestion of the "interrupt" feature.

The interrupt feature has the following characteristics. When an interrupt signal is sent to the Univac Scientific, the computer, immediately after finishing its present instruction, notes the location of which instruction it would normally do next, jumps to a fixed address which will allow the value of this location to be stored and from which the computer will find the information as to what it is to do because of the computer having been interrupted.

As an example, in the wind tunnel operation, when the wind tunnel is ready to use the computer, an interrupt signal is sent from the wind tunnel. At the completion of the current instruction, the Univac Scientific notes where it is in the current problem, stores the current problem away, reads in the program for the wind tunnel problem, and starts that program. The wind tunnel problem then reads in its own input from the wind tunnel, executes the needed computation, and returns the answers to the wind tunnel. As soon as the answers are returned it reads the previous problem back into the rapid access storage and continues it from the point where it had been interrupted. A delay of microseconds has been incurred in getting ready for the switchover rather than the much lengthier delay which might be expected without the use of a program interrupt.

This application of a program interrupt inspired much thought as to other applications on which the program interrupt might profitably be used. The wind tunnel application is an input-output application with a device that is not normally considered part of a computer. Thought was naturally given to whether the program interrupt might be profitably used with ordinary input-output equipment. Such pieces of peripheral gear as punched card input and output equipment turned out almost to demand the aid of the program interrupt.

In handling punched cards the normal procedure is that, for 12 very short intervals during the whole card cycle, information must be either read from the rows of the card or punched onto these rows or both. Since on the Univac Scientific less than 1 per cent of the time of the card cycle is required for the reading or writing of rows, the programmer usually attempts to perform computation during the remaining time. This, unfortunately, requires that he carefully segment his computation into 12 groups none of which is time-wise long enough to interfere with the processing of row information. This is an onerous task and usually results in either safe but inefficiently short segments of computation or segments that cause machine errors due to being too long.

The card input-output device on the Univac Scientific was modified so that it would send an interrupt signal when it was ready either to release or receive row information. The need for the careful segmentation of the computation disappears. When the interrupt signal is received, the computation is stopped, the row information is either written, read, or both, and then the computation is continued from the point of interruption. The elimination of the need for careful timing of computation usually allows much more computation to be performed during the card cycle.

Probably the most interesting uses of the "program interrupt" center around the task of using two computers on the same problem.

Some Univac Scientific users intend to use their machine in conjunction with analogue computers. In this case the analogue computer interrupts the digital computer whenever it is ready to transfer information. The Univac Scientific then processes the analogue data, returns it if needed to the analogue computer, and then awaits new data from the analogue computer. If much delay is expected in receiving information from the analogue computer, computation on other problems may take place during the wait. This, of course, greatly increases the efficiency of use of the computer.

One user is planning to use two Univac Scientifics in tandem. His is a real time problem. One computer smoothes the raw data that comes in from the outside; the other computer processes the smoothed data. In such a case whenever the second computer is ready to receive more data it interrupts the first computer and through the Univac Scientific's versatile input-output system the smoothed data are transferred from the first machine to the second machine. This allows the maximum amount of smoothing to take place within the time limits of the problem. Many other applications can be visualized where in real time problems it is necessary to use two computers in tandem since for these applications no one computer is fast enough to do the job by itself. Since real time problems at such installations would occupy only a small part of the computing day, the two computers are freed to be used separately during the rest of the day.

The last application occurs when, because of the form of graphs displayed on

JULES MERSEL is with The Remington Rand Corporation, St. Paul, Minn.

the Univac Scientific's scope display system, the programmer wishes to change the flow of his problem. If he has previously decided on the alternate course or courses he wishes to pursue due to scope display information, he need only press the interrupt button on the console to have the computer proceed on its new course. This avoids the stopping and resetting of the computer and the many errors that are inherent in trying to do this quickly.

Like many other new ideas in computer design it is expected that the stimulus of the needs of the many computing installations around the United States will father many other and exciting uses for the program interrupt. The program interrupt is confidently expected to be a feature of most digital computers in the future.

A Pulse-Duration-Modulated Data-Processing System

J. R. LOWE J. P. MIDDLEKAUFF

A COMMON engineering problem is that of measuring physical quantities which cannot be directly observed. In the case discussed here, these quantities exist in a missile during flight. Since it is not practical to mount recording instruments in the missile and recover them after the flight, the data must be recorded somewhere outside the missile so that they can be reduced to graphical or tabular form.

The usual solution to this type of problem is some form of telemetry. This paper considers one such form, namely, pulse duration modulation (PDM) telemetry, and describes a system for reducing data so obtained. The system represents a great improvement in speed, flexibility, and accuracy over previous reduction methods. Considering the time factor alone, the system described in this paper is a striking improvement. From 3 to 6 months were formerly required to obtain analyses of the data from a missile firing. Now, these can be provided in 2 or 3 days.

In the system described here, the data are transmitted by radio link from the missile and recorded in analogue form on magnetic tape at the ground station. This analogue tape is converted to digital tape which is processed by an IBM type *701* Electronic Data-Processing Machine. The *701* punches cards which are used to produce graphic presentations on an IBM type *407* Accounting Machine. The system provides for a "quick look" at the data, handles nonlinear calibration factors, plots only selected portions of the data, allows for many kinds of errors in the telemetry, permits subcommutation and supercommutation and numerous other exceptional circumstances. The reduced data are also left on magnetic tape ready to be used as input for other *701* programs.

J. R. Lowe and J. P. Middlekauff are with the Douglas Aircraft Company, Inc., Santa Monica, Calif.

Brief Description of PDM Telemetry

This is a standard Applied Science Corporation of Princeton (ASCOP) system and will be described very briefly.

The missile carries transducers which measure such quantities as deflections, temperatures, pressures, and accelerations. All these quantities are developed as percentages of a battery voltage.

The transducers are sampled by a commutator (see Fig. 1). In the example shown here, the commutator has 30 contacts (can sample 30 transducers) and rotates 30 times per second, thus producing 900 samples per second. However, the system is entirely flexible in this regard, and other numbers of segments or commutation rates can be readily handled.

Square wave pulses whose durations are proportional to the sampled voltages are formed, and are differentiated twice. The resulting pulses frequency-modulate a radio wave which is transmitted to a ground station, where the pulses are demodulated and recorded on magnetic tape in analogue form. Simultaneously, a series of 0.01 second time pulses are recorded on a separate track of the magnetic tape. The data track is in the form of a series of groups of 30 "channels," each channel corresponding to a point of the commutator. Each group of 30 channels, corresponding to one revolution of the commutator, is called a "frame."

In practice, two adjacent segments of the commutator are not used. This produces two blank channels in every frame which are useful in identifying the frames. Also, two segments are used to measure the two sides of the battery called "ground" and "reference" voltages. If two batteries are used, as is frequently the case, there are two reference voltages and a common ground.

Fig. 2 is a sample telemetry record called a "lines record." This is a 35-millimeter film containing the same information, and produced by the same type of signal, as the analogue tape. Here the pulse widths are represented by line heights and the time marks by dots along the bottom of the film.

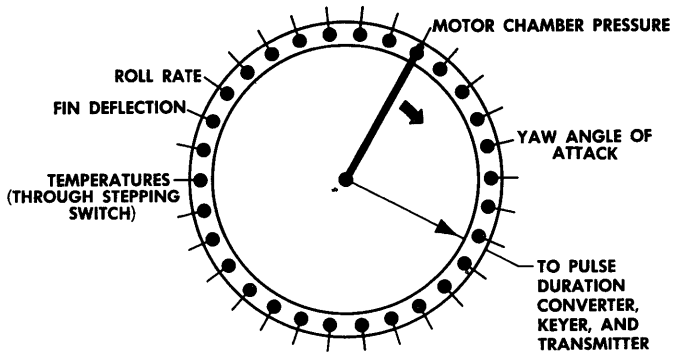


Fig. 1. Commutator

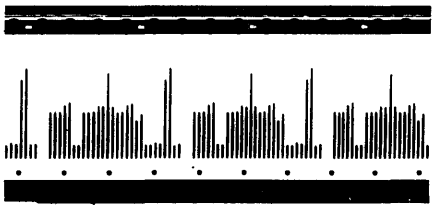


Fig. 2. Normal lines record. Pulse duration is represented by line height and time marks by dots above bottom edge

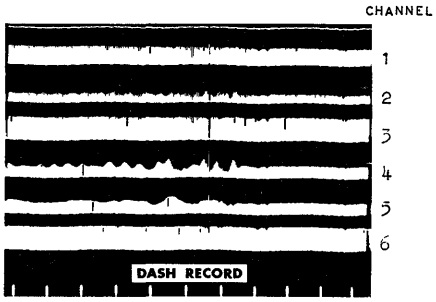
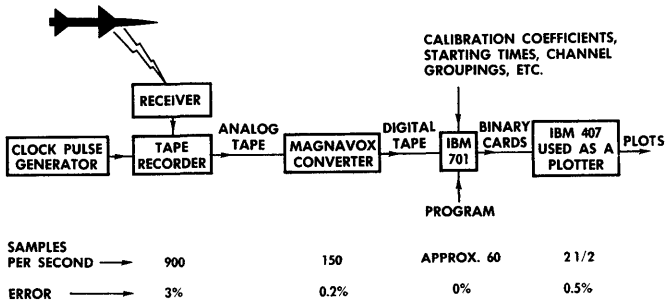


Fig. 3. Dash record. Vertical lines along bottom edge are time marks

Fig. 4. System block diagram



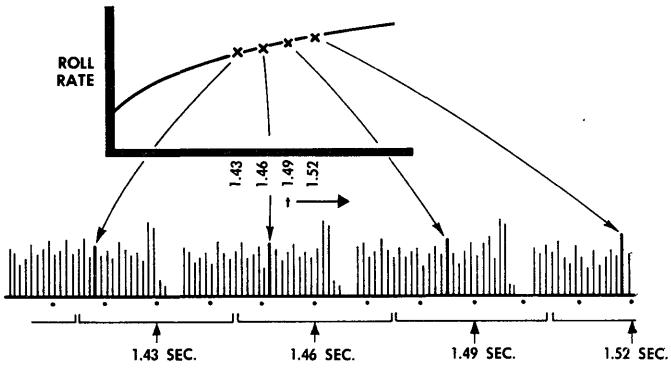


Fig. 5. Stripping

During the missile flight, a set of "dash records" (Fig. 3) are prepared and are available very soon after the flight. In these, the information is "decommutated" or "stripped" so that each channel appears separated from the others. Dash records are useful as qualitative "quick look" media and as an aid in determining how much of the total data should be reduced and plotted.

Outline of the System

Fig. 4 shows a block diagram of the system. The first step in processing the data is to convert the analogue tape to a digital form using a Magnavox Series *200* Data Converter which was specially built for the Douglas Aircraft Company. This will be described more fully in the following paper, "A PDM Converter" by Arsenault (pages 57–61). As will be developed in that paper, the design of the converter presented two basic problems: First, while the pulses on the analogue tape vary in their repetition rate due to variations of commutator speed in the missile and other factors, the recording density on the digital tape must be constant. This problem was solved by controlling the speed of the output tape unit by a phase-sensitive synchronizer.

Second, it was necessary to provide "inter-record gaps" or 1-inch spaces on the output tape after a predetermined number of samples, e.g., 1,500 samples. Since it was impractical to stop and start the input tape to provide these gaps, this problem was solved by using two intermediate tape recorders, recording blocks of data on them alternately, and finally playing them back alternately onto a composite tape in such a manner as to leave the required gaps. Incidentally, it is probably possible to solve this problem more elegantly using a more recent idea. This would involve inserting, after each block, a bit on an otherwise unused channel of the output tape which could be used to cause the *701* to stop reading the tape and back up to the marker.

The channels on the digital tape are in the form of *701* half words, or 18 binary digit groups. Ten binary digits are used for data, one is used to represent the time pulses, and seven are blank.

As noted in the introduction, this tape is then processed through the *701*, which punches binary cards. The cards are used to produce plots on the *407*. These two steps will be discussed in the following two sections.

The 701 Program

The raw data as transcribed by the Magnavox Converter must be processed in several ways in order to produce the desired final graphical output. One of the salient advantages of the system lies in the use of a general-purpose computer, already on hand, and a very flexible program.

The same function from successive frames must first be selected and grouped together in a time sequence. This is referred to as "stripping." Also, a time must be associated with each sample. Since the plotter, the Type *407*, can give only uniform increments in time, the system selects, for each plot, the quantities which occur nearest the successive times defined by the successive abscissas or grid lines on the graph paper. These two steps are illustrated in Fig. 5.

Note that this method of ascribing times to the successive data points introduces some error, since the events do not occur at the precise times that the graph would indicate, but merely nearer to those times than to the adjacent times. However, experience indicates that this limitation introduces no objectionable error.

Defective data, in the form of missing pulses, overlong pulses, or broken pulses (see Fig. 6), are represented as zeros by the Magnavox converter. The program rejects these points, and inserts for each such point a signal which causes the *407* to space without printing (a space representing an advance in time). It is thus possible for the program to indicate time accurately over a long series of bad or missing data points.

Defective timing, either in the form of missing time marks or spurious time marks, is also occasionally encountered. Normally, time marks occur at the average rate of one every nine channels, ±10 per cent. Making use of this fact, the program deletes spurious time marks or supplies missing time marks as required.

The program next handles the problems

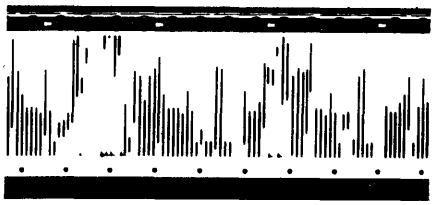


Fig. 6. Lines record showing defective data

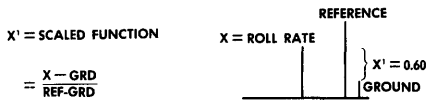
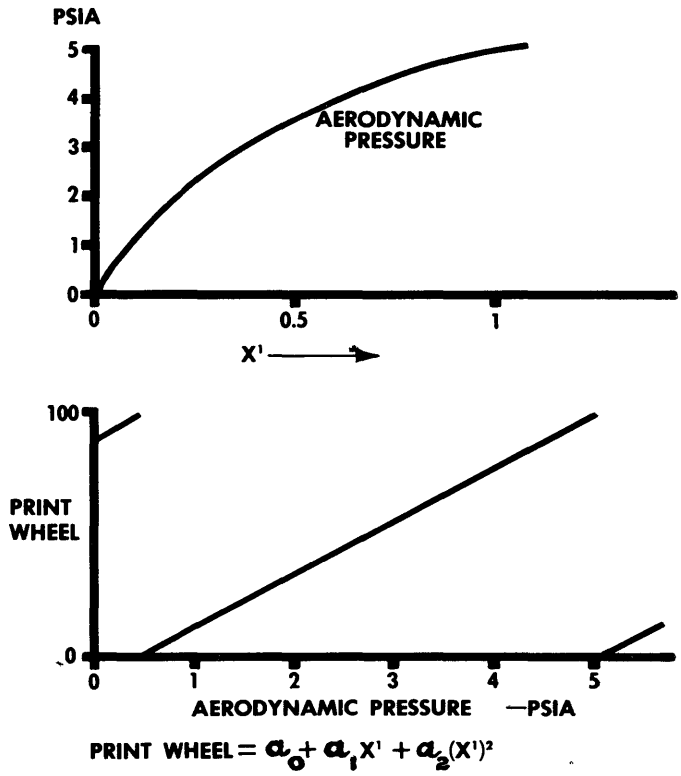


Fig. 7. Scaling

Fig. 8. Calibration



of scaling and calibration. For the purposes of telemetry, a quantity is represented as a percentage of the difference between a ground voltage and a reference voltage. With such a representation, it is not necessary for the voltages in the missile to be accurately regulated. Fortunately, any distortion of the data through the preparation of the digital tape is entirely harmless, so long as the distortion is no worse than linear. It is clear from the equation of Fig. 7 that any such linear distortion drops out in the scaling operation. The *701* program uses the ground and reference channels closest in time to a particular channel in the scaling of that channel.

The purpose of calibration is to convert the scaled quantities to physical units according to calibration curves determined for each of the transducers in the missile. These physical units are then converted to a *407* type-wheel location in such manner that the final graph will be on a convenient scale. This operation is illustrated by Fig. 8, where the 101 type wheels used for plotting are arbitrarily numbered 0 to 100. (Actually, the two steps of calibration and type-wheel determination are combined in one step by the computer.)

The calibration curves are first approximated by polynomials which are mostly linear but which may be of any degree up to five. The coefficients of these curves are supplied to the computer as part of the identifying and specifying information for a particular flight.

The normal commutation rate of 30 frames per second is not always high enough for the sampling of rapidly changing functions. This rate can be increased at the sacrifice of one or more channels by jumpering two or more commutator segments so as to sample a particular function two or more times per frame. This is called "supercommutation." On the other hand, it is frequently desired to sample more quantities than there are available commutator segments. This can be done by connecting several transducers to one segment through a stepping switch which steps once each frame. This is called "subcommutation." As is shown in Fig. 9, one of the quantities in the sampled group, called the reference voltage, is larger than the rest, and provides for the recognition of the sequence of functions with the group. Information regarding supercommutated and subcommutated channels is supplied to the *701* as part of the identifying information, and the machine selects and organizes the data appropriately.

The computer punches its output in the form of binary cards (binary rather than decimal to conserve *701* output time), one deck for each plot to be prepared. These decks must take cognizance of the number of curves to appear on each plot and this information is again supplied as part of the identifying information.

As another form of output, the scaled data are written on magnetic tape and are available for use by other programs. Examples of this use are the computation of Mach number, drag, and other aerodynamic forces.

In order to allow the operation of the program to be followed as it is being run, a table is printed showing the starting and ending times for each record and other relevant information (see Fig. 10).

It is worth emphasizing that these steps require only one reading of the input tape. Any number of plots, each with its own start time, end time, and time increment, may be prepared.

Plotting on the Type 407 Accounting Machine

The plots are printed on continuous rolls of vellum graph paper. Time is represented by the paper feed spacing, one space being equivalent to one time increment. Ordinate values are represented by printing from one of 101 type-wheels. From one to five functions are put on one plot by using five different symbols selected from the 47 available.

Seven binary digits are necessary to specify a type wheel. Ten numbers are punched in each row of the card, and all 12 rows are so used. Thus, 120 points are punched in each card (see Fig. 11). The *407* reads each card 120 times, printing one symbol at each reading, and taking 48 seconds for each card. On each line, the machine prints the first symbol, then the second, and so on. When it has printed a number of symbols equal to the number of functions to be plotted (this number is also punched in the cards), it spaces and repeats the symbol printing cycle.

It is desirable for one abscissa grid line to represent a convenient time increment. This is achieved by choosing the proper combination of graph paper, carriage

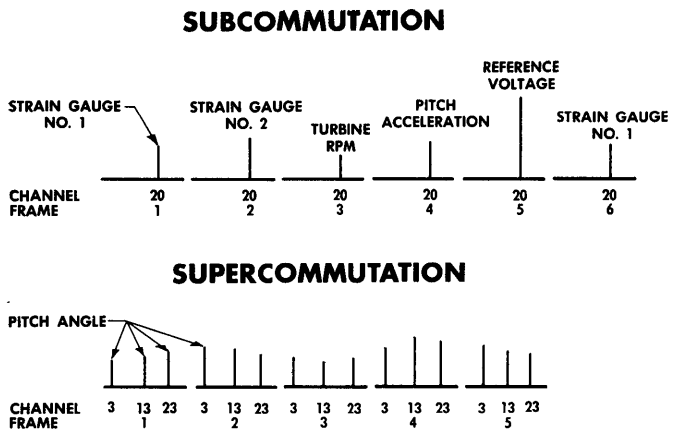


Fig. 9 (left). Subcommutation and supercommutation

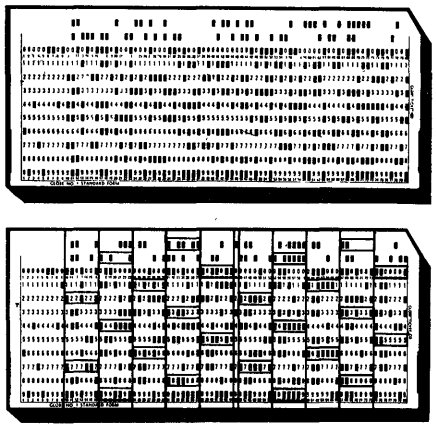


Fig. 11. Binary cards. The circled numbers represent successive values of one function

JOB 5569 TELEMETRY DATA REDUCTION RUN 475

INITIAL REFS	CHANNEL 25 679.00	CHANNEL 26 694.00	CHANNEL 27 118.00	CHANNEL 28 123.00

RECORD NO.	FIRST T	LAST T	NO. FRAMES	EMPTY FRAMES	TAPE ERRORS
1	1.43–	.15	50		1
2	.16	1.74	50		
3	1.75	3.33	50		1
4	3.34	4.93	50		
5	4.94	6.52	50		
6	6.53	8.11	50		
7	8.12	9.70	50		
8	9.71	11.29	50		
9	11.30	12.88	50		
10	12.89	14.47	50		
11	14.48	16.06	50		
12	16.07	17.66	50		
13	17.67	19.25	50		
14	19.26	20.84	50		
15	20.85	22.43	50		
16	22.44	24.02	50		
17	24.03	25.62	50		
18	25.63	27.21	50		
19	27.22	28.80	50		
20	28.81	30.40	50		

Fig. 10. Printed output

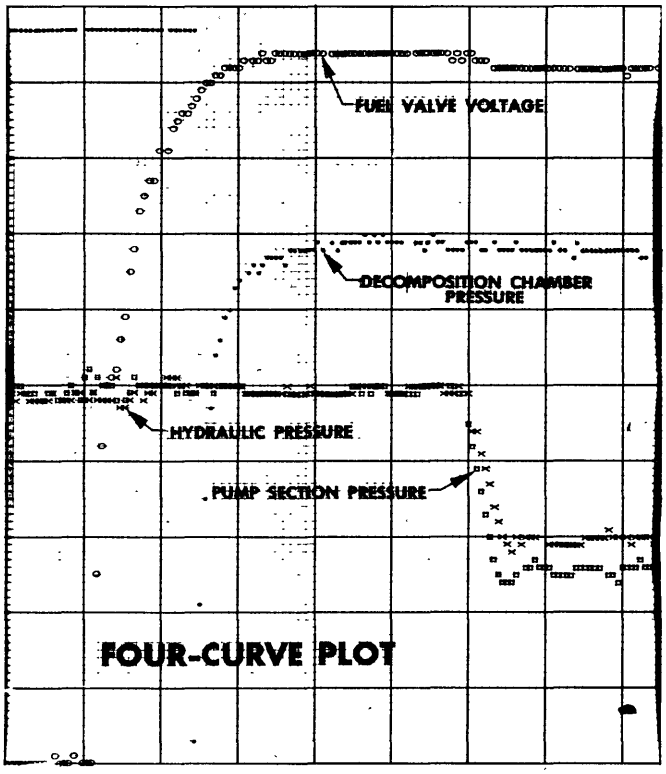


Fig. 12. Output of 407

spacing, and time interval between successive lines of printing.

A portion of a plot is shown in Fig. 12. Note that the plot is rotated 90 degrees from its position in the *407*. The symbols "X", "□", "0", and "." are used here to denote four curves. Because of the *407* spacing employed for this plot (one space equals 0.06 inch), the symbols overlap somewhat. The time increment in this example is 0.03 second, and the resulting scale is 1/2 second per inch.

The vertical range is covered by the 101 type wheels. If the result of calibrating a point is a number lying outside the range 0 to 100, the type wheel represented by the difference between 100 and the absolute indicated value is selected instead (Fig. 8). Thus 103 is represented by type wheel 3 and −7 by type wheel 93. The curve labeled "Decomposition Chamber Pressure" in Fig. 12 is an illustration of this.

Summary

A system has been described for reducing data from standard ASCOP PDM telemetry. This system has certain advantages and certain disadvantages.

One of the design criteria for the system was that special-purpose hardware be kept to a minimum. The only device especially constructed was the Magnavox Series *200* Converter, which actually is not particularly special-purpose, since it will handle almost any form of PDM magnetic tape. This was a necessity since the system was designed to handle telemetry data from widely different missiles and from other sources such as static tests and wind tunnels. There were so many different circumstances and such a variety of possible errors in the data that special-purpose devices to handle them all would be very expensive. It was decided to relegate as much as possible of the work to the *701*, where a flexible program could easily take care of unusual circumstances.

An outstanding advantage of the system is the fact that the reduced data are automatically available to the *701* on magnetic tape for use in performing further calculations. Sometimes these data are correlated with other data from the same flight, such as radar tracking data.

The error introduced in the data by the reduction system is considerably less than that introduced by the telemetry system. This point is illustrated by Fig. 4.

The system provides for a preliminary look at qualitative information and for reducing only the regions of interest.

The speeds of the system, as illustrated by Fig. 4, are entirely adequate for handling data from missiles, whose flights last

no more than a few minutes. However, where data are being taken for very much longer periods of time, as would be the case in aircraft flight testing, several improvements are indicated, the most important being increased plotting speed.

There seem to be two paths that this development can follow. First, put the output on magnetic tape, instead of binary cards, in a form suitable for a tape-driven high-speed line printer. This would enable the printer to print all the characters for one line in one machine cycle, and also take advantage of the higher printing speed of the new printers. The second way would be to use a cathode-ray oscilloscope output for the *701*. This would reduce *701* output time, and leave as the only subsequent step the development and reproduction of the film. This method would have the further advantage of reducing the error presently introduced because of fixed time interval in the plotting.

In general, the system has been very satisfactory and has adequately fulfilled a definite need.

A PDM Converter

W. R. ARSENAULT

DIGITIZING Pulse Duration Modulated (PDM) data at a rapid rate and presenting it in a suitable form for data reduction has been a problem of data reduction centers for some time. The Magnavox Series *200* Converter is designed to accept PDM data recorded on magnetic tape, automatically digitize it, and record the digital information on the magnetic tape in a form suitable for input to a digital computer or other data reduction equipment.

The original PDM data are obtained by recording telemetered or ground data in the usual way during test runs. Using two intermediate tape units, the converter produces appropriate gaps in the digital output tape, making it compatible with such formats as that used by the International Business Machines Corporation (IBM) *701*. A data-tracking servo is incorporated in order to keep the digital output tape at constant density regardless of variations in the sample rate of the input tape. The servo also acts as a noise filter, producing a recording continuity on the final digital tape during periods of sporadic noise or long intervals of interrupted telemetered data.

Introduction

Pulse Duration Modulation has been used for some time now as an information carrier in telemetry systems. The method is to sample various analogue types of signals, usually appearing as a direct voltage, and converting this into a pulse, the duration of which is a function of the magnitude of the signal. This same result may also be realized by having a pulse, say positive, indicate the start of the duration and a second pulse, negative, indicate the end of the duration.

The system for which the Magnavox Series *200*, model *201* converter was designed, transmitted these pulses via a frequency-modulation system and eventually recorded the data on tape. Fig. 1 shows how these data appear on tape. The original sampling system contains a commutator that sequentially switches various direct voltages into a unit that generates the PDM data. This unit is called a keyer. The output from the keyer is a sequence of pulses, spaced equally in time but of varying duration. This shows up in Fig. 1 as pulses with equal intervals. In the particular system described the commutator has 30 segments. Of these 30, 28 contain data and two are left blank.

There are various ways of reducing these data to a usable form. Analogue methods have been used to scale and calibrate the samples and plot them directly. Digitizing the data allows processing by a digital computer. The general subject of processing these PDM data in a computer is covered by Lowe and Middlekauff.[1]

Conversion Problem Defined

The major design problems encountered were those in making the output tape compatible with the tape units and format of the computer with which it is being used. Specifications on both the input and output tapes will be discussed before discussing these problems further.

As pointed out in the "Introduction," the information recorded on the input tape originated in an air-borne keying system. In this sampling equipment, a commutator with 30 segments revolves at 30 revolutions per second providing 900 samples per second. The commutator scans sequentially the various instrument channels and provides the inputs to the keyer. The output eventually appears on a frequency-modulated carrier, telemetered to a ground station. On the ground it is converted to a pulse form and recorded on magnetic tape.

The ground recorder is an Ampex Model *309*. This recorder uses a 1/4-inch tape and two recording channels are provided, one for PDM data and one for frequency-modulated (FM) recording. In this instance, the major data are on the PDM channel with the FM channel used for timing markers.

During recording on the ground the tape runs at 60 inches per second. A pictorial view of the information as it is recorded at the ground station is shown in Fig. 1. The pulse interval from leading edge to leading edge is shown to be constant (within the specified tolerances) and corresponds to the interval from one segment to the next on the commutator. The pulse duration is a function of the parameter being measured at that time. There are 28 pulses of varying width on this tape corresponding to the sampling of 28 different parameters by the commutator. Two segments are left blank in order to define a single frame or commutator rotation. Timing in Fig. 1 is for playback at 15 inches per second.

This magnetic tape now becomes the input tape to the converter. Specifications as they apply to a single pulse are shown in Fig. 2. The nominal pulse interval of 0.067 inch is derived from 900 samples per second being recorded at 60 inches per second. This interval may

W. R. ARSENAULT is with the Magnavox Company, Los Angeles, Calif.

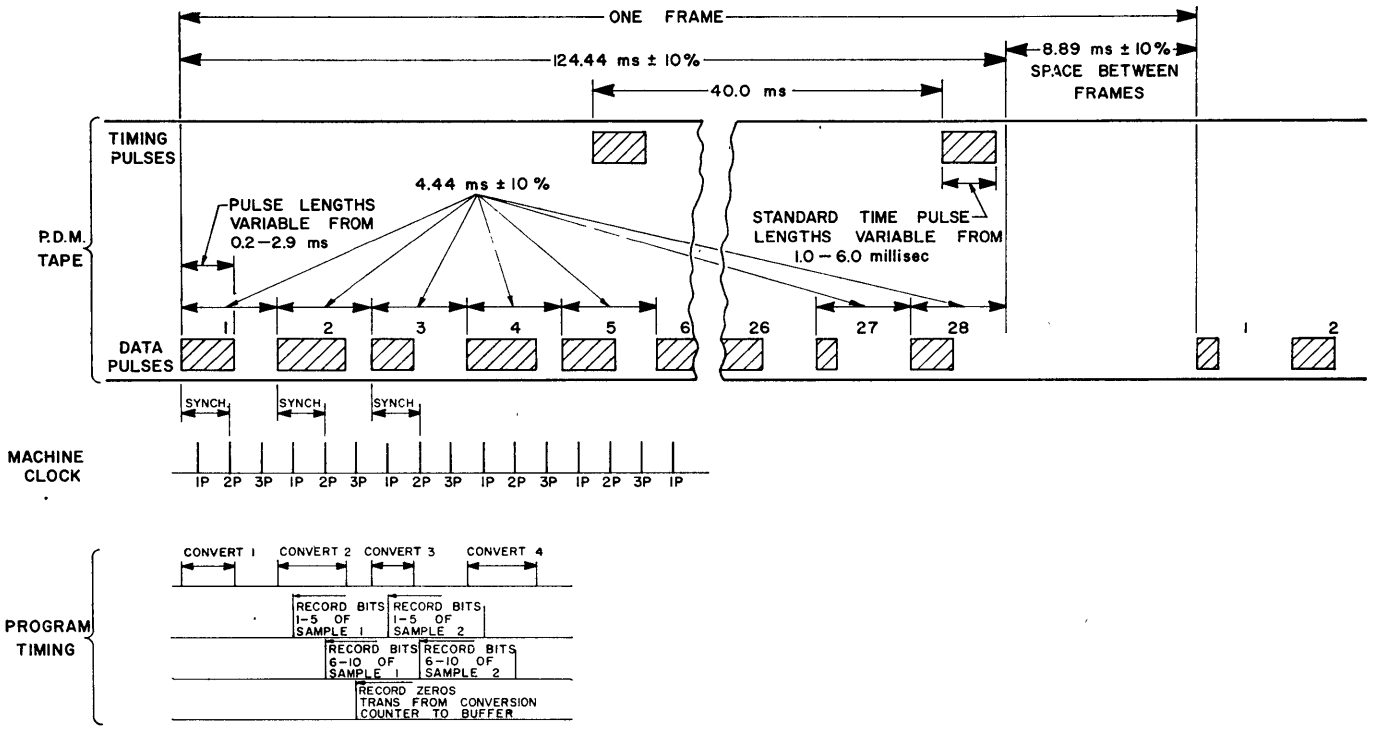


Fig. 1. PDM input tape and machine timing

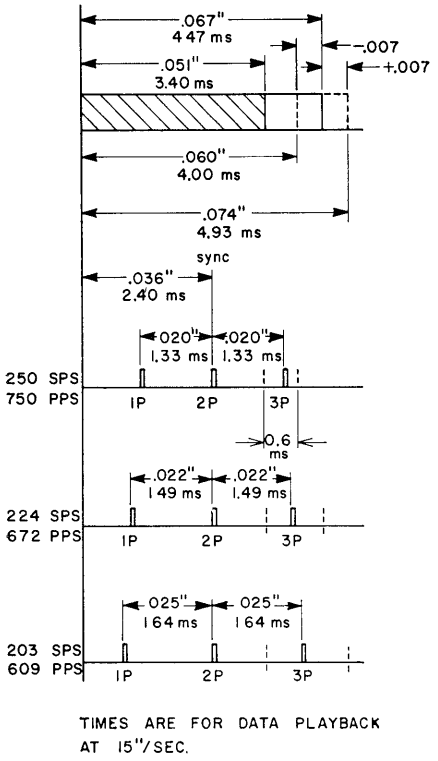


Fig. 2. Specifications of input sample pulse. Times are for data playback at 15 inches per second

vary ±10 per cent over a long period of time due to variations in the commutator speed; thus the tolerances of ±0.007 inch. The longest allowable pulse width, representing the sampled information, is 0.051 inch.

Fig. 2 also gives the pulse intervals in time when the input tape is played back at 15 inches per second. In the lower portion of the drawing the relation of the 3-period converter clock is shown and will be explained later.

In the particular application being discussed here, the output tape from the converter must be compatible with an IBM 726 tape handler. This is a 7-channel 1/2-inch tape with a format shown in Fig. 3. Six rows of six tracks constitute the 36-bit IBM *701* word. The seventh channel is a parity check channel; the information stored here is such that the sum of the seven digits in one row is always odd.

The 36-bit IBM word may be divided into two half words. The format of Fig. 3 shows a digitized sample recorded in each half word. The next word is adjacent to this one as is the following word. There is no discontinuity between these words, each being identified as a group of six rows. At predetermined intervals there is a 1-inch intra-record gap. The interval between intra-record gaps defines the amount of information to be read into the high-speed memory of the data processor during one reference to the IBM *726* tape handler. The density of the output tape must be 100 bits per inch with an allowable variation of only ±3 bits per inch.

With the specifications for the input and output tapes cited, the main problems for the converter can be defined. These may be broken down into two categories:

1. Operating the converter in synchronism with the input tape, the output digital information must be recorded at a constant density, although the input sample rate may vary as much as ±10 per cent.

2. A 1-inch gap must be inserted in the digital output tape, at predetermined intervals, while accepting continuous input samples.

Elaborating on these two points, it is necessary to keep a machine clock in synchronism with the incoming data samples so that the conversion and recording in digital form can keep in step and not lag behind these incoming data. With the machine clock, and, therefore, the recording rate, varying, it is necessary to vary the speed of the output tapes so as to record a constant information density.

The second item mentioned was that of inserting a 1-inch intra-record gap in the final tape without loss of the input data. The two blank channels, 29 and 30, do not allow enough dead space to stop and start the input tape, even with the fastest digital tape handler. Further, these digital tape handlers do not have the flutter and wow characteristics required to play back the analogue input tape.

Other functional requirements of the converter were relatively easy to solve. In previous discussions, it has been pointed out that several of the analogue samples are carried as calibration and scaling factors. All information within

a frame, 28 samples plus two blanks, is referenced to these scaling factors. Requirements set on the converter do not include any absolute reproductions but only relative accuracies. The speed of the input tape must be kept constant over several frames so that there is no speed change within a frame. If these specifications are met for playback, then the samples can be treated as time duration pulses from the playback tape rather than the physical distances that they are.

System Description

The first problem outlined in the previous section, that of keeping the conversion in step with the incoming data and keeping the digital recording on the output tape of constant density, was solved by incorporating a data tracking servo. This was designed to keep the converter clock synchronized with the incoming data, with a minimum of phase error. As the incoming sample rate increases, the machine clock increases accordingly. The machine clock directly controls both the conversion to digital form and the recording, thus keeping these functions in step.

A simplified block diagram of the converter is shown in Fig. 4. The relation between the incoming data, the machine clock, and the machine program is shown in Fig. 1. There are three machine clock pulses for each data sample. This corresponds to recording the three rows of information per sample on the digital tape.

The PDM data are routed to both the data tracking servo and the conversion counter. The data tracking servo uses the leading edge of the sample as synchronizing information. The leading edge of the data pulse is delayed a fixed amount and called the "sync" pulse. The machine clock $2P$ is kept in line with the sync pulse by the servo.

Fig. 5 is a simplified diagram of the data tracking servo showing the phase error detection supplying an input to the operational amplifier. The delayed leading edge pulse is compared with $2P$ of the machine clock. If there is a leading phase error, machine clock ahead of data, a positive pulse of width equal to the magnitude of phase error is input to a smoothing network. The output of this network, and input to the operational amplifier, is a function of the amount of phase error and is greater the larger the error. This signal is inverted in the amplifier and appears as a negative going signal. This makes the multivibrator bias more negative and reduces its frequency.

If the machine clock lags the input data, a negative pulse drives the amplifier, and inversion in the amplifier increases the multivibrator frequency. There are two paths for the error signals through the amplifier. The first, through the capacitor, provides a fast action phase correction and the second, through the resistor, gives an integrated frequency correction. Both are needed since it is possible to have a phase error without having a frequency error.

Referring again to Fig. 2, a more detailed drawing of the input pulse timing and the machine clock variation as a function of input pulse interval is given in the lower section. The tracking servo is designed so that it lines up the sync pulse, which is the leading edge of the sample delayed 2.40 milliseconds, and the mid-clock pulse $2P$. As the input sample interval changes from minimum (4.00 milliseconds), to nominal (4.47 milliseconds), to maximum (4.93 milliseconds), the position of these clock pulses is shown relative to the input sample. In all cases the $3P$ pulse is approximately centered in the interval between the end of the longest pulse width and the leading edge of the next sample. $3P$ is used to gate the conversion counter, and centering $3P$ in this manner allows correct conversion of data over the greatest range of phase error between the machine clock and the incoming data.

The output from the operation amplifier also controls a second multivibrator which in turn controls the capstan speed of the intermediate tape units. As the input data rate increases, and therefore the machine clock, the capstan speed increases accordingly. This guarantees that the digital information recorded on these units will be of constant density regardless of the variation in the input sample rate.

The data tracking servo also acts as a noise filter providing a continuity of digital recording during periods of sporadic noise or temporary periods of complete absence of data on the input tape. The machine will continue to record zeros during an absence of data at the input at a repetition rate indicated by the most recent input samples.

Referring to Fig. 3, the PDM data go into a conversion counter. The conversion counter is a straight 10-digit binary counter. The leading edge of the input sample opens a gate and allows pulses from a crystal controlled oscillator into the counter. The pulses are counted until the trailing edge of the sample closes the gate. At this point a digital number proportional to the width of the

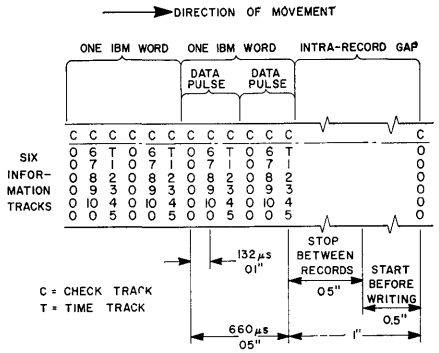


Fig. 3. Output tape format. This figure illustrates format for two channels of data. "1" is the most significant digit of a number; "10" is the least significant. The third row except for check track contains zeros in all cases

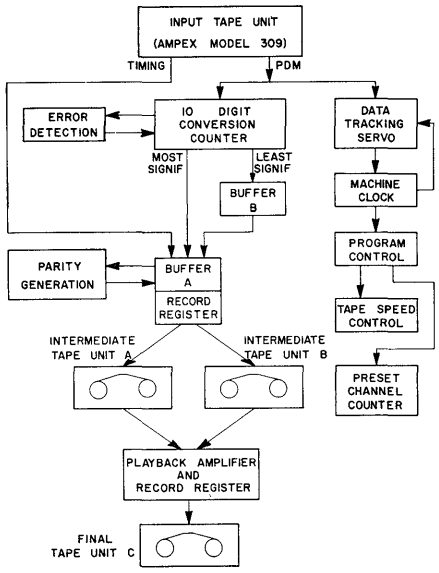


Fig. 4. Simplified block diagram

particular sample converted is in the conversion counter. This is transferred to a buffer register prior to the appearance of the leading edge of the next sample. During the time the second sample is converted, the first is recorded on the digital tape. This program sequence is depicted in Fig. 4. The format of the output tape (Fig. 2) shows the first five digits recorded in one row and the second five in the second row. This recording takes place at machine clock times $1P$ and $2P$ respectively. The third row is left blank, recording only zeros, making up the IBM 18-digit half word. The seventh track contains the parity digit. There is a parity bit for each row and each is generated in the buffer just prior to recording of that row.

The format of the output tape allows one digit for a timing bit (T). If a timing mark appears on the FM track of the input tape during the time a particular

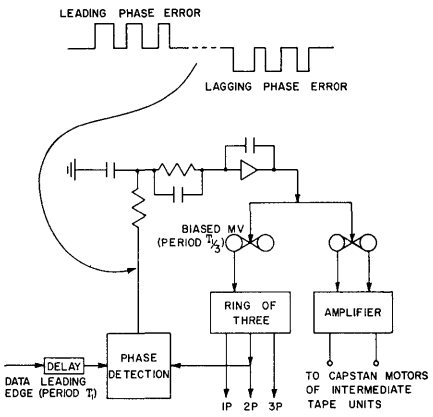


Fig. 5. Data tracking servo

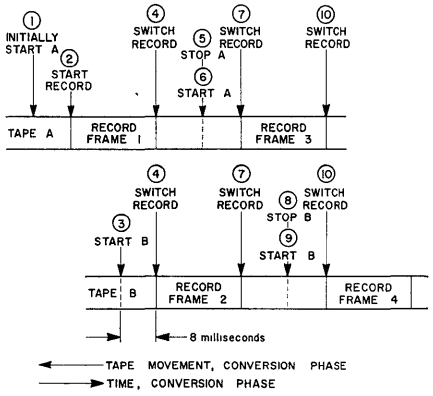
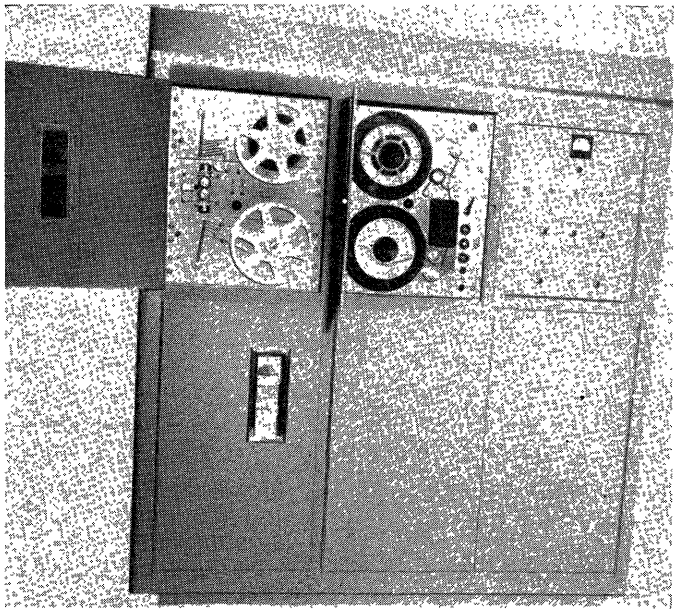


Fig. 6. Data arrangement on intermediate tapes. Note: Numbers in circles indicate sequence of events in time

Fig. 7. Front view, model 201 converter



sample is converted, a one is recorded in this spot along with the digital equivalent of the sample.

To this point in the discussion nothing has been said about inserting the gap and little has been said about the digital tape units. The system block diagram shows the information from the record register going to two intermediate tape units. The two units may be thought of as a large buffer storage. During the conversion process, called phase 1, the digital information is recorded in blocks on these units, alternately, until all of the analogue samples on the input tape have been converted. Then these tapes are played back, recording on a final tape unit *C*. This final unit has the information in a form required by the IBM *726*.

When the program is first started—the beginning of phase 1, the converted data are recorded on tape unit *A*. A preset channel counter is used to determine the number of samples to be recorded per record frame. The number of samples per record is selectable for a particular run and is at the discretion of the operator. It is dictated by the amount of memory space in the data processor available for 1-read reference to the tape units. A typical number might be 1,500 samples or 750 words. When information from the channel counter indicates that a gap is to be inserted in the final tape, the recording is switched from unit *A* to unit *B* without interruption, unit *B* having been started just before the record switch. Unit *A* is stopped shortly after the switch to *B*. When the next gap is to be inserted, the complementary action takes place, switching recording back to *A*. This process of switching back and forth continues until the input tape is exhausted of data.

The arrangement of the data on the two tape units after the first phase of conversion is completed as shown in Fig. 5. All of the odd record groups appear on tape unit *A* while the even groups appear on unit *B*. Although only four frames are shown in the drawing, the quantity of data is limited only by the output tape capacity. The total amount of information converted is dependent upon the quantity of analogue information on the input tape.

When the conversion processes are completed, the intermediate tape units contain all the digital information. This is in the same format as that required by the output tape and depicted in Fig. 2. As discussed, the record groups, which consist of many converted samples, are sequentially arranged in record groups on alternate intermediate tape units. It is now necessary to collate the record groups and insert the 1-inch gap while recording on the final unit. When phase 2, playback mode, is started, the intermediate unit last recorded starts in reverse. A gap-sensing device indicates when the end of a record group is reached. This immediately stops the running unit and starts the opposite unit. This then plays back until a gap is sensed on this second unit. It will be remembered that, during the conversion phase, when recording was switched from one unit to the other, there was a delay before stopping the first unit. This delay created a blank gap between groups shown in Fig. 6 and is of the correct amount to produce a 1-inch gap in the final tape.

The process of alternately playing from each intermediate unit continues until both units run out of information. When all digital information is read, both intermediate units run back and are automatically stopped by a photocell sensing a clear leader.

The intermediate units are played back in the opposite direction from which they were recorded. The digital information is read and simultaneously recorded on the final unit *C*. This implies that the final unit must run in the opposite direction from which it normally will be used in order that the data be intelligible. For this reason, the final tape unit *C* is made to run out blank tape during the conversion phase. When it is time for phase 2, the correct amount of tape is on the take-up reel and the tape rewinds as it is recorded upon.

Machine Operation

All controls necessary to operate the converter are brought out to a panel on the front of the cabinet. Here the alternating and direct voltages may be turned on. Both a-c and d-c blown fuse indicators are here.

Once the alternating current and direct current are turned on and proper warmup

time is allowed, the machine is ready for operation. The external Ampex unit must be loaded with the tape containing the sampled data; a clear leader of several feet should be threaded so that the tape unit is up to speed when the data are read. The final tape unit *C* is loaded with 1/2-inch tape.

Pushing the "start phase 1" button automatically starts the external input unit as well as all three tapes on the converter. When data arrive on the input tapes, intermediate tape unit *B* stops and converted data are recorded on unit *A*. Recording continues on *A* until the present-channel counter indicates that recording should be switched. Unit *B* starts and recording is switched to *B*, unit *A* stopping 180 milliseconds later. During this time unit *C* is running out tape, the amount that will be needed to hold all of the converted data on one tape.

The cycling process of recording on unit *A* or unit *B* continues until all of the PDM data are converted. At this time the operator pushes the "stop" button. All tape units stop, the intermediate units delayed only enough to record the last converted sample. Either tape unit may be recording when the stop button is pushed and it may be anywhere in a record group.

The operator now pushes the "start phase 2" button which starts the playback process and the recording on the final tape unit. Initially, tape unit *C* is the only one that starts. This runs for approximately 8 seconds producing a 1-foot end-of-file blank required by the IBM *726*. At the end of the 8-second period, the last intermediate unit recorded upon during phase 1 starts in the reverse direction. There is a 1-to-1 correspondence between the data on the intermediate units and that on the final unit. Digital information is read directly from the intermediate units and recorded on the final unit.

The playback continues from the unit started at the beginning of phase 2, say unit *B*, until a gap is sensed. Playback is immediately switched to unit *A* and unit *B* is stopped. Unit *A* now plays until the gap is sensed on this unit, and playback is switched again. This process continues until all of the record groups on the intermediate units are played back. The tape units are stopped automatically by photocell sensing. Tape unit *C* runs out an end-of-file gap and is then read for processing on the IBM *701*.

Conversion during phase 1 proceeds at the rate of 224 samples per second. The playback during phase 2 proceeds at twice this rate making an average conversion rate of approximately 150 samples per second. The amount of conversion is limited by the capacity of the output tape and for a 2,400-foot reel this is over 900,000 samples.

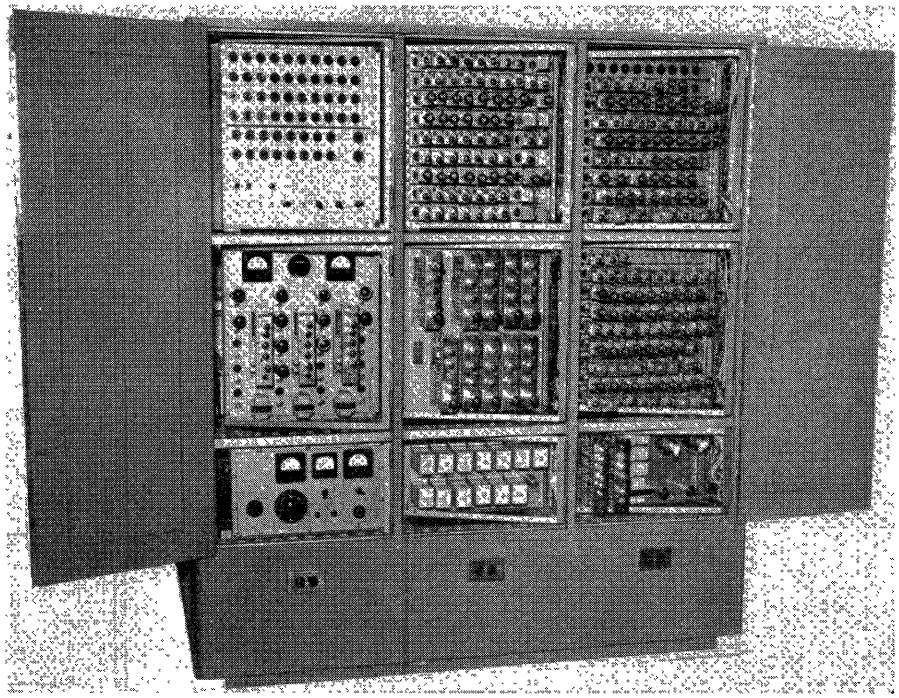


Fig. 8. Rear view, model 201 converter

Machine Design

A front and rear view of the converter is seen in Figs. 7 and 8 respectively. In the front view, the three tape units associated with the converter are shown. The two units on the left are the intermediate units and the center unit is the final unit *C* On the right upper is the control panel. The meter shows the output from the servo amplifier. Below it is an adjustment for removing any drift in the amplifier before operation is started. In the lower center and right are drawers containing the d-c power supplies.

The rear view shows the general approach of constructing all circuitry on plug-in units. This is designed as an aid in trouble-shooting and routine maintenance. The multitube units in the center are the playback amplifiers for the intermediate tape units. The panel in the lower left is for metering alternating and direct voltages and making adjustments in some direct voltages. These adjustments are used for marginal checking during routine maintenance. The panel in the upper left is a test panel where the majority of the plug-in units may be checked for operation independent of the machine proper.

There are 333 tubes in the machine of which a majority are *5670*'s used in trigger circuits and *7AK7*'s used in gated pulse amplifiers. The record tubes are *5687*-type. The converter also uses some 900 diodes of which the majority are *IN38A*'s.

The power supplies use germanium rectifiers and the total power consumption, a-c and d-c, is approximately 3.5 kva.

Conclusion

The foregoing is a discussion of a specific converter designed to do a specific task. The speed of conversion was decided to be 1/4 that of real time because of the short flights involved and because of the 10-digit accuracy desired. For this conversion rate the fastest trigger in the conversion counter operates at some 300 kc. Further, the output tape format was dictated by the data processor with which it was to be used.

The machine is versatile in that both the conversion rate and the output tape format may be modified to meet the needs of other data processing centers. Further it is possible to include editing features that will control the conversion process so that only sections of the PDM tape are converted. This feature would be particularly valuable where data from long flights are recorded but only certain sections are of interest.

Reference

1. A Pulse-Duration Modulated Data-Processing System, J. R. Lowe, J. P. Middlekauff. *AIEE Special Publication T-85*, 1956, pp. 53–7

An Improved Multichannel Drift-Stabilization System

P. G. PANTAZELOS

AT PRESENT, all but the smallest d-c analogue computers employ some method of drift stabilization to reduce drift at the output of the computing amplifiers. The method most often used is called chopper stabilization.[1] With this method, some drift-free gain is added to the forward loop of a d-c feedback amplifier. If the added drift-free gain is placed in the loop ahead of the primary sources of drift, the steady-state drift with stabilization is equal to the drift without stabilization divided by the amount of drift-free gain added. The required drift-free gain can be achieved with a chopper and an associated stabilization amplifier.

By the technique of single-channel chopper stabilization, excellent drift stability can be obtained, particularly with well-built and well-shielded choppers such as the Leeds and Northrup unit. Unfortunately, during a 5-year period, experience with a large number of choppers at the Dynamic Analysis and Control Laboratory (DACL) at the Massachusetts Institute of Technology has shown that maintenance of these choppers is necessary after their first year of operation, and, as the choppers become even older, they must be maintained more and more frequently. Also, the amplifiers associated with each chopper must be checked periodically. In a large installation with more than 200 computing amplifiers, such maintenance presents a problem. Another difficulty is that these choppers are bulky, and they, together with the added amplifiers, prevent the construction of smaller computing amplifiers. These disadvantages of single-channel chopper stabilization are avoided by using multichannel drift stabilization.

Multichannel drift stabilization[2] is an extension of the chopper-stabilization technique. In the multichannel system as shown in Fig. 1, one stabilization system is time-shared by a group of d-c amplifiers, thus effecting a reduction in equipment size, cost, and maintenance. Unfortunately, the multichannel systems in use at the time that work began on the DACL system were inferior in some respects when compared with single-channel systems. For example, sufficient gain could not be achieved in the common stabilization amplifier to eliminate the need for a balancing adjustment in the computing amplifiers and in the stabilization amplifier. Crosstalk existed between channels, particularly if one channel was badly overloaded. Also, none of the systems at that time incorporated overload indicators that operated at the incidence of an overload and remained on after the overload until reset by the problem operator.

The design techniques described in this paper extend the usefulness of a multichannel system by eliminating some of these defects. The problem of eliminating crosstalk between computing positions without sacrifice of stabilization gain has been solved by the use of intermittent feedback and self-biased diodes, and the problem of providing a useful overload indicator has been solved by sampling the size of the signal in the stabilization system and by igniting gas tubes when an overload exists.

These techniques are now incorporated in a multichannel system in a small computer at the DACL. The computer has 30 d-c computing positions all of which are drift-stabilized with one commutator and one stabilization amplifier. The computing amplifiers require about 1/6 the volume of the older, single-channel chopper-stabilized amplifiers. The maintenance required is negligible when compared with the older units.

The response speed and the transient characteristics of this multichannel system as well as a statistical evaluation of the drift encountered in the DACL computing positions have been described elsewhere.[3]

Theory of Operation

In the multichannel system of Fig. 1, one stabilization amplifier is used to stabilize the 30 d-c amplifiers with the aid of a single commutator that samples the summing-point voltage of each amplifier in turn. Any direct voltage present at the summing point of an amplifier is applied to the stabilization amplifier as a pulse occurring at the repetition frequency of the commutator. These pulses, after being amplified, essentially without drift, and inverted in the stabilization amplifier, are channeled to the same d-c amplifier by the output section of the commutator and applied as a stabilization voltage through a smoothing filter. The computing amplifiers are conventional high-gain d-c amplifiers.

The stabilization circuit is used also for overload indication. When any d-c amplifier in the computer is overloaded, a large d-c error voltage appears at its summing point. This voltage is sampled by the input section of the stabilization switch, and the resulting pulses are amplified by the stabilization amplifier (see Fig. 1). Whenever the pulses in the stabilization amplifier exceed a predetermined level, they trigger a monostable multivibrator. The large, positive output pulses from the multivibrator override the normal output of the stabilization amplifier and are applied through the output section of the commutator to the overloaded d-c amplifier. In the d-c amplifier, where the large size of these pulses distinguishes them from a normal stabilization output, they are detected and used to trigger an overload indicator.

The Stabilization and Overload-Indication Unit

Fig. 2 is a schematic diagram of the stabilization and overload-indication unit circuitry. Tubes *V1*, *V2*, *V3*, and the first section of *V4* constitute the amplifier section. This is a conventional d-c amplifier. To eliminate drift in this amplifier, its gain is reduced between pulses to less than one by briefly closing a feedback path, as shown in Fig. 3. Each time the input grid of the stabilization amplifier is grounded, the charge on capacitor *C* in Fig. 3 is adjusted to the value required to keep the quiescent amplifier output very nearly at ground potential. No energy-storage elements appear in the stabilization-amplifier circuit during the time it receives signals from the computing-amplifier error points. The rotor of the input section of the commutator is wider than the rotor of the output section in order to ensure that the input be grounded whenever the stabilization-amplifier feedback loop is closed; therefore, crosstalk between adjacent

P. G. PANTAZELOS is with the Massachusetts Institute of Technology, Cambridge, Mass.

This paper is based on work done at the Dynamic Analysis and Control Laboratory, under Air Force Contract No. *AF 33(616)-2263* with the Division of Industrial Cooperation of the Massachusetts Institute of Technology.

ERRATA

Illustrations for the paper by P. G. Pantazelos entitled "An Improved Multichannel Drift Stabilization System" published on pages 62–4 of Special Publication T–85 "Proceedings of the Western Joint Computer Conference" held February 7–9, 1956 in San Francisco, Calif.

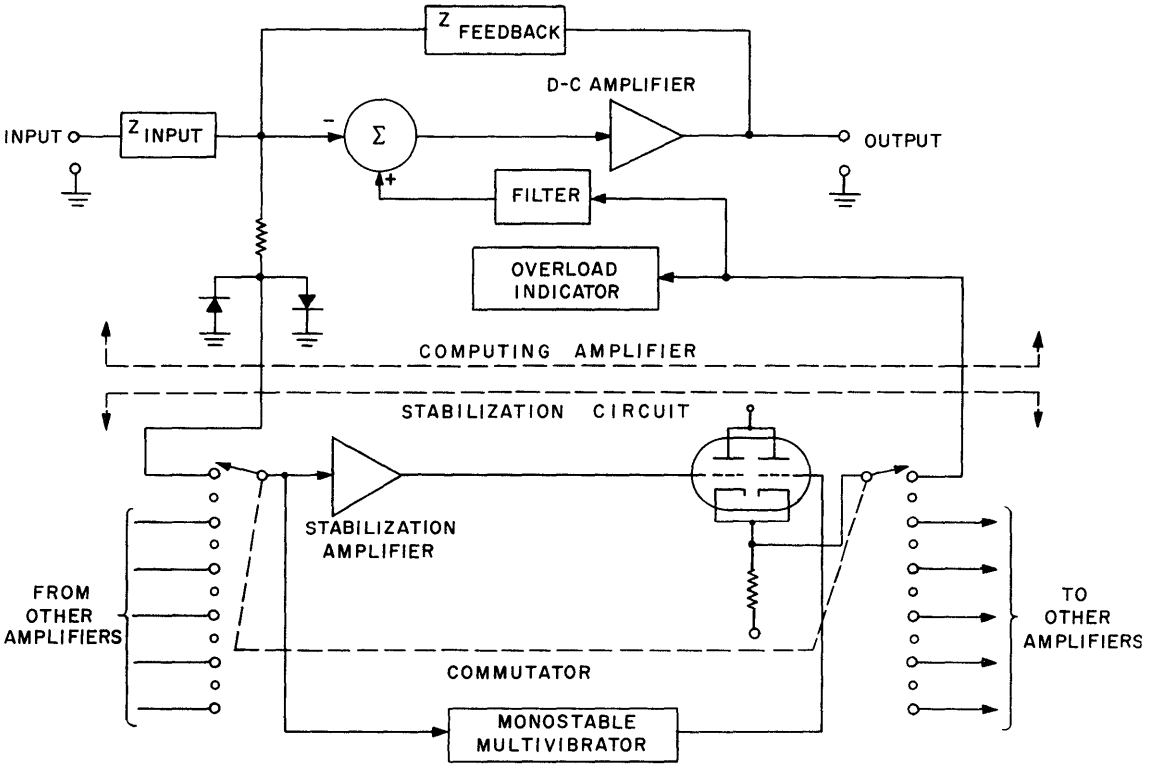


Fig. 1. Stabilization circuit with typical d-c amplifier

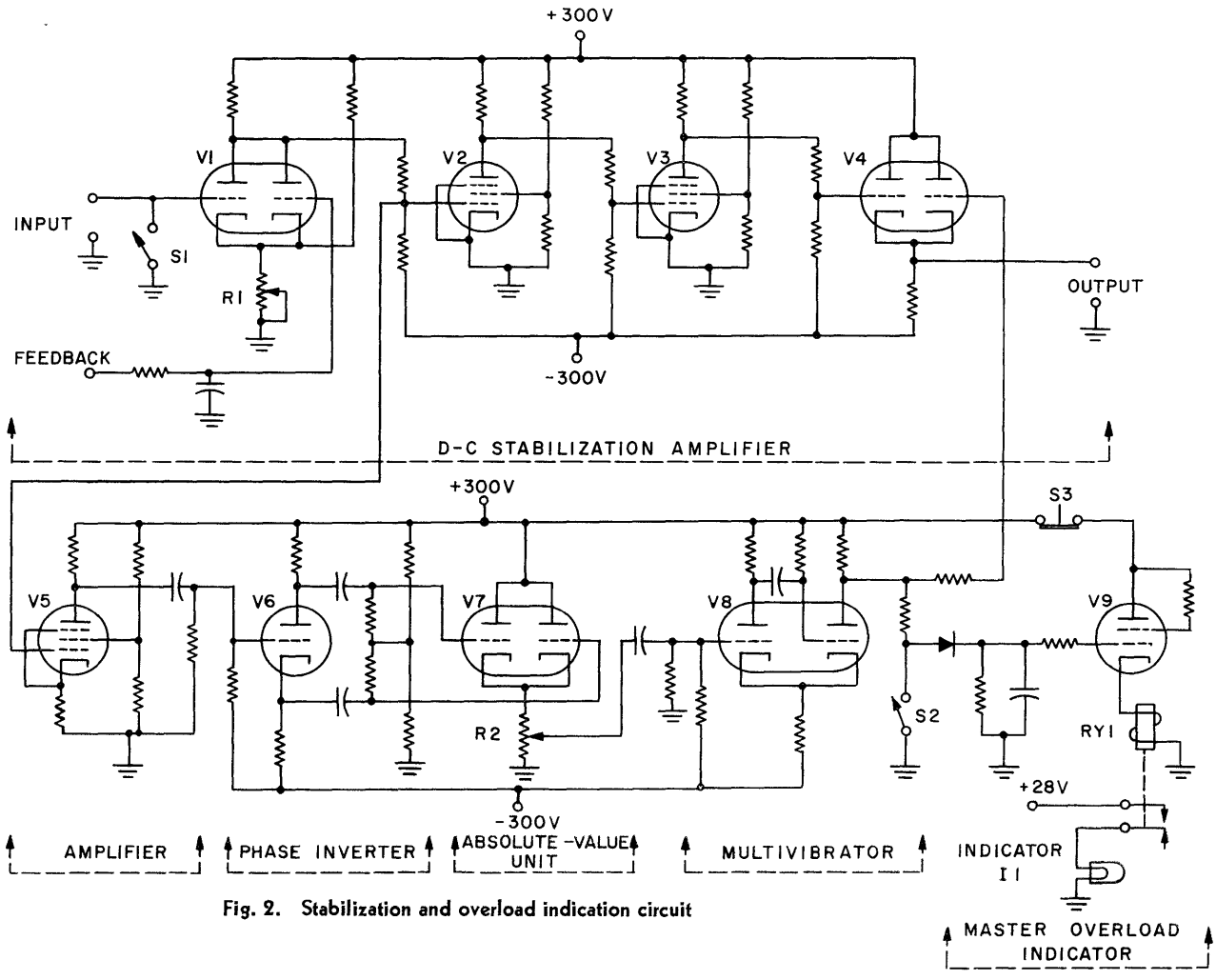


Fig. 2. Stabilization and overload indication circuit

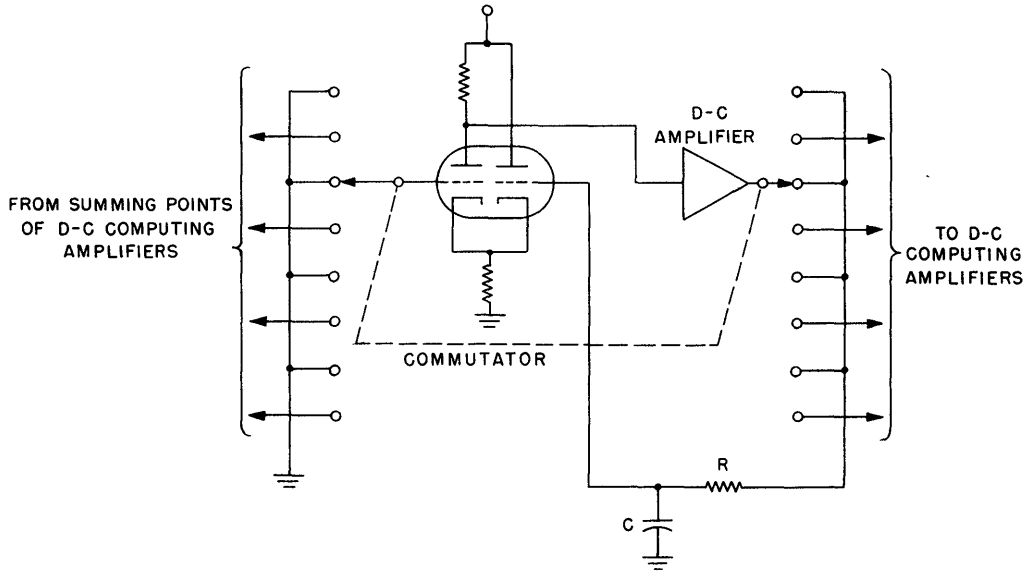


Fig. 3. Simplified stabilization on amplifier circuit

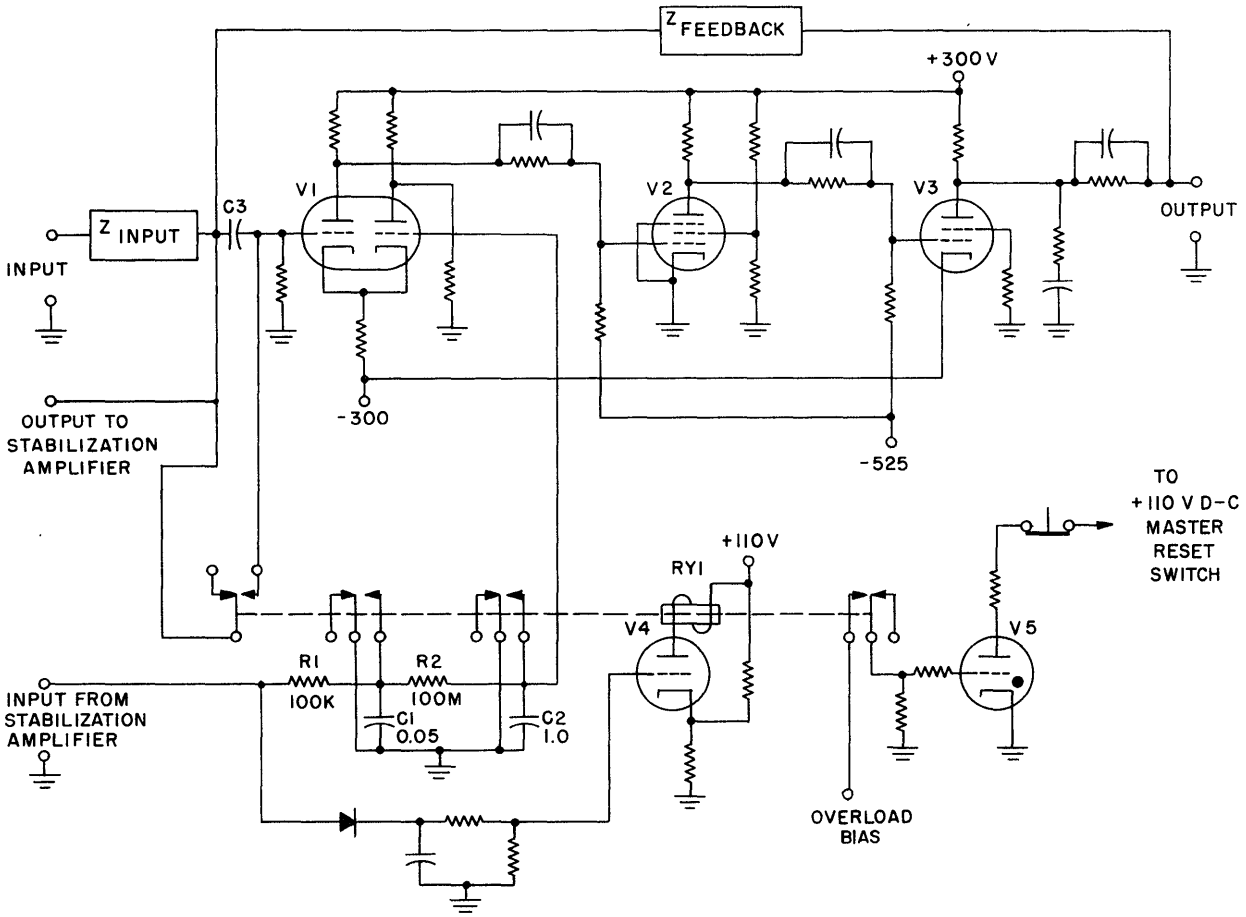


Fig. 4. Computing-amplifier unit

computing channels as the commutator rotates is not possible in the amplifier. The commutator used in this system is built by the Applied Science Corporation of Princeton, N. J., and is designed specifically for use in drift-stabilization circuits. It is a 60-position single-pole 2-deck switch that rotates approximately four times per second.

The centering control *R1* in Fig. 2 is provided to allow the output of the amplifier to be adjusted to ground potential when the input is grounded by switch *S1*. This adjustment, which allows the output to be set easily to within 100 millivolts of ground, is very stable and need be made only when a tube has been replaced in the stabilization amplifier or after a component failure has been repaired. The stabilization amplifier has a gain of approximately 2,000.

Overload-Indication Circuitry

The remainder of the circuitry in Fig. 2 is used in the following way for overload indication:

1. Overload-signal amplifier *V5* amplifies the pulses at the grid of *V2*.

2. Phase-inverter *V6* makes a push-pull signal of the output of the amplifier *V5*.

3. Tube *V7* acts as an absolute-value device in that it inverts the sign of negative input pulses while not affecting positive input pulses.

4. Potentiometer *R2* sets the trigger level of the multivibrator *V8*. It does this by setting the magnitude of the pulses so that a predetermined overload voltage at the error point of any d-c amplifier produces a pulse just large enough to trigger the multivibrator.

5. The two sections of *V8* with their associated circuitry constitute a monostable multivibrator. This multivibrator is triggered by any large positive pulse applied at its input. When idle, the first section of the tube is cut off, and the second section is conducting with a plate voltage of −100 volts. When triggered, the first stage conducts, and the second stage is cut off. The second-stage plate voltage then is equal to the supply voltage of +300 volts.

6. The large pulses from the output of the multivibrator are applied to one grid of *V4* where their magnitude is sufficient to override any input to the other grid. Thus, the output of *V4* always is determined by the multivibrator output whenever the multivibrator is triggered. When the multivibrator is idle, the second section of *V4* is cut off and inoperative, and the first section operates as a cathode follower.

7. Tube *V9* is triggered by the multivibrator pulses, and it in turn energizes the relay *RY1*. The relay contacts are used to turn on a master overload indicator *I1*. Tube *V9* can be extinguished by removing its plate voltage with switch *S3*. The existing status of tube *V9* (on or off) can be retained independently of multivibrator pulses by grounding the overload-bias lead with *S2*. The master overload indicator is ignited by an overload in any of the 30 computing amplifiers.

Voltage-Limiting Network

As shown in Fig. 1, back-to-back silicon diodes are used to limit the voltage swing at the input to the stabilization switch. Voltage limiting is necessary to prevent crosstalk between computing channels through the leakage in the insulation between contacts of the switch. Silicon diodes are used because their forward resistance is high until approximately 0.5 volt is applied in the forward direction. Thus, they have essentially a built-in bias of 0.5 volt, and no bias supply is needed. Conventional germanium diodes and a bias supply cannot be used because their back resistance is not high enough to prevent current flow from the bias supply when the diodes are not conducting. Any current flow at this point can reach the computing-amplifier summing point and can cause an offset at the output of the computing amplifier.

Stabilization and Overload Circuitry in the Computing Amplifier

Fig. 4 shows the circuit of a computing-amplifier unit. The unit contains a conventional high-gain d-c amplifier as well as some overload-indication and drift-stabilization circuitry. The circuitry used for overload indication and for filtering the stabilization-amplifier output is shown in detail.

Stabilization Filter

As shown in Fig. 1, the stabilization commutator and amplifier sample the summing-point voltage of each amplifier, and deliver output pulses with peak values approximately 2,000 times the summing-point voltage. The pulses occur every 1/4 second and are 4 milliseconds long. Because the output stage of the stabilization system is a cathode follower and because a mechanical switch is used, the driving impedance is low when a pulse occurs and high between pulses. Consequently, *R1* and *C1* act as a holding circuit, and a steady voltage equal to approximately the pulse magnitude is developed across the capacitor *C1* in Fig. 4. (The time constant *R1* times *C1* is small.) The filter section *R2* and *C2* further smooths the voltage appearing across *C1*. The voltage applied to the second grid of the differential amplifier *V1* is at very low frequencies nearly an exact replica of the summing-point voltage multiplied by approximately 2,000. A large voltage applied across capacitors *C1* and *C2* in the stabilization filter causes dielectric absorption. The relay *RY1* is used to prevent this large voltage from occurring by short-circuiting the capacitors when an overload occurs. The d-c amplifier input coupling capacitor *C3* likewise must be short-circuited for the same reason. Short-circuiting these capacitors also reduces overload recovery time. Overload recovery time is 5 seconds or less, depending on the type of overload.

Overload-Indication Circuit

As already described, when an overload occurs in a d-c amplifier, a large voltage appears at the d-c amplifier summing point. This voltage is sampled by the stabilization switch, and the resulting pulses are amplified in the stabilization-amplifier unit shown in Fig. 2. The presence of pulses larger than a predetermined value is sensed in the stabilization amplifier, and a triggered multivibrator produces very large positive output pulses. These large output pulses are directed by the output section of the stabilization switch to the overloaded d-c amplifier.

In the d-c amplifier, the large pulses trigger relay *RY1* which in turn short-circuits the proper capacitors to prevent dielectric absorption. Also, the relay ignites a *1C21* thyratron (*V5*) whose glow provides visual indication of the overload. When the overload indicator (*V5*) is triggered, it remains ignited until manually reset either by switch *S1* or by a master reset switch which simultaneously removes the plate voltage from tube *V10* in every amplifier.

To clamp the overload indicators so that their status (on or off) at any time can be retained independently of the occurrence of further overloads, the d-c overload bias voltage is removed. When the voltage is removed, the overload relay (*RY1*) is unable either to ignite or to extinguish the indicator tube (*V5*).

Conclusions

A multichannel drift-stabilization system with high gain and negligible crosstalk has been attained. With proper care taken to prevent pickup and to provide good signal grounds, the degree of drift stabilization obtained with a multichannel system is equivalent to that of a single-channel system. The following specifications are met by this system:

1. The stabilization channel gain is high enough (approximately 2,000) to reduce drift and offset referred to the summing point of the computing amplifiers to less than 1 millivolt. No balancing control is incorporated in the computing amplifiers.

2. Virtually no crosstalk occurs between computing positions even if one amplifier is overloaded and its summing point is at a high potential with respect to ground.

3. An overload indication occurs at each computing position when that position overloads and at a master position when any amplifier overloads. The indicators, triggered, stay ignited until manually reset.

To the author's knowledge, a combination of high stabilization gain and no crosstalk has previously not been attained. Furthermore, this paper describes the first multichannel system with an overload indication that remains ignited after being triggered. The usefulness of this feature in finding and correcting overloads has been demonstrated during operation of the computer at the DACL. This overload system is also useful in preventing dielectric absorption in the stabilization filter capacitors and input coupling capacitor, and in reducing overload recovery time.

References

1. Stabilization of Wide-Band Direct-Current Amplifiers for Zero and Gain, **E. A. Goldberg.** *RCA Review*, Princeton, N. J., vol 11, June 1950, pp 296–300.

2. Time-Shared Amplifier Stabilizes Computers, **D. W. Slaughter.** *Electronics*, New York, N Y, vol. 27, April 1954, pp 188–90.

3. Analysis of a Multichannel Drift-Stabilization System, **P. G. Pantazelos.** M. S. thesis, Massachusetts Institute of Technology, Cambridge, Mass, January 1955.

Combined Analogue and Digital Computing Techniques for the Solution of Differential Equations

P. A. HURNEY, JR.

ONE of the difficulties in the use of an analogue computer for the solution of ordinary differential equations involving variable coefficients is its relative inability to perform certain multiplications rapidly and accurately. In instances where variables must be multiplied by a function of another variable, this difficulty is particularly apparent. This paper describes how a digitally stored table of functions may be used with an analogue computer to solve this general class of ordinary differential equations.

In the preparation of the machine for the problem, the functions of the variables are read into a magnetic drum by employing conventional digital techniques. During operation, the analogue inputs are compared continuously with their digital equivalents that are on the drum, and the digital functions of the input are read and digitally stored between readings. Digital-analogue multipliers are used to multiply the digital output by the appropriate analogue output. The advantages of the computer are (1) the shortness of the machine-preparation time, (2) the operating speeds which are faster than those of electromechanical multipliers using potentiometers or resolvers, and (3) the accuracy which is greater than that obtainable with electronic multipliers and function generators.

P. A. Hurney, Jr., Hycon Eastern Inc., Cambridge, Mass., was formerly with the Dynamic Analysis and Control Laboratory, Massachusetts Institute of Technology, Cambridge, Mass.

Introduction

The first part of this paper discusses a few of the limitations of analogue computers used in the solution of ordinary differential equations. Next are described several types of digital computers which are designed to perform certain operations more efficiently than the equivalent group of analogue components. Finally a specific analogue-digital computer is described which is currently under development at the Dynamic Analysis and Control Laboratory of the Massachusetts Institute of Technology.

Limitations of Analogue Computers

Analogue computers are capable of performing certain mathematical operations with rapidity and high accuracy. Some of these operations are addition, subtraction, multiplication by a fixed coefficient, and integration with respect to time. In general these operations permit solutions to problems involving ordinary differential equations at speeds many times that of the physical system under study and with accuracies often more than adequate for engineering purposes.

A second class of operations performed on analogue computers which are either less accurate or have slower solution times is multiplications of two or more variables. In general, two methods are used for the multiplication of independent variables. One is the electronic multiplier which may have a useful bandwidth of several hundred cycles but long-time accuracies seldom better than 0.25 per cent. A second method of multiplication is the servomultiplier. This multiplier is capable of static accuracies better than ten times that obtainable with the electronic multiplier but one input variable is restricted to a band width of several cycles per second if good accuracy is to be maintained. In addition, there are limitations on the maximum rate of change of the input.

A third class of operations which are restricted both in accuracy or bandwidth is the generation and multiplication of function of a variable. Electronic function generators have approximately the same bandwidth as the electronic multipliers but long-time stabilities are seldom better than 0.25 per cent. Additional errors are introduced in approximating complex functions by a series of straight lines. As a result, the product of one variable by a function of a second variable using electronic multipliers and function generators may have static errors as large as 0.5 per cent.

Servomultipliers with tapped potentiometers may also be used to obtain prod-

ucts of the function of a variable. The tapped potentiometers are loaded with appropriate resistors to make the potentiometer conform to the desired function. The static accuracy and repeatability of the servo function generator are excellent but, as in the case of servomultipliers, the bandwidth is restricted to several cycles per second. In addition the tapped potentiometer also approximates the function with straight line segments.

For certain classes of nonlinear functions special techniques are used. The trigonometric functions are common to problems. The most widely used method for generation of sines and cosines is the servoresolver. This unit is similar to the servomultiplier except that special nonlinear multiplier potentiometers are used. These potentiometers generate sine and cosine functions to good accuracies. However, for static accuracies better than 0.1 per cent the bandwidth of the servoresolver is even less than the servomultiplier. This is because accurate sine-cosine potentiometers are large, having high inertias and friction levels.

One device used in the resolution of vectors in analogue computers is the a-c resolver. These electromagnetic units have small inertias, low friction levels, high accuracies, and infinite resolution. As a result a servo using a-c resolvers has a bandwidth superior to the servoresolver using potentiometers. However, a serious limitation in the use of the a-c resolver is the a-c carrier needed in the operation of these units. The a-c carrier system generates many special problems. One difficulty is in the combined use of a-c and d-c computing equipment with the present lack of accurate high-speed modulation and demodulation apparatus. It is often desirable in a large problem to perform a portion of the computation on d-c analogue equipment and the trigonometric computation on the a-c section of the computer. It is not economical to use only an a-c computer in the solution of many problems because of the complexity of a-c equipment needed to perform many computations which are basic to the d-c analogue computer.

The foregoing brief and incomplete description indicates some of the limitations of the modern analogue computer. There is a need for better methods for the generation of functions of variables, multiplication by functions of variables, in particular, the trigonometric functions, and, to a lesser extent, multiplication of two variables. The need for faster, accurate computation methods arises from two sources. First, complex systems involving a number of independent variables may be studied rapidly to determine effects of parameter variation. Secondly, simulation of systems involving physical components requires the computer to operate in real time. For example, the study of a guidance and control system for a missile involves simulation in 3-dimensional space. This requires a number of geometric resolutions and multiplication by arbitrary functions. A study of a complete missile system may involve hundreds of solutions to include the many possible initial conditions and parameter variations. Very often, components of the guidance and control system are included in a final study of an over-all system. The section of the computer simulating the aerodynamic and space geometry must be capable of operation in real time to be able to test the system components. In the case of missiles where the actual flight tests are expensive, it is desirable to be able to check the effect of the operation of the missile components on the over-all system. The accuracy of computation of space geometry is very important in a problem of this type. Since these computations are mathematical in nature rather than a simulation of physical equipment, small errors in geometry may result in large over-all problem errors.

Digital Auxiliaries to Analogue Computers

The digital computer which may be used with analogue computers is naturally a specialized device and may differ in many respects from conventional equipment. Input-output conversion is needed which will allow the digital machine to operate simultaneously with the analogue computer. For convenience in programming, the digital equipment must be capable of operating much as an analogue element in the complete system. The digital computer must operate at a time scale compatible with the analogue computer so that the lags in the digital computer will not effect the accuracy of the over-all problem solution.

Several types of digital computers are useful in the applications described. The first of these is a computer consisting primarily of a multiplier. With a multiplier and suitable programming, problems involving multiplication of variables, trigonometric function generation, reciprocals, and powers, and, to a lesser extent, arbitrary functions may be solved. The second type of digital computer consists of a storage element in which is stored an input and a function of that input. This type of digital computer may be called a function table. The stored function of the input may be any function of a single variable. The function table may have stored trigonometric, inverse trigonometric, or any arbitrary relationship between the input and output. With external multipliers, products of variables involving any of the stored functions are obtainable.

The basic difference between the two digital methods described is that in the case of the multiplier or polynomial generator the relationship between the input and output is stored as a series of instructions in the program. The function table has stored all the relationships between the input and output. Each method has advantages and the choice of either depends upon the type of problems to be solved.

The polynomial generator is primarily a serial type of computer, i.e.; for one multiplier only one operation or product may be computed at one time. However, if the multiplier is fast enough, the unit may be time-shared among many inputs. The analogue input is first converted into a digital number by a time-shared analogue-to-digital converter (ADCON). Next the operation is performed on the input number according to the instructions in the problem control. Operations involving trigonometric functions are computed with several successive multiplications and additions until the required accuracy is obtained. Several registers are generally needed for internal storage. Arbitrary functions may be computed from one or more polynomial approximations, although very often this type of function is very difficult to program to the required accuracy. In addition, programming time for even simple arbitrary functions may be excessive. More accurate and convenient function generation may often be obtained using conventional electronic analogue function generators. However, if a problem involves a number of trigonometric functions and multiplications, the polynomial generator provides a fast accurate method for performing these computations. The output digital number is converted into analogue form by the digital-to-analogue converter (DACON) which is time-shared among the various outputs. The output information is stored in an analogue storage device between successive digital solutions.

The function table computer which involves digital storage of each value of the input and the function of the input offers a method of computing any function of a single input variable. It is this type of computer which is under development at

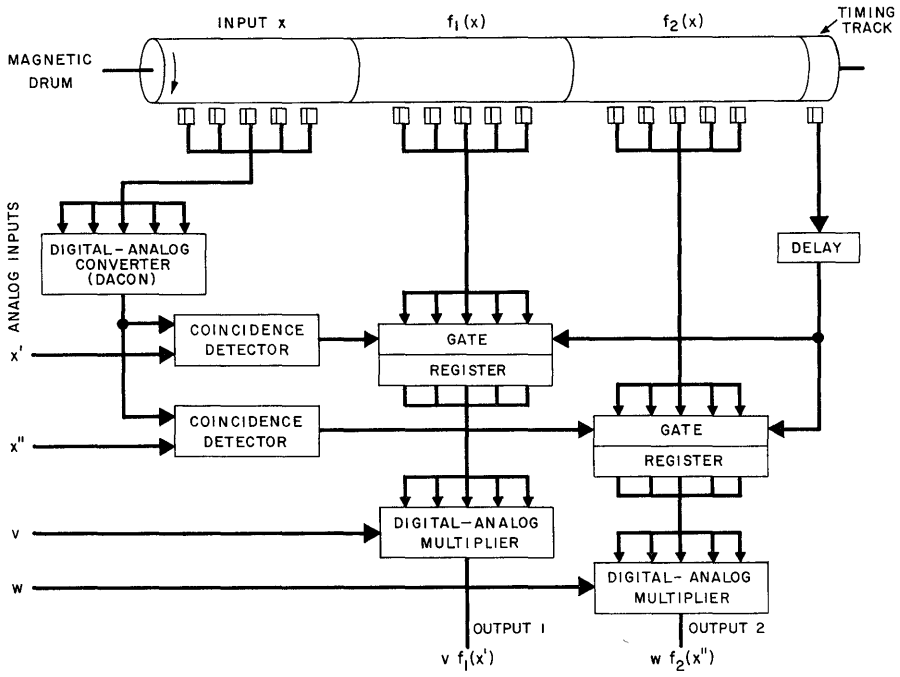


Fig. 1. Digital function table

the Dynamic Analysis and Control Laboratory (DACL) at the Massachusetts Institute of Technology. The function table computer, together with a digital-analogue multiplier, provides a general-purpose auxiliary to an analogue computer which results in an extremely flexible over-all machine. Not only does the system allow rapid and accurate solution to many type of problems, but it also results in a simple programming procedure.

The system described in the next part of this paper will allow the solution of large-scale problems. It may be used in solving problems involving resolutions, function generation, and multiplication. The system bandwidth should be higher than conventional d-c electromechanical computing elements, and compatible with most electronic analogue computers. Over-all reliability should be greater than an equivalent electromechanical system because of the elimination of the servos with their associated potentiometers, gearing, and motors.

Description of the Function Table Computer

The basic element in the system is the storage device, a magnetic drum. On this drum are stored columns of tabulated numbers similar, for instance, to a table of sines. In the first column are the values of the angle, and adjacent to this, a column with the values of the sines of these angles. On the magnetic drum, not one, but a set of desired functions may be stored. Each function shares a single column of nondimensionalized inputs. As the drum is rotated, the input column is scanned, and when the desired value of input is reached, a function of the input is read from the proper column and stored. Fig. 1 is a block diagram of a drum and associated equipment. In this example the drum has three columns of numbers. The first is the input x, the second $f_1(x)$, and the third $f_2(x)$. As the drum rotates, the output of the magnetic reading heads which scan the drum read a particular value of x, $f_1(x)$, and $f_2(x)$.

The digital numbers representing the input are converted by DACON into an analogue voltage. This voltage is fed into the coincidence detector along with the analogue input x'. When the analogue voltage representing the digital drum input equals the analogue input x' the coincidence detector opens the gate which reads the instantaneous value of $f_1(x')$ on the drum and puts it into a storage register. When the second analogue input x'' equals the drum input, in like manner, $f_2(x'')$ is put into another storage register. If the products of these functions with other input variables are desired, the outputs of these registers are fed into digital-analogue multipliers along with the analogue variables v and w, resulting in the products

$$\text{Output}_1 = vf_1(x')$$
$$\text{Output}_2 = wf_2(x'')$$

The system may be expanded in an obvious manner to obtain additional products and functions of other input variables.

The basic elements of the system are:

1. Magnetic drum.
2. Digital-to-analogue converter.
3. Coincidence detector.
4. Gate.
5. Storage register.
6. Digital-analogue multiplier.

The Magnetic Drum

The magnetic drum is the information storage unit and must be large enough to hold the required number of functions. It must have around its periphery enough values of any particular function to represent this function to the desired precision. The speed of the drum should be as high as possible in order to obtain the shortest possible sampling interval, which results in the widest bandwidth. Presuming that a basic precision of 0.1 per cent of full scale is desired, the input which is a linearly increasing ramp must be divided into 1,000 parts for the positive voltage range and 1,000 parts for the negative voltage range or a total of 2,000 parts. Therefore, there should be 2,000 different digitally coded values of the ramp around the periphery of the drum. Since the ramp is coded in binary form, an 11-bit word is the minimum necessary if 2,000 different values are to be represented (2 inches = 2,048). Therefore the input ramp consists of 2,000 11-bit words equally spaced around the drum. For each value of the input there is a corresponding value for each function. The speed of the drum determines the number of times per second that a new value of function appears at the output of the system. At present, a speed of approximately 125 revolutions per second appears feasible. In this case the basic output frequency is 125 cycles per second and the corresponding sampling interval is 8 milliseconds. The equivalent lag is 4 milliseconds because the error is zero at each sampling time and builds up to a maximum just before the next sample. Phase compensation circuits may be used on the input analogue signal. This compensation effectively cancels the drum lag and results in an equivalent phase lag of the over-all system of less than 1 degree up to 10 cycles per second. In addition, there are no restrictions on the maximum rate of change of the input variable.

Information is read into and out of the drum through magnetic heads and read-write amplifiers. The "write" equipment may be as simple as a keyboard, but for convenience in programming, punched cards or tape should be used. The "read" amplifiers must be capable of driving several gates in parallel.

A typical drum would have provision for the storage of seven functions plus the input ramp function. In addition several tracks are used as timing tracks. On these tracks are stored timing pulses which are used to synchronize the write and read gates.

The Digital-Analogue Converter (DACON)

The digital-analogue converter is an electronic device which converts paralleled binary-coded information to analogue form. Basically the unit consists of a series of accurate current switches driving a resistance ladder. As many current switches are needed as there are bits in the coded information. A converter of this type can be built to respond in less than a microsecond; in this system several microseconds are available. The drum rotates at about 125 cycles per second and there are 2,000 words in each revolution; therefore, a new word occurs about every 4 microseconds. Approximately half this time would be available to operate the DACON associated with the drum input circuitry.

The input DACON is isolated from the coincidence detectors by an amplifier which prevents loading of the converter by the detectors. The amplifier must have a wide bandwidth and an accurate gain. Only one such amplifier of this type is needed.

The Coincidence Detector

The coincidence detector is essentially a zero-sensing detector. When the analogue input to the system and the analogue signal from the input DACON have the same value, the output of the detector changes sign rapidly. If the output of the DACON is less than the analogue input signal the output of the detector is at a negative potential. As the ramp signal from the drum increases, the output from the DACON increases in a positive direction until the voltages are equal. At this time the sign of the output of the coincidence detector reverses and a positive gating signal is generated.

The Gate

The gate is used to control the transfer of information from the drum into the storage register. The gates are controlled by signals from the coincidence detector and from the timing unit. See Fig. 1. The timing signals from the drum are used to prevent the gating of a signal into the register if the number on the drum is changing value. The timing signals are spaced so that the gate may be opened only when the output from the coincidence detector and the input DACON have reached a steady state value.

The gate is associated with the register and therefore is logically a part of the register. However, it is discussed separately here because of its function in the operation of the system.

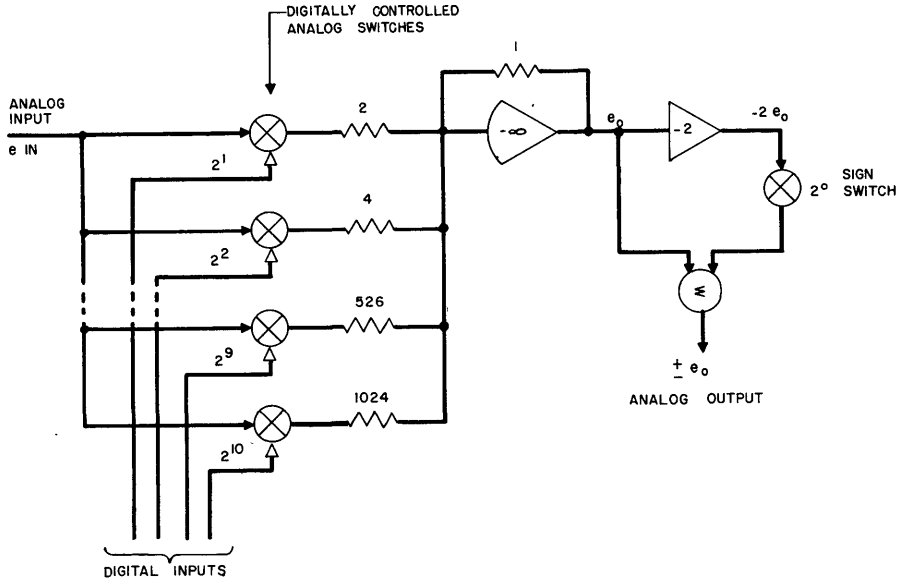


Fig. 2. Digital-analogue multiplier

The Storage Register

The function of the storage register is to store information from the drum between samples. The input to the register is a binary coded parallel signal from the drum which is switched into the register by the gates from the readout amplifiers. The register is reset just before the new information is to be stored. The reset function is accomplished by the timing circuit. When the gate associated with a particular register opens, information is supplied to each of the flip-flop elements in the register simultaneously. The presence of a pulse on any line sets the flip-flop associated with that line. Because of the low repetition rate of the registers, nondestructive readout magnetic cores may be used in the construction of the register.

Digital-Analogue Multiplier

The digital-analogue multiplier is a unit which has one analogue and one digital input and an analogue output which is a product of the two inputs. Basically the multiplier consists of ten parallel conductances arranged in binary ratios. The conductances are switched into the circuit according to commands from the register. The ten switches and ten conductances can be made to have 1,028 different values of total conductance. These conductances form the input resistor to a feedback amplifier resulting in an amplifier which has a gain between 0 and 1 depending upon the condition of the switches. See Fig. 2 for a simplified schematic of the multiplier. The 11th bit from the register controls the sign of the digital multiplication. It operates an electronic switch at the output of the multiplier which changes the sign of the multiplication. Since the analogue input may have either sign the over-all multiplication is in four quadrants.

Since the digital input to the multiplier changes in a stepwise manner the gain of the unit also changes in increments. Therefore the output wave form contains components of the sampling frequency. The wave form is similar to that obtained from a servomultiplier which uses a wire-wound potentiometer. The switching time of the digital-analogue multiplier which has been developed at Dynamic Analysis and Control Laboratory has a switching times of a few microseconds and a change in the digital code does not introduce spurious signals into the output. The output wave form therefore has small components of the switching frequency which are easily filtered.

Operation of Function Table

The computer would consist of the magnetic drum, the input digital-to-analogue converter, one coincidence detector for each input signal, a gate and register for each desired function, and a digital-analogue multiplier for each product. The analogue inputs and outputs would be handled on the analogue computer patchboard in the standard manner.

The interconnection of the digital components could be accomplished with multiconductor patching or switching within the digital computer itself. The actual computer setup time for interconnection of units would be approximately the same as a conventional machine.

The information to be stored on the drum would best be handled with punched cards. Functions such as sines or cosines could be calculated and punched on cards using standard digital computers. Empirical function could be either punched into cards using a manual keyboard, or if desired, curve followers with digital outputs could be used to encode the desired information automatically. Once the set of data has been placed on the cards the information is read into the drum using well known techniques. If standard equipment were used each function would require approximately 10 minutes to be read into the drum. Once the function is on cards it could be stored and would be available at any future time for use on the machine.

Conclusions

The function table computer together with a suitable analogue computer would be a machine capable of solving many types of problems faster than is now possible with analogue computers. The accuracy of the proposed system would be comparable with present computers. Programming is simplified because of the ease of computing with any function of a variable. The combined computer should be easier to maintain because of the absence of servomultipliers and resolvers. Checking the operation of the digital function table may be accomplished by standard analogue techniques. At present the machine is feasible because it uses available equipment and techniques. Future development in components may reduce the cost of the computer and enhance its usefulness.

An Experimental Monitoring Routine for the IBM 705

H. V. MEEK

THERE is a pressing need for aid in the knotty business of checking a code for a large digital computer. What better instrument is to be used than the computer itself? Because an automonitoring feature is absent from the circuitry of most computers, the monitoring operation should be programmed and will furnish an effective means of detecting many coding errors.

A routine for the International Business Machines Corporation (IBM) *705* has been prepared which monitors the instructions of a code being tested, and gives a complete history of the computer action as a result of that code.

Before the format of the output of the monitoring routine is discussed in detail, a brief description of the *705* internal nature should be given. Also, some of the questions which arose during the monitor planning phase and the decisions which were made should be mentioned.

The *705* is a high-speed stored-program electronic data-processing machine. Its main memory is either 20,000 or 40,000 character positions of magnetic core storage. Each character consists of seven binary digits; a 4-bit "numeric" part, a 2-bit "zone" part, and one check bit, and may be a decimal digit, a letter, a punctuation mark, or a special symbol. A 256-position accumulator, and 15 auxiliary storage units which together comprise 256 positions, provide temporary working storage for arithmetic and logical operations. Instructions have five characters each: one operation-code character and a 4-decimal digit address part. The thousands position of the address part must be appropriately zoned to specify a location with address greater than 9999. If one of the 15 auxiliary storage units is to be specified, the tens and hundreds positions of the address part are zoned so that the four zone bits together form a binary number equal to the address of the desired auxiliary storage. There is no fixed word length in the *705;* the length of an operand depends on the current condition of the specified accumulator or storage, the type of instruction being executed, and/or the characters of the operand and its adjacent fields.

Now then, what exactly should the monitor tell about a code? It was decided that each executed instruction, its location, its interpretation, the operand it calls for, and the results it produces would be given. The instruction interpretation consists of a 3-character mnemonic operation symbol, a 5-decimal digit address part, and a 2-digit reference to the accumulator or auxiliary storage unit. Each operand, including its length and sign, each change in main memory content, and the length, content, and sign of the accumulator or an auxiliary storage unit after each reference will be noted alongside the corresponding instruction. Flags will call attention to coding irregularities, such as specifying an auxiliary storage unit for a "multiply" instruction, or allowing the character to the left of a field stored to be signed. Also, before any monitoring commences, the initial length, sign, and contents of the accumulator and each auxiliary storage unit will be given.

Experience with the routine may show that some of the output is not useful enough to warrant the memory space and additional execution time required, and a subsequent version of the monitor may be written which does not include such features. Meanwhile, starting with the assumption that it will be easier to trim than to add, an effort was made to make the monitoring routine output a really complete record of the computer activity.

Paradoxically, a prime requirement of a monitoring routine is that it have the ability to execute sections of a code without monitoring and at full speed. There was no question that this feature would be included in the *705* monitoring routine, the problem was how to specify those sections. The method chosen is simply this: The coder prepares sequence cards, each specifying the location of the first

H. V. Meek is with the Hughes Aircraft Company, Culver City, Calif.

instruction of a sequence to be monitored and the location of the last instruction of the sequence; and the number of those times that the sequence is encountered it is to be monitored. As many as five different sequences may be specified for a single monitoring run.

Before monitoring begins, a table of the information from the sequence cards is automatically compiled in the computer, and the first instruction of each sequence is saved and replaced by a "transfer" instruction which will transfer control to an appropriate monitor entry point. When the table is completed, control is transferred to the first instruction, the location of which is given, along with the address of the unit selected for monitor output, on another card, of the code being tested. Thus, full-speed execution of the code takes place until a sequence to be monitored is encountered; then the contents and length of the accumulator and each auxiliary storage unit is given, and monitoring commences, continuing until the last instruction of the sequence has been executed. At that time, the accumulator is restored to its current condition with respect to the code being tested, the "transfer any" indicator and the check indicators, except for "sign" and "overflow," are off, control is returned to the code being tested, and full-speed unmonitored operation again takes place until another sequence, or the same sequence, if it is to be monitored more than once, is encountered.

Another very important question is: Should monitoring time, or monitoring routine memory requirements be minimized? The nature of the *705* (with its variable word length feature and somewhat complex instruction composition) requires that a great many operations be performed on a code to produce the desired information. Memory space is at a premium for, after all, the routine could give no service if no other code would fit into the memory to be tested, and its use must be minimized, even at the expense of slower monitoring.

Finally, what restrictions are to be placed on the coder who wishes to use the monitoring routine, and what rules must he follow? Naturally, the fewer the restrictions, the better. The less complex the rules, the better. Alteration switches and check indicator switches should be set as if for a normal, unmonitored run. A sequence of monitoring must neither begin nor end between a "compare" instruction and its associated "transfer on high" and "transfer on equal" instructions, nor between a "select" instruction and its associated input, output and "transfer on signal" instructions, nor begin between a "receive" instruction and its associated "transmit" instructions. The coder must be careful to specify for monitor output the use of a tape unit or printer which will not affect the operation of the code being tested. For example, if he uses a type-*760* control unit in his own code, he should not choose for monitor output a tape unit or printer which is controlled by that same *760*, since then the contents of the *760* storage would be changed after every instruction. However, monitor output to a tape unit controlled by a type-*777* tape record coordinator (TRC) is written so that the TRC storage is by-passed and its contents remain undisturbed. And, naturally, neither the code being tested nor the data it uses should overlap the monitor in the memory. The monitoring routine which has been machine tested has no provision for monitoring the instructions peculiar to *760* and *777* operations, but a subsequent version of the monitor will have that provision.

Because the user of the monitor has only a few restrictions, the writer had several. For one thing, there is no practical way to save the contents and sign of one or more of the auxiliary storage units, use the storages, and then restore them to their original conditions. Therefore, only the *00* accumulator is used to compile and edit the output information. Since it is possible that the code being tested has a "shorten" instruction designed to recover the remainder of a divi-

Table I

Operation Code	Mnemonic Symbol	Information Given in Monitor Output (See Legend)	Approximate Central Processing Unit Time in Milliseconds
A	NOP	(0)	25
B	SET	(0), (6), (7), (8)	30
C	SHR	(0), (9), (10), (11), (12)	33
D	LNG	(0), (9), (10), (11), (12)	33
E	RND	(0), (9), (10), (11), (12)	34
F	ST	(0), (4), (5), (13)	39
G	ADD	(0), (1) (2), (3), (6), (7), (8)	52
H	RAD	(0), (6), (7), (8)	32
I	TRA	(0)	26
J	HLT	(0)	Manual restart time
K	TRH	(0)	23
L	TRE	(0)	22
M	TRP	(0), (16)	28
N	TRZ	(0), (16)	28
O	TRS	(0)	28
P	SUB	(0), (1), (2), (3), (6), (7), (8)	52
Q	RSU	(0), (6), (7), (8)	32
R	WR	(0), (15)	20+execution time
S	RWW	(0)	28
T	SGN	(0), (1), (3), (4), (6), (7), (8)	60
U	RCV	(0)	19
V	MPY	(0), (1), (2), (3), (9), (10), (11), (12)	53
W	DIV	(0), (1), (2), (3), (9), (10), (11), (12)	53
X	NTR	(0), (6), (7), (8)	43
Y	RD	(0), (15)	20+execution time
Z	WRE	(0), (15)	20+execution time
1	TR	(0)	21
2	SEL	(0)	26
3	Control Instruction[1]	(0)	20+execution time
4	CMP	(0), (1), (3), (6), (8)	58
5	SPR	(0)	30
6	ADM	(0), (1), (2), (3), (4), (5), (14)	52
7	UNL	(0), (6), (7), (8)	31
8	LOD	(0), (6), (8)	31
9	TMT	(0)	19

Legend:

(0) location of instruction (5 decimal digits), mnemonic symbol, address part (5 decimal digits) accumulator or auxiliary storage unit (ASU) reference, and the uninterpreted instruction
(1) operand
(2) minus sign if operand is negative
(3) length of operand
(4) result in memory
(5) length of result in memory
(6) result in designated accumulator or ASU
(7) minus sign if result in designated accumulator or ASU is negative
(8) length of result in designated accumulator or ASU
(9) result in accumulator
(10) minus sign if result in accumulator is negative
(11) length of result in accumulator
(12) "*" if an ASU is designated
(13) "*" if the character to the left of the field "stored" was signed plus
(14) "S" or "U" according to whether the ADM is "signed" or "unsigned"
(15) "X" if reading is under control of Memory Address Counter II and TMT has been issued since the last RWW
(16) address of the last changed ASU if an ASU is designated

[1] The appropriate mnemonic symbol, determined by the address part of the control instruction, will be given.

sion, it is necessary that the starting point counter of the *00* accumulator remain unchanged by the monitor. This requires that the monitor have no "shorten," "lengthen," "round," "multiply," or "divide" instructions. One further restriction on the writer: The "sign" and "overflow" check indicators may never be turned on by the monitor, for that would interfere with the use of those indicators by the code being tested.

The instructions needed to produce the desired output and still satisfy the foregoing rules are, of course, more numerous than those needed if complete freedom in the use of the *705* had been possible. Between 4,000 and 5,000 memory locations are used by the monitor to store its 675 instructions and 400 characters of constants, and to provide 650 positions of temporary storage. The constants include a table of 192 characters which relates the operation codes to their corresponding mnemonic symbols. The large amount of temporary storage is necessary for concurrently saving the contents of the accumulator, determining the length of an operand or result, and compiling the output record. The monitor code is a relative code and may be assembled to operate in any part of the memory, provided the 192 table character location addresses all have the same thousands digit.

Table I lists each operation, the type of information that will appear in the history, and the approximate central processing unit time required to collect that information and write a record.

The Logical Design of a Digital Computer for a Large-Scale Real-Time Application

M. M. ASTRAHAN B. HOUSMAN J. F. JACOBS
R. P. MAYER W. H. THOMAS

THE Lincoln Laboratory and International Business Machines Corporation (IBM) have, over the past 3 years, worked out the design for a new digital computer which is the central component of a large-scale real-time system. In this system, data from a large number of sources are fed automatically into the computer where they are processed under programmed control. A complete compilation of the real-time situation is compiled by the computer and presented to operators by means of a special display system. The computer automatically generates control commands for the external environment in response to corrections and command information fed into it by the operators.

The purpose of this paper is to describe the performance criteria of the computer in general terms, and to present some of the outstanding features of the design. These features are necessary in the computer because of its particular real-time application which requires a greater emphasis to be placed on reliability, speed, capacity, and flexibility than is usual in scientific or commercial applications.

M. M. ASTRAHAN, B. HOUSMAN, and W. H. THOMAS are with the International Business Machines Corporation, Poughkeepsie, N. Y.; J. F. JACOBS and R. P MAYER are with the Lincoln Laboratory of the Massachusetts Institute of Technology, Lexington, Mass.

Disregarding its special features for the moment, the computer is a large-scale general-purpose single-address parallel digital computer with a 32-bit word length. A high-speed magnetic core memory is provided which contains 270,336 bits of storage arranged in two banks of 33 planes, each plane consisting of a 64 by 64 core matrix. The 33d plane is used for parity check bits. In addition to its buffer drums for communicating with the external environment, an auxiliary drum storage system is provided. This system contains 3,244,032 bits of storage divided among eight cylinders each consisting of six fields of 33-bit words. Each field has 2,048 words distributed around the circumference of a cylinder. Five magnetic tape units of the IBM *728* design and a large cathode-ray tube display system are also provided. In order to obtain a reliability capable of providing continuous 24-hour operation, the whole machine, with the exception of some of the input-output equipment, is duplicated. This guards against over-all system catastrophies caused by sudden machine failures and allows planned partial shutdowns for maintenance purposes.

One major consideration in the design was the availablity of components, particularly in the case of the high-speed memory. The original thinking on this computer resulted in a need for a memory faster and larger than in any existing computers. This pointed toward the magnetic core memory development which had been under way for some time at the Massachusetts Institute of Technology (MIT). The logical design of the computer was centered around the engineering judgment that the minimum memory cycle time would turn out to be in the order of 6 microseconds and that the largest matrix of cores should be 64 by 64.

The availability of components also affected the choice of the high-capacity storage medium for which magnetic drums were chosen. The drum which seemed the most attractive and the one which was chosen was that used on the IBM *650*. The vacuum tubes and circuitry in the computer were developed from designs previously made by the Digital Computer Laboratory at MIT and the Electronic Data Processing Machine (EDPM) development laboratories at IBM. The redesign of these basic circuits was dictated by the increased reliability requirements. In many cases this necessitated the use of more vacuum tubes and other components than are required in scientific or commercial applications.

The features of the computer which are the primary subject of this paper are in the logical design areas. The majority of these features were dictated by the need to obtain the most efficient use of the drum capacity and core-memory speed with the minimum number of circuits. As a consequence, special attention was given to overlapping operations within the machine and to the balance between the available components and the logical arrangements which would maximize their usefulness.

The first of these features which will be discussed is the arithmetic element. Many of the data which are processed by the machine are in Cartesian co-ordinates. A large number of computations which are involved in the application perform the same operation on x as on y. A saving in time is effected by the use of a dual arithmetic element which treats these quantities separately and simultaneously. The second feature is a high-speed multiplication technique which dictated the design of the adder in the arithmetic element. It was found that all of the proposed arithmetic operations, with the exception of multiplication and division, would be performed during the cycle time of high-speed memory. A special adder was developed so that the multiplication process could be made more compatible with the memory cycle.

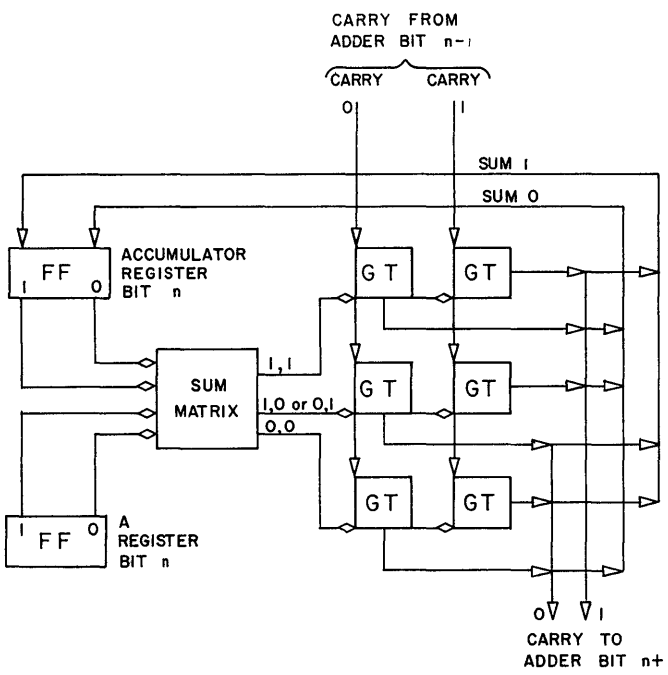
The third feature is an indexing system which automatically takes care of the large class of bookkeeping operations (i.e., address modification) without taking extra operation time. The fourth of these features is an input-output control system which was designed to allow the computer to utilize the time which might otherwise be wasted during periods when data are being transferred between it and the input-output system. This is done by allowing the computer to continue doing work while waiting for the input-output equipment to present, or receive, its information. The fifth and final feature is a buffer drum system which matches the computer speed to the incoming data rate.

Arithmetic Element

In the design of a digital computer, much consideration must be given to the characteristic known as "word length." For many real-time applications, much of the input data are generated by analogue devices such as thermometers, and much of the output data are sent to analogue devices such as valves or indicators. A precision of 10^3 is often sufficient for such input and output data, with a precision of about 10^5 being used to reduce the effects of roundoff and truncation errors during intermediate calculations. Thus a word length of from 10 to 16 bits is implied.

In a stored-program computer, consideration must also be given to the number of bits required to specify an instruction. A basic operation code of 64 instructions requires six bits. For memory sizes ranging from 2,048 to 16,384 registers, the address part of an instruction requires from 11 to 14 bits. Special control bits, such as those used for automatic indexing, add four or five more bits to the instruction word. Thus, the word length for a full instruction (including the basic operation code, control bits, and addresses) might be between 21 and 25 bits.

Fig. 1. Adder circuit



The conflict between the 16-bit word length for data and the 25-bit word length for instructions was resolved by the choice of a 32-bit word length. This allows two 16-bit numbers to be placed in any register and allows additional control bits to be included in instruction words. These additional control bits are used for increasing the potential speed of the machine. Since the memory must be capable of delivering the full register simultaneously for instructions, it can also deliver both numbers simultaneously. A separate arithmetic unit was provided for each of these numbers so that both could be manipulated simultaneously, thus doubling the speed of the computation. This arithmetic unit was termed the "dual arithmetic" unit.

Special operations and control bits have been added to the basic operation code to increase the utility of the dual arithmetic unit. A conditional transfer of control, or branch, can be made to take place if a specified arithmetic unit is negative, or only if both are negative. It is possible to store the full 32-bit word, or either half independently. (The address part of an instruction is placed in the right-half word; hence a "right store" instruction can also be used for modifying addresses.) Shift and cycle instructions are provided for shifting either arithmetic unit itself, or both together, or for cycling data between the right and left arithmetic units. A "twin" feature, available on some instructions, allows the left-half word from memory to be sent to both arithmetic units simultaneously. Thus, a vector in the dual arithmetic unit may be multiplied by a scalar from a left-half word of memory by a single "twin and multiply" instruction.

Each instruction which can cause an overflow contains a pair of control bits which allow suppression of an overflow alarm from either or both arithmetic units. Of course it is not always possible to make efficient use of the dual arithmetic element. Single 16-bit words can be handled just as easily by using the afore-mentioned controls, with no loss in speed except for an occasional cycle to exchange left for right.

Regardless of the word length used, there is usually a sizable body of data consisting of items only a few bits long. Storage space can be saved by packing a number of such items into a register of storage. Packing and unpacking such items can be very time consuming unless special instructions are provided. In this computer, an "extract" instruction allows a given item to be obtained from a memory register without obtaining other items from the same register. A "deposit" instruction allows the item to be replaced in storage without disturbing other items in the same register.

High-Speed Multiply

The memory unit of a general-purpose computer must be used one or more times for every instruction. In a real-time computer the memory should be made as fast as possible and should never have to wait

for other processes. In this computer, with the design goal of 6 microseconds per memory cycle, it would have been desirable to design a 6-microsecond multiplier because of the frequent use of the "multiply" instruction. On the other hand, a basic flip-flop speed of 0.5 microsecond and a word length of 16 bits allowed a reasonably simple 8-microsecond multiplier, using 0.5 microsecond for each addition and shift.

Each step of a multiplication should allow for the possibility of a carry propagated a full register length, or 16 bits. Although such a carry condition cannot occur mathematically, it can be used to simplify the design procedure because it is a reasonable approximation to mathematically possible carries. If the basic gate-tube propagation time is 0.04 microsecond, then a 16-bit carry will require 0.64 microsecond. To reach the goal of 0.5 microsecond per multiplication step, the carry is accomplished concurrently with the 0.5-microsecond flip-flop resolution time. This is done by making sure that all pulses which arrive at a given flip-flop are spaced 0.5 microsecond apart even though they are carry pulses which have arrived as much as 0.64 microsecond after the initiation of the carry.

As shown in Fig. 1, each bit position makes use of a diode matrix to form the sum of two bits: one from the augend in the accumulator, and one from the addend in the *A* register. The matrix controls two sets of gate tubes: one set to be sensed by an incoming "0 carry" pulse, and the other by a "1 carry" pulse. When a carry pulse arrives, only one gate tube passes it and indicates the sum of the two bits plus the carry. This pulse is then sent out as either a "0 carry" or a "1 carry" pulse to the next highest bit, and as either a "zero" or a "one" sum pulse to the accumulator flip-flop, depending on the value of the sum. Thus, the carry takes only the gate-tube propagation time to pass each bit. Carry pulses supplied to the right (the least significant) end at 2 megacycles will cause every bit to receive carry pulses (of either "0" or "1" value) at a 2-megacycle rate, although the pulses at each bit will be displaced in time, depending on the bit's distance from the right end of the register. This principle forms the basis of the so-called "asynchronous adder."

For multiplication, each "add" step must be accompanied by a shift-right of one place, and some steps (for a zero multiplier bit) will require a shift-right without any addition. The shift on "add" is easily accomplished by sending the "sum" pulse from the adder to the accumulator flip-flop to the right instead of to the flip-flop which generated the sum. The shift-right without addition must propagate down the accumulator at the same rate as the carry in order to avoid conflict with a previously initiated carry. This so-called "ripple shift" is accomplished simply by letting the shift pulse from one bit initiate the shift command for the next bit.

Special controls could have been included to allow the sum to be displaced to the left instead of to the right during a divide process, and to be undisplaced during a plain addition. However, the expense of such controls was not justified. A corrective shift-left for "add" instructions (including a second corrective shift-left after an end-around carry) can be accomplished without delaying the memory cycle. Unlike "multiply," which is used extensively in most programs, the extra time required for "divide" does not appreciably increase the average instruction execution time because of the infrequent use of the "divide" instruction in most programs.

Indexing System

Most digital computer programs involve sequences of instructions which are repeated many times on different data. It is very wasteful of memory space to include separately stored instructions to process each piece of data since the instructions will vary only in the address to which they refer. On the other hand, time is required to modify instruction addresses for each execution. An indexing feature has been included in this machine to minimize this time.

The indexing feature consists of a set of four index registers, an index adder, and associated control circuitry. Each instruction which refers to a memory address may have control bits inserted into it to specify an index register. The contents of the specified index register are added to the address part of the instruction prior to its execution; however, the instruction as stored in memory remains unchanged. This indexing addition does not require any extra time since it is executed while waiting for the memory cycle to be completed. Thus, if an "add" instruction with an address part containing 1,000 is executed, and if its control bits specify an index register which contains 24, the instruction will be executed as if it had an address part of 1,024 but will remain stored in memory as "add 1,000."

In order to utilize an index register in a cyclic program consisting of a "loop" of instructions, it must be possible to set up the index register to an initial value. Two instructions, "reset index register" and "reset index register from right accumulator," have been provided in this computer. They load the specified index register with the address part of the "reset index register" instruction or with the contents of the right accumulator.

There are three other functions that the indexing system must provide: (1) the modification of the index register each time the loop is executed, (2) the testing to determine if the loop has been executed the desired number of times, and (3) the branching of control back to the beginning of the loop unless it has been executed the desired number of times. All of these functions are handled by one powerful instruction, "branch and index." Control bits included in the "branch and index" instruction specify an index register and a decrement. Each time it is executed, the specified index register is first inspected to determine if it contains a negative number. If the number is positive, the contents of the register are reduced by the specified decrement and by the instruction branches to the address specified in its address part, usually to the beginning of the loop. Thus, execution of the loop will continue until the index register contents have been reduced to a negative number, at which time the branch is not executed and the program continues in sequence.

In addition to the four index registers, the right accumulator may also be used as an index register. This feature was included to facilitate table look-up programs. Assume that a program has been executed so that the table argument has been computed in the right accumulator. If the next instruction executed is a "clear and add" instruction which specifies the right accumulator as an index register and has the first address of the table as its address, the first address and the argument will be added before the instruction is executed. Thus, with only one instruction, the desired table contents are obtained.

When an indexed loop is used in searching for a desired value in a table, it is often necessary to determine the contents of the index register when the desired value is found. An instruction called "add index" has been provided to permit this important operation.

Input-Output Control

The system application requires that large quantites of data be entered into the core memory, processed, and delivered

out again. The data sources and destinations have varying and unpredictable word rates and access times. Since computing time is at a premium, the input-output control design goal was that an arbitrarily sized block of words be transferred with a minimum of time devoted to the transfer. This minimum consists of the time needed to execute the program steps which set up the transfer, plus the one memory cycle required to transfer each word into or out of core memory.

Reading into core memory will be assumed in the following: Writing out of memory is analogous. The design goal required the use of an independently operating input-output control which could remember the input-output unit selected, the number of words to be transferred, and the location in core memory for the block of words. A "break" system was needed which could interrupt the program operation for one memory cycle whenever the input-output unit had provided a word. The instructions in the computer are designed in such a way that a "break" memory cycle can be initiated at the end of any memory cycle with no effect on the instructions other than to delay their execution.

The system chosen operates as follows: Special instructions in the computer connect the proper input-output unit to the information transfer paths, load counters which keep a record of the location in memory in which the data are to be stored and keep a count of the words transferred, and start the flow of data. The program operation then continues normally, except for interruptions of one memory cycle per word caused by the break system. Each time a word is transferred, the counters are stepped accordingly in preparation for the next word.

The operation is terminated when the counters signal that the requested number of words has been received. It can also be terminated earlier by a disconnect signal from the input-output unit. This occurs when the program has requested more words than the input-output unit has to send and when the input-output unit has run out of words. Facilities have been included to allow the computer to examine the counters in order to determine how many of the words requested were transferred.

Interlocks hold up the program if an input-output operation is called for before the preceding one is finished. Therefore, to use the system efficiently, the programmer must provide enough work for the computer in order to consume all the time taken by the input-output operation before another input-output operation is started. To help accomplish this function, a "conditional branch" instruction is provided which can detect whether an input-output operation is still in process. This instruction also provides the programmer with a means of determining whether an input transfer has been completed before attempting to use the data.

Drum Buffer System

The real-time application for which this computer was designed required that the computer receive its input data from many independent, asynchronous sources. The exact quantity of data to be received from any one source could not be determined; however, it was possible to estimate the average and maximum amounts with a reasonable degree of accuracy. The application is such that the total amount of input data received from all the sources combined is less than the sum of the individual maximum amounts.

Another characteristic of the data sources is that they operate at a much lower speed than that of the computer. It is not operationally feasible to interrupt the computer operations to accept each piece of data as it arrives. A buffering mechanism is necessary to gather the data at the slow incoming rate and then pass on large blocks of the data to the computer at the computer's speed. Magnetic drums were chosen for this buffer.

The desired characteristics of the drum buffer system follow: The input data should be written on the drum as soon as possible after they are received and definitely before another piece of data is received from the same source. The writing of input data on the drum should not interfere with the reading of data by the computer. The computer should be able to read selectively from the drum; that is, to read data from only one source at a time. In order to minimize the number of drum storage registers required, the buffering system must be able to combine the data received from many sources.

To provide the "no inference" characteristic, a dual-access drum is used. This drum has two sets of drum heads. One set, called the outside of drum system (OD), is used to communicate only with the outside world. The other set, called the computer-side of drum system (CD), communicates only with the computer.

To provide the "minimum storage" and "write as soon as possible" characteristics, the "random storage" drum was developed. Each drum register has a status bit associated with it. This bit is used to indicate the "full" or "empty" status of the register. As a register passes under the OD drum heads, the status of the register is sensed. If the register is empty, a "drum demand" pulse is generated and sent to interrogate the data input equipment. If there are data from one of the sources, a "data available" pulse is sent back to the drum status circuits and the data are sent to the drum write register. As the information is written on the drum, the status marker is changed to indicate that the register is now full.

A single drum channel is not used for the status indication. There is not enough time to read the status bit, decide whether it indicates "full" or "empty," generate the "drum demand" pulse, interrogate the data sources, and write the full indication with the same drum head. If a second head were used on the same drum channel, for physical reasons, it would have to be located many registers away, thus necessitating the use of a shift register or delay-line device between the two heads. Therefore, two drum channels are used for the status indication, and the indicator bit is shifted back and forth between them. The two channels are called the OD and CD status channel since they are read by the OD and CD side of the drum, respectively.

Consider first the OD side of the drum system as shown in Fig. 2. When a piece of data is received by the input equipment associated with a data source, a "data received" pulse is generated and is used to set the corresponding flip-flop in the drum demand chain to the ONE condition.

As each drum register approaches the drum write heads, the status bit on the OD status channel associated with the register is read. The status read head is located slightly before the write heads. If the status bit is a ONE, indicating that the register is full, the only action that takes place is the writing of a ONE in the CD status channel by the status write head. If the status bit is a ZERO, indicating an empty register, a "drum demand" pulse is generated and sent to the drum demand chain. This pulse interrogates each of the flip-flops in order until it finds one which has been set to the ONE state or until it reaches the end of the chain. If the pulse encounters a flip-flop in the ONE state, it becomes a "data available" pulse and (1) resets the flip-flop, (2) causes the data from the source associated with the flip-flop to be transferred to the drum writing circuits, and (3) causes the status circuit to write a ONE in the CD status channel. If there are no data available, the ZERO indication is passed on to the CD status channel.

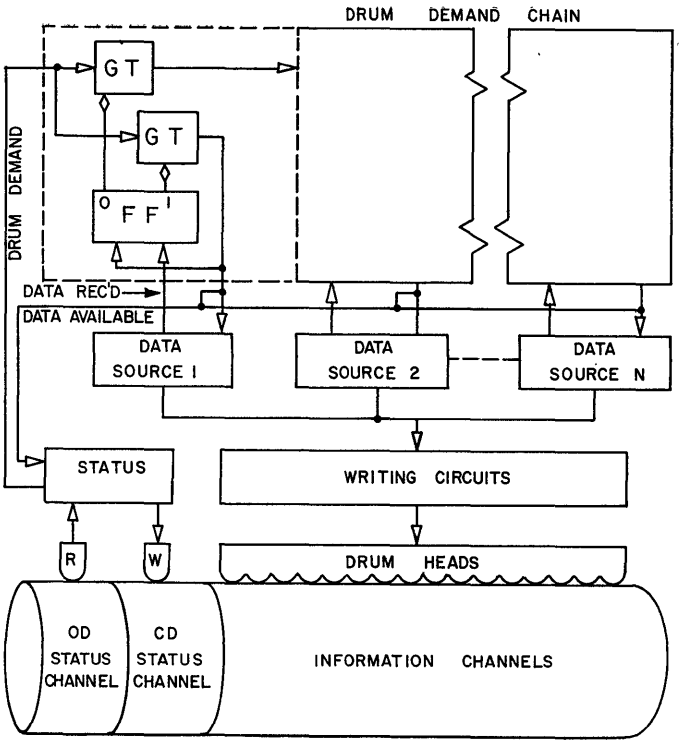


Fig. 2. OD side, drum system

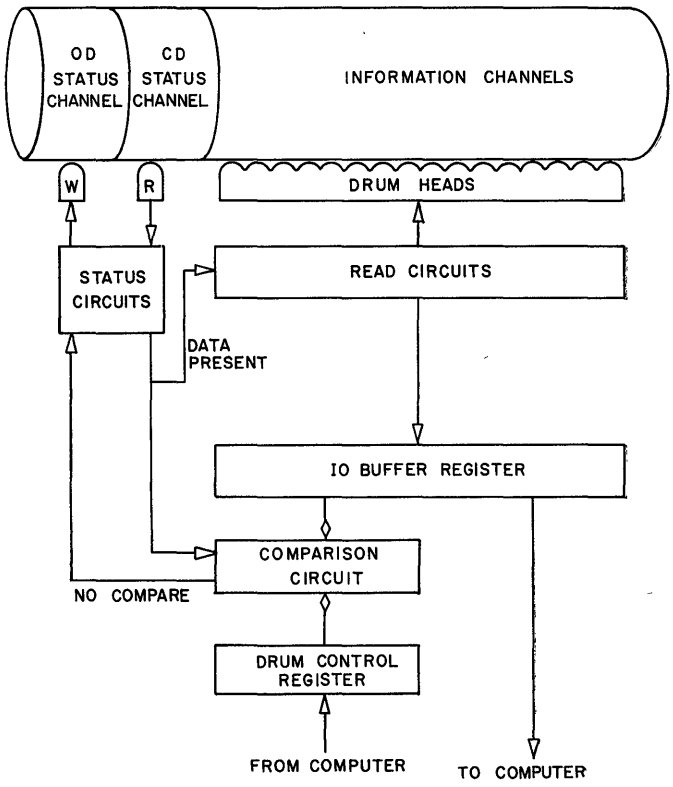


Fig. 3. CD side, drum system

Each piece of data written on the drum has some identity bits associated with it to identify the source of data.

There is a possibility that as the drum fills up, some of the data received will not be stored on the drum before the next piece of data is received from the same source. This is particularly true of the sources at the tail end of the demand chain. The probability of storage on the drum depends on the rate of input data and on the number of empty registers on the drum. Most of the data rates are such that if the drum is kept at least 50 per cent empty by the computer, the probability of storage is better than 0.99. The data received from this type of source are such that the occasional loss of a piece of data does not seriously affect the system. Fortunately, the rate from the critical sources is slow enough to allow a complete search of the drum for an empty register before a second message can be received from the same source.

Now consider the CD side of the drum as shown in Fig. 3. The status heads and circuits on the CD side are almost identical to those on the OD side. During periods when the computer is not reading the drum, the status information is automatically and continuously transferred from the CD status channel to the OD status channel.

The computer may read the drum in either of two modes, status or identity. When reading in the status mode, the contents of every full register passing under the heads is transferred to the core memory through the input-output buffer register and a ZERO is written in its associated status bit in the OD status channel, indicating that the register is now empty. When reading in the identity mode, the computer first places the identity code desired into the drum control register. Only those words with matching identity are transferred to the core-memory.

The transfer of words continues until a disconnect pulse is received from the computer, indicating that it has received the number of words it has asked for, or until the drum itself generates a disconnect pulse. With an unknown quantity of data received from each source, the computer may ask for more data than has been received. In fact, in order to insure

Fig. 4. Portions of computer in test cell

that the computer receives all the data that have collected on a drum field between readings, the normal procedure is to ask for more than expected. A disconnect pulse is automatically generated at the end of a complete drum revolution, and the number of words read is determined from the word counter contents.

Some of the data sources have messages which contain more information than will fit into one drum register. To handle such information, another feature has been added to the drums associated with these sources: the drum has been divided into multiple word slots of adjacent registers. For this application the only meaningful status bit is the one associated with the first register of a slot, and the source identity is contained in the first register of a slot. The operations associated with those drums are almost identical to those of the single register drums.

In addition to the input buffer drums just described, there are output buffer drums which handle outgoing data. They operate in a similar, though inverse, manner to the input drums.

Conclusion

Fig. 4 illustrates a portion of the computer in the test cell. Because of its size and layout, it was not possible to obtain a picture of the whole computer. The unit in the foreground is the operator's maintenance console. It contains switches for manual data or instruction entry and manual control, neon indicator lights for the major flip-flops and registers in the computer, visual and audible indicators for computer generated alarms, marginal checking controls and indicators, and power system and air-conditioning indicators. The cutout contains a view of one of the memory units.

The computer, including the directly connected input-output equipment, contains approximately 12,500 tubes. It has an execution time of 12 microseconds for arithmetic instructions, excluding "multiply" and "divide," which require 15.5 and 53 microseconds, respectively. The prototype model has been in satisfactory operation for more than one year.

Computer Design to Facilitate Linear Programming

R. C. GUNDERSON

AT the risk of being redundant, this paper will begin by stressing the growing importance of linear programming in business, industry, and government, as it is this importance which is its motivation.

Primarily, it is the application of linear programming with which the paper will deal. Optimal planning of procedures has become a necessity, rather than a luxury, to present-day management. For example, one organization is presently saving an estimated $20,000 a day by optimal planning through linear programming procedures. Without too great a stretch of the imagination, the fascinating possibilities of linear programming linked with automation might be pictured. This could yield factories staffed with skilled technicians to feed data from changing markets into computers, which would then choose the optimal combinations of specifications and direct the machinery to produce under these new specifications. There are countless other such possibilities which could make effective use of this powerful tool.

R C. GUNDERSON is with Remington Rand Univac, St. Paul, Minn.

It is not intimated here that the use of linear approximations is a new addition to mathematics or economics. Rather, it is the wider acceptance of their usefulness which is new. This, coupled with the fact that much work has been done in the past decade to develop a general formulation of the computational procedures involved in linear programming, presents an exciting facet of computer application to users and manufacturers.

It is evident, then, that some consideration should be given to the requirements of this problem in the building of our future computers. Essentially, the actual needs are for the most part familiar to the computer industry. Moreover, the logical properties of computers which this problem requires are extremely compatible with those desired by logicians.

Let some of the qualities of this problem which make it especially well adapted to high-speed digital computation be examined for a moment. First, since input and output time still lags behind computation time on all present-day large-scale computers, the relatively small amount of input, simple but voluminous computations and logical operations, and the small amount of output required, lend themselves well to computers. Second, the iterative nature of the matrix manipulations is ideally suited to stored program computation. Finally, the ability to generalize the procedure, enabling the solution of a number of maximization or minimization problems containing dissimilar data, reduces the programming involved to mere data preparation.

In the following, some of the essential physical properties of a computer which make handling linear programming problems more efficient will be discussed. Operating on matrices by rows or columns necessitates much greater rapid access storage than an element by element operation would require. Present-day problems, which undoubtedly will soon be dwarfed, require a minimum of 4,000 words, and would run much more efficiently with 8,000 to 12,000 words of rapid access storage. The so-called "housekeeping" operations required by the limitations of present storage systems increase running time by approximately one fifth.

The Simplex method of solution, probably the most efficient and most used linear programming procedure, requires access to the stored matrix of coefficients at random, as dictated by the computation. Moreover, the generation of an additional vector during each major iteration necessitates a large-capacity secondary storage in the more sophisticated problems. There is, then the additional requirement of a large, random access, secondary storage media, probably 15,000 to 30,000 words in size,

backed up by several hundred thousand words of magnetic tape storage.

The scaling and storing of the matrix elements, the packed floating vector representation of numbers, and the necessity of shifting for multiplication and division of scalars suggests the importance of addressable shifting registers with the possibility of both left and right shifts.

Since the solution process is iterative in nature, and since it is impossible to predict at any point the number of iterations necessary to reach a solution, it is essential that there be some facility for repeating general sequences. This may be accomplished by either or both of two methods, "b-boxing" or repetitive commands. Although b-boxes have some advantages, these seem outweighed by the versatility and flexibility of a command structure which accomplishes the same and possibly more. For example, the entire linear programming procedure is cyclic in nature, and within this major cycle are contained several minor cycles, each of which has subcycles, and in some instances sub-subcycles. So the possibility of an array of b-boxes approaching the number of rapid access storage cells required by the problem is presented. Furthermore, some thought should be given to the considerable cost of additional hardware required by b-boxing.

In addition to the afore-mentioned physical properties, some thought should be given to the command features most desirable for this problem. Let the implications of address structure be considered first. It is quite evident that 2-address logic has distinct advantages over single-address logic in so far as the mathematical operations are concerned. A significant reduction in the number of commands required to perform the arithmetic can be realized by combining several steps in one command. For example, in forming the sum of two vectors, it is necessary to set a component of one of the vectors in the sum register, add the corresponding component of the second vector to it, and store the result. With single-address logic, this would require three operations, but with 2-address logic the procedure may be accomplished by a single command. Similarly, in forming the scalar product of two vectors, it is possible with a 2-address structure to form the product of two corresponding vector components and add the result to the product of the preceding components in a single operation. In fact, by proper modification, the entire scalar product may be performed by a single command. The arithmetic instructions present one area of command structure where 3-address logic might be contemplated.

Evidence indicates that 2-address logic is also desirable for the logical operations required by the procedure. Although a certain amount of the "housekeeping" operations could be done rather efficiently by a single-address scheme, the extreme versatility of the 2-way conditional jumps allows a much more general approach to the problem with less housekeeping. Furthermore, the logical sums and products which prove very useful in the matrix and vector manipulations require two addresses to make their operation feasible. In no instance does it appear that 3-address logic could increase efficiency to any great extent.

In keeping with the logical structure of commands, the actual properties of the command for which linear programming, or any type of problem involving matrix or vector arithmetic, has a particular need will also be considered. As indicated previously, a command which may be easily modified is necessary to perform the sum of two vectors most efficiently. This command should be such that the sum of two corresponding components may be formed and stored in one operation. Similarly the scalar product of two vectors should be performed in such a manner that the procedure may be generalized without extensive housekeeping. It is quite evident that the more instructions that are required to perform these general and often used operations, the more presetting or resetting that becomes necessary. As a result of these first two arithmetic operations, a third requirement is evidenced, namely a method to rescale the resultant vector and scalar to their proper positions; this requires a command which will explore the resultant for the first significant digit of each component, and indicate in which position this digit appears.

Before investigating the properties of the logical commands required for an efficient linear programming procedure, it is necessary to have some understanding of the nature of the linear systems involved, and the methods of representing the numbers occurring in these systems. The matrix of coefficients of the original systems are generally rectangular, with a high incidence of zero coefficients. For greatest efficiency in storage and speed of arithmetic operations, some method of suppressing these zero elements seems to be indicated. For, as the user of any linear program gains experience and confidence, the dimensions of the systems increase to the point where time and storage space are of very real importance. Therefore, the problem of developing some form of number representation which will suppress zero coefficients and still preserve the significance of the nonzero elements of the matrix is presented.

Both of these may be accomplished by a type of floating point representation known as floating vector. This system allows 32 bits of significance plus a sign bit, and a characteristic or scale factor of 15 bits for each column of the matrix. Furthermore, there is no need for additional floating point arithmetic hardware, since each vector is operated on as a unit using the normal arithmetic operations, which are generally faster and more efficient. The use of such a representation is contingent of course upon the existence of a double length accumulator or sum register to prevent overflow in the scalar product of two vectors.

By giving each vector one, or several, keywords, which have digits indicating the position of nonzero components and zeros indicating the position of zero components of the vector, the significant elements only may be stored and all zero elements of the matrix ignored. For example, the vector

$$\mathbf{X} = (3, 0, 1, 0, 0, 5) \tag{1}$$

would have the binary keyword

$$101001 \tag{2}$$

and the vector would be stored thus:

Location	Binary Representation
A	101001
A + 1	000011
A + 2	000001
A + 3	000101

(3)

By a representation of this sort, both increased speed and reduced storage may be accomplished without losing the significance of the numbers involved.

The question now arises as to how this floating vector notation may most efficiently be used. First, the numbers must be packed and the vector keyword formed. To do this, each component of the vector must be tested for significance, a digit inserted in the keyword in the proper position, and the significant components stored. Second, there must be some method of interpreting the keyword in order to unpack the vector for some part of the procedure such as the vector sum. Finally, to reduce the amount of time consumed in forming the countless numbers of scalar products, the corresponding significant components of two vectors must be predetermined to preclude the possibility of multiplication by zero.

Again, caution should be exercised in determining the numbers of commands necessary to perform these operations so that a generalization of the procedure will not entail too extensive housekeeping.

The first of these logical problems may be solved by a number of different approaches. However, the second problem, the interpretation of the keyword for unpacking, requires an instruction which examines the word digit by digit and controls the storing of the significant components in their proper position in the vector, suggesting a type of conditional jump command. The zero suppression multiply may be accomplished by forming the logical product of the keywords of two vectors, forming a new keyword which would control the multiply sequence exactly as the unpacking procedure. To illustrate the advantage of this operation, let it be supposed there is the vector,

$$\mathbf{y} = (2, 1, 0, 4, 0, 0) \tag{4}$$

with the binary keyword,

$$110100 \tag{5}$$

and suppose forming the scalar product of $\mathbf{y}$ and the vector $\mathbf{x}$ given by equation (1) is desired. This would be accomplished by forming the logical product of the keywords (2) and (5)

$$\begin{array}{ll} (2) & 101001 \\ (5) & 110100 \\ \hline & 100000 \end{array} \tag{6}$$

indicating that the only corresponding significant components of the two vectors are the first, and the scalar product would then involve only a single multiplication rather than six multiplications and five additions.

It is estimated that a computer designed with all of the afore-mentioned qualities would reduce the running time of present procedures by as much as a factor of ten. Since most of the requirements of this problem are compatible with many other types of problems, and furthermore, since many of these features may be found on existing machines, such as the Univac Scientific, it is the contention in this paper that all of these requirements should be fully investigated so that they may be incorporated in the future machines of this industry.

Considerations in Making a Data-Gathering System Compatible

B. L. WADDELL

THE paper discusses the design problems facing the data systems engineers who are required to produce a data-collection system that will be able to enter a computer easily. Some empirical formulas are presented with a discussion of how to use these formulas. The many recording tools and their application to data systems which are being prepared for entry into computers are described with a discussion of the place of each recording device. The second section of the paper is a critical analysis of four data recording or gathering systems designed to go directly to a digital computer.

B. L. WADDELL is with G. M. Gianninni and Company, Inc., Pasadena, Calif.

Using a Variable-Word-Length Computer for Scientific Calculation

FRED GRUENBERGER E. H. COUGHRAN

IN discussing the use of a variable-word-length computer, this paper will be restricted entirely to past history; that is, ideas and practices that are actually in operation. This implies, of course, that only the *702* will be talked about and the implied comparison to fixed-word-length machines is to a machine like the *701*.

At Hanford, Wash., the *702* has been used for scientific computing, with what is regarded as considerable success, since its installation in June 1955. The percentage of available machine time devoted to numerical analysis has steadily increased, standing currently at about 20 per cent. This is not to say that mathematics is taking time away from commercial work, but rather reflects the increase in efficiency on commercial problems and a general increase in mathematical problems as time goes by. Scientific computing has covered a wide gamut of problems, including linear programming, numerical integration, differential equations, and many complicated formula evaluations, as well as work on reactor data and weather data which might be called scientific data processing.

This work demands one large machine to serve for both commercial and scientific problems. This is a common situation, and perhaps any machine would suffice; it has been pointed out that through an automatic programming system, any machine can be made to appear like any other. Indeed, most programming is done with reference to the automatic programming system rather than the particular machine. The main contention here is that a variable-word-length

FRED GRUENBERGER is with the General Electric Company, Richland, Wash., and E. H. COUGHRAN is with the International Business Machines Corporation, Richland, Wash.

machine makes the over-all job easier and, in particular, it is felt that the variable-word-length machine offers some distinct advantages for scientific computing.

First, since the word length is arbitrary, floating decimal or fixed decimal calculations can be carried to any degree of precision with equal efficiency and equal ease of programming. At present, the floating routines use a 10-digit mantissa with a 2-digit exponent, which is treated as a single 12-character word; the exponent has a range of 200. Should greater (or less) precision be needed, no great effort would have to be expended to alter the routines for any other word length; in fact, the word length can be made a parameter. The appearance of the machine to the programmer is the same in all cases, and programs written for one word length will function properly in all cases, both on paper and in machine code. Moreover, the variable-word-length machine makes it attractive to carry through many calculations completely in fixed point which would certainly call for floating point on a fixed-word-length machine. It is awkward to do multiple-word arithmetic on a fixed-word-length machine, but easy to do the equivalently precise work on a variable-word-length machine. A fixed-word-length machine working in its own word length is obviously at an advantage; it is when the work calls for an odd word length that the variable-word-length machine excels.

Generally speaking, for an installation doing only scientific work, it would be natural to think of a pure computer, such as the *701*, *650*, or Datatron. However, even in this situation, the variable-word-length machine offers an advantage, if the data to be handled are intrinsically of an odd length. In data reduction, for example, large volumes of 2- or 3-digit numbers tend to waste either memory space or machine time or both, in a fixed-word-length machine.

There is much to be gained from having two groups of people working around one machine within one organization. The diverse backgrounds and experience of commercial and scientific people blend into a powerful team when the common problem, outwitting a machine, is approached. Programming insights and triumphs of either group become helpful to the other. For example, the floating point arithmetic subroutines, developed for the numerical analysts, have been used on commercial problems, where a given variable, such as stock quantity, may have low precision but a wide range. Similarly, a routine of particular value to the commercial people for producing tabular reports has proved to be just the thing for processing meteorological data and preparing tabular reports. A generalized sorting routine has become widely used by both groups, since internal sorting is common to all sorts of problems. Characteristically, commercial people have a background which is strong on procedures and flow-charting; mathematicians have an ingrained love of precise definition and compact symbolism. The two groups thus tend to complement each other.

It seems to be intrinsic to a data-processing machine like the *702* that it become surrounded by a greater quantity of more flexible peripheral equipment than does a pure scientific machine. This is no drawback to doing scientific computing on such a machine. For example, there was an initial feeling at Hanford that problems from the numerical analysis unit involving only a small amount of the peripheral equipment could be sandwiched in between commercial problems which tend to use every gadget available. This has not come about. The mathematical programmers have used the devices available to the fullest, with a corresponding speed and efficiency increase.

It is reasonable to assume that, regardless of the nominal capacity of main memory for a given machine, one pays for memory in direct proportion to the number of characters one can store at one time. Here the variable-word-length machine pays off again; it is parsimonious of main memory and tape memory, and hence of tape-read time. Main memory is always packed solid, regardless of how various numbers vary in length. Extremely long (blocked) records can be recorded on tape so that tape is packed with information and very little tape movement is wasted. It should be recognized that the *702* is an alphabetic machine, so that for pure numeric work some of its memory capacity is perforce wasted. Oddly enough, the drum on the *702* is very nearly a fixed-word-length device, with the word length 200 characters; precisely for this reason, drug storage space is sometimes wasted.

It is felt that a variable-word-length machine helps to ease the burden of devising an automatic programming system. This is difficult to prove or disprove; it can only be pointed out that this one was written and "debugged" in a matter of a few months. What seems clearer is that changes in the system are relatively easy to make on such a machine, and are being made constantly on the basis of developments and new requirements.

Under some conditions matrix algebra is at an advantage on a variable-word-length machine. In a large linear programming problem, it was found that a scalar product using a modified 10-digit floating arithmetic took roughly the same amount of computation time as the same scalar product using a fixed 15-digit word arithmetic.

Due to the serial nature of the *702*, built-in automatic checks are cheap and plentiful; no machine time need be wasted in programming the checks. Incidentally, the record of the *702* for not passing undetected memory errors is partically perfect.

An interesting feature of the *702* is that the machine seems to get better, programming-wise, as time goes by. On other equipment, the weaknesses, and features which should have been built in but were not, rapidly become of great importance to the programmer. Naturally, machines of infinite speed and capacity, taking no space, and renting for $5 a month would be liked by all; short of that, the designers of the *702* deserve commendation for a splendid piece of equipment. If one were to talk to the designer of a computer, one would probably ask "Why didn't you?" and he would always have a ready answer. When, however, one asks "Why did you?" he might be quite embarrassed; the point is, few "Why did you?" type of questions have been found for the *702*.

One of the significant advantages of the *702* accrues from its very simple structure. This is evidenced in both scientific- and commercial-type problems in two ways: (1) The simple structure of the machine makes it possible to define an abstract or "automatic coding" system from without rather than from within. That is, the machine is sufficiently simple and yet sufficiently flexible so that any automatic coding technique or philosophy can be described from the point of view of the problems to be solved and the demands for particular manipulations inherent in those problems, rather than starting from a set of machine restrictions and doing the best one can or going to great length to overcome inherent difficulties. (2) The machine is extremely easy to code in its own language, if necessary (indeed, where the number of different jobs is small, this is the only method used at some installations). This feature gives one tremendous power in two operationally vital areas. First, in processing a job, when machine failure, operator error in judgement or accident, or erroneous or unanticipated data cause one to "fall flat on his face," it is not necessary to

"go back to the drawing board." It is possible, and, on the basis of this experience, eminently practical, to pick oneself up off the floor. Several times, in one or the other of these circumstances, a corrective set of instructions has been programmed, keypunched, and a set of correct answers run off within the hour. Second, at certain unpredictable and unannounced times, problems of moderate difficulty are presented to the computing section for immediate solution. These are problems of extremely high priority with a machine solution time of 1/2 to 3 hours and a total elapsed time from problem presentation to final solution of 4 to 24 hours. This time scale does not permit the normal cycle of algebraic manipulation, programming, debugging, test cases, and solution. The *702* is simple enough in its order structure to permit coding on an "on the spot" basis. Note particularly that corrections or other machine instructions can be entered directly via the card reader or console.

The last-mentioned advantage is part of the final point: The *702* is decimal and alphabetic, which makes for ease of communication between the machine and the programmers, and the machine and the console operators. Further, although numerical analysts work largely with a 10-character vocabulary, the fact that the *702* has 47 characters available is frequently useful, if only for dressing up printed reports.

Lest it be thought that the picture is too rosy, let some of the disadvantages to using a variable-word-length machine for scientific computing be stated. (1) The chief disadvantage probably lies in the economics of the operation: A pure numeric, fixed word-length, parallel machine with equivalent circuit speeds would perform each operation at greater speed, hence lower cost, particularly on long production runs where the word length happened to fit the problem's needs or could be made to fit. This statement would apply with even greater force to a machine with built-in floating point. (2) For some types of logical mathematical work, there is an advantage to using a binary machine, and all present variable-word-length machines are decimal. (3) The variable-word-length principle itself is responsible for programming slips that might not otherwise be made, in that word boundaries occasionally slip. For example, it is easy to store a 4-digit word at a place in memory where a 3-digit bucket has been set up; this defaces the word to the left. It is hoped that much of this sort of nuisance will be eliminated through improvements in the next automatic programming system.

The dual-purpose installation has only one serious drawback, which comes about from the diametrically opposite nature of the work of the two groups when they are sharing one machine. By and large, commercial problems tend toward the routine, "chiseled-in-granite," repeating problems; numerical analysis problems tend to be on-demand one-time (or few-time) affairs. This can lead to questions of priority; fortunately for this thesis, there are questions of priority with any kind or kinds of work on any type of machine.

On the whole, however, it is felt that the advantages far outweigh these somewhat tenuous disadvantages. Attention should be focused on the possibility of adding numerical analysis work to a commercial installation and/or the attractiveness of a dual-purpose installation. In any installation involving commercial problems, the record-keeping (which is the essence of present-day commercial work) gets done; there may then be an opportunity to work on commercial problem solving. When that situation evolves, it may be quite a help to have a scientific group at hand. Under a common leader, the two groups become compatible and ideas cross-fertilize each other.

Unusual Problems and Their Solutions by Digital Computer Techniques

L. ROSENFELD

TEN years ago the primary role of an electronic computer was to develop mathematical tables and to solve the scientific problems which arose during the research and development work being done at that time for the military and other government agencies. Since then, and particularly of late, the emergence of the computer as a powerful and versatile tool for the management of most major business and industrial corporations in assisting them to carry out their normal business operations has been witnessed. These would include such standard operations as payroll accounting, inventory control, production scheduling, invoicing, billing, and cost accounting.

It is natural that company management should now seek other fruitful areas where the computer might be utilized for profitable advantage. By and large, with the possible exception of those companies which are large enough to afford to have on their staff a group similar to an operations research group, the places where management will seek additional areas of work for its computing facility will be chiefly limited to those areas which are normally under the cognizance of the comptroller or finance officer. Usually, management will not think in terms of using the computer as a device which could possibly aid in improving a company system or subsystem which is part of the way by which part or all of the business of the company is carried out. For example, management of a large trucking company probably would not use the computer to help determine the most economical time to replace worn-out vehicles though it might use it to bill its customers; similarly an automobile manufacturer probably would not use the computer to predict what his spare parts inventory should be for the next 10 or 20 years although he would use it to maintain his inventory.

This paper is essentially a brief description of three case histories of what would generally be defined as operations research

L. ROSENFELD is with Melpar, Inc., Boston, Mass.

problems. Basically, there were two reasons why it was written: One was to indicate to company management several examples of a large number of vastly different operational problems which were or are being solved with the aid of electronic computers. These problems initially would not have been considered by management as problems which would readily lend themselves for solution by electronic computation. For corporate management, then, this paper perhaps will serve the function of exciting its imagination to suggest other problems or other areas where the computer can be effectively used, and give it the necessary incentive and encouragement to do so.

There was a second reason for writing this paper, namely, to point out to the computer specialist and to the numerical analyst another area where the capabilities of the computer are limited by the state of the art, i.e., for want of the mathematical tools and techniques to solve the problem efficiently. By this is meant that in each of the three case histories which this paper discusses there did not exist any practical set of mathematically defined rules to improve an existing solution except by trial and error. This subject will be discussed in more detail in another section.

Problem I. The Determination of an Optimal Trucking Route Through a Given Traffic Congestion Pattern

This problem can be classified as a large-scale traveling salesmen problem. However, the enormous number of points involved prohibited applying any of the standard methods for solving this type of problem. A large trucking firm, X, with a fleet of more than 250 vehicles, contemplated relocating its main warehouse and dispatching depot from a southerly point, B, to a southwesterly point, A, of a metropolitan midwestern city, Y. An approximate annual savings of $20,000 would be realized because of the relocation. However, more than 60 per cent of the volume of business of company X was the transportation of goods to points in areas or cities to the northeast of city Y. The question arose whether the increase in cost both in additional time and wear of equipment from originating in their new location would more than wipe out the possible annual savings of $20,000.

A survey was taken by a Traffic Planning group which determined the time involved to travel from a point of intersection of two roads to another point of intersection. The total number of roadways considered included the following:

(a). Six dual highways which permitted commercial highways.
(b). More than 40 2-way streets.
(c). More than 130 1-way streets.
(d). Two toll bridges.

Consequently, if R_j is defined as one of these roads and P_{ij} as the point of intersection of roads R_i and R_j, more than 5,000 ΔT's were enumerated where ΔT_{ij} equals the time elapsed to traverse from P_{ij} to P_{ij+1}.

In the first approximation to this problem, only the additional cost to traverse the distance from A to B was considered. This reduced the problem by a factor of 100 and hence, a modified simplex method to obtain answers to this approximation was going to be applied. However, the minimal increased costs that were incurred to traverse the distance from A to B were more than $20,000, nullifying this approach completely.

Essentially, the problem was to determine the costs incurred to travel from A to points $D_1, \ldots, D_n$ which were on the perimeter of the city's limits since to travel from D_j to the ultimate destination was the cost common to both the old and new dispatch points. Moreover, the time involved to traverse from B to D_j was known from past experience.

After the initial failure, it was decided to approach this problem from a different point of view which was in essence a trial and error method. The basic problem then, since labor costs and, to a large extent, wear of equipment, were a linear function of time, was to find the minimum time required to traverse the pattern from A to D_j (for discussion, D_j will be considered to be D_1).

If every P_{ik} and D_j is assigned an x and y co-ordinate, a sequence of segments (a segment denoting the distance between two successive P_{ik}'s) will be a possible path for destination D_j (in this case D_1) if every segment of the sequence does not have either an x or y component in a direction opposite to the x and y components of the vector $\mathbf{AD}_1$.

If t denotes a measure of time, $S(t)$ is defined to be the set of points P_{ij} which can be reached by possible paths to D_1 in time t. For sufficiently small values of t, obviously D_1 cannot be reached so that in this case only those points P_{ij} of a possible path to D_1 that can be reached in time t_1 are considered. Hence,

$$S(t)=(P_{i_1j_1}, P_{i_2j_2}, \ldots, P_{i_rj_r})$$

where the P_{ikjk} are points of parts of a possible path.

Let $P^*_{ij}(t)$ be a point such that $P^*_{ij}(t)$ satisfies the following conditions:

(a). $P^*_{ij}(t)$ belongs to the set of points $S(t)$.
(b). $\mathbf{AP}^*_{ij}(t) \cdot \mathbf{AD}_1 \geqq \mathbf{AP}_{ij} \cdot \mathbf{AD}_1$ for all P_{ij} in $S(t)$.
(c). $l(AP^*_{ij}(t)) \leqq l(AP_{ij})$ for those P_{ij} satisfying equality in (b).

where $l(AP_{ij})$ denotes the length of path to go from A to P_{ij}.

If $P^*_{ij}(t)$ is redesignated by A_1 the problem of getting from A_1 to D_1 can now be considered. Proceeding as above, a $P_{ij}(2t)$ is obtained and redesignated by A_2. Iterating in such a manner an A_r is ultimately obtained which is, in fact, D_1. In effect by trial and error a sequence of paths which minimizes the objective function as a function of a time increment t has been determined. Of course, now the size of the increment can be varied to obtain a minimum-minimum solution for a given set of increments $t_1, t_2, \ldots, t_n$.

This solution was coded for and worked out on the IBM *701* for each D_j and for five different increments t. The weighted additional costs, the weighting factors being a function of the percentage of the volume of traffic destined for D_j compared to the total volume of business, were determined. The smallest additional cost for all points D_j and for the t's used was approximately $14,000 more than justifying the contemplated relocation.

Problem II. Baseball Forecasting

This problem is of interest to management from the point of view that there exists a large number of business and industrial problems whose solutions require the development of a strategy or a set of rules or criteria which can be applied in a given situation. Generally, the situation or problem varies as a function of time. Moreover, if this variation could be predicted on the average and a set of rules or strategy developed accordingly, then a greater expected return or achievement might result in contrast to just establishing a fixed rule which would be satisfactory for all situations.

Briefly, the problem of baseball forecasting[1] was whether a strategy could be developed which would enable one to wager on major league baseball results with a greater expectancy of winning than of losing. If one wishes to win one unit whenever his selection on a given game is the correct one, then he must risk an amount R units, $R>1$ if his choice has the greater likelihood of winning, i.e., the favorite team, and an amount r units, $r<1$ if his choice has the lesser likelihood of winning, i.e., the underdog team. The assigned risk costs are the handicappers' estimate of the opposing teams' relative

worth and these risk costs are assigned prior to the start of a game.

Major league baseball consists of two distinct leagues with eight teams in each league. Each team plays 154 games per season playing the other seven teams in the league 22 times, 11 of which are on the home field, while the other 11 are played on their opponent's field. The data which were available for analysis consisted of the results and associated risk costs for each of more than 6,000 major league baseball games played during the period of the 5 years from 1950 through 1954. These data were put on International Business Machines Corporation (IBM) punch cards and for each team, the following information was listed:

(*a*). Team t_i's opponent, i.e., t_j.
(*b*). The risk cost for team t_i.
(*c*). The game number.
(*d*). Whether t_i was the home team or not.
(*e*). The outcome of the game.

A number of parameters or variables which might play an important role in the development of the strategy were enumerated. Among these were·

(*a*). For a given game, whether a team t_i was the home team or not.
(*b*). For a given game, whether t_i was an underdog or not.
(*c*). The frequency of m consecutive losses for each team and for $m \leqq 15$.
(*d*). The frequency of m consecutive wins for each team and for $m \leqq 15$.
(*e*). The ratio of the average risk costs of team t_i is t_j to the number of times that t_i beats t_j at home games.
(*f*). The ratio of the average risk costs of team t_i versus t_j to the quotient. The percentage wins of t_i divided by the percentage wins of t_j.
(*g*). The ratio of the average risk cost of a team t_i to the percentage wins of t_i.

All in all more than 15 parameters were listed and analyzed but the foregoing 7 turned out to be the most significant ones.

Routines were written for the IBM *650* which tabulated a number of frequency or sample distributions of the various parameters as a function of either a team t_i the number of games played, the risk costs, etc. From these distributions a number of observations were made, and the important variables selected for future study. At this time a routine was written to determine the multiple linear regression coefficients of the least squares regression hyperplane, where the variable approximated by the linear form was the number of units won in a season. These coefficients indicated the relative importance of each of the variables each of which was acting simultaneously.

The next phase consisted in determining the strategy of wagering. It was decided that a decision for a particular game should be conditional on what had transpired previous to that game. The so-called system of progression was adopted whereby every time a particular team on which there had been a wager won, a unit would be won, and if the team lost, enough would be wagered the next time in order to (1) recuperate the previous losses since the last win and (2) win the initial one unit. However, it was apparent that under these conditions if no limit was set on this progression, there would be times when a risk as high as 500 units or more would be necessary in order to recuperate previous successive losses and to win the initial one unit. Consequently, a new variable had to be introduced, a so-called cutoff point which was the limit above which the progression would cease and it would be assumed that the successive losses were irretrievable. At this time one unit would begin anew to be won.

Twenty strategies were enumerated, each utilizing the significant parameters determined by the distribution tables and weighted accordingly by the multiple linear correlation coefficients. Each strategy was programmed for the *650* and the five years' history of games was wagered by the computer in accordance with the rules of the strategy. This was done for a number of different values for the cutoff point. It became apparent that a successful strategy was possible for certain values of the cutoff point. Moreover, confidence limits were established in order to assure that there would not be any ruin during the season's play even though the strategy eventually won out.

Statistically the results can be summarized as follows: Let G represent the number of units won or lost after a season's play where $G>0$ if units were won and $G>0$ if units were lost and suppose that the initial investment was 20 units. Then the probability function for G was

$$\begin{aligned} \text{pr}(-20 \leqq G<0) &= 0.30 \\ \text{pr}(0 \leqq G<20) &= 0.40 \\ \text{pr}(20 \leqq G<80) &= 0.20 \\ \text{pr}(80 \leqq G) &= 0.10 \end{aligned}$$

Of course, such a probability function indicates a successful strategy. Moreover, the strategy when applied to the 1955 baseball season would have yielded a G equal to 102 units, a return of more than five times the initial investment.

Problem III. Scheduling of Work Assignments of Pilots and Airline Hostesses

Flight crews' salaries and related expenses represent an item of substantial magnitude in the operating costs of an airline. The problem was to determine whether more effiicent utilization of the available manpower and equipment could reduce these costs. This problem consisted of three basic phases.

Phase I. Trip Analysis

Determining the most efficient and economical method to operate flight out of the various bases by taking into consideration the following:

(*a*). The number of crew members at each base, and cost of possibly moving them elsewhere as required.
(*b*). Equipment qualifications of crew members at each base.
(*c*). Route qualifications of crew members at each base.
(*d*). The advantage of combining trips, i.e., combining an outbound trip from a base to a layover station with a possible shuttle to a third layover station, and then combining with a return flight to the base.
(*e*). Expenses incurred by trips requiring crews to remain at layover stations (hotel, meals, transportation).
(*f*). The cost of training personnel to meet requirements prescribed by the schedule, e.g., the costs to train crews who are only qualified to fly *DC-3*'s to fly Convairs.
(*g*). The savings incurred, if any, by mixing equipment in flying schedules.
(*h*). Adequate rest provisions as defined by Civil Aeronautics Regulations and working agreements.

Phase II. Flight Assignments

Constructing the flying schedules for various types of flight categories, i.e., captain, copilots, navigator, flight engineer, stewardess, pursers, subject to the following constraints:

(*a*). There is a maximum number of hours that can be flown in any month.
(*b*). There is a maximum number of hours that can be flown in any week.
(*c*). There is a maximum number of hours that can be flown in any 24-hour period.
(*d*). There is a maximum number of consecutive hours of on-duty time that can be assigned.
(*e*). Flight time schedules are based on a 7-day period.

Phase III. Assignment Improvement

Improve the existing schedules constructed in Phase II taking into account the following:

(*a*). Adjusting schedules and portions of schedules disrupted by leaves, vacations, illnesses, etc.

(*b*). The probability that a percentage of flights will be cancelled in any month because of equipment failure, weather, etc.

Representing the problems in Phases I and II by a mathematical model presented no extraordinary difficulty. The former is, in essence, a linear programming

problem to determine the types of flights to be flown in order to reduce the cost for operating these flights. The latter is merely determining the probability distribution function for the various variables involved and producing a table which would indicate, based on a given assumption, what are the revisions in terms of the number and type of flights and flight assignments that have to be eliminated from the assignments already constructed in Phase II.

It is in Phase II where the most difficulty is encountered. The basic problem is simply this: The trip schedules represent the workload W, which has to be perfomed in any given month. To accomplish the task of Phase II, W has to be broken down into individual work assignments subject to the constraints which were enumerated above; moreover, in doing so, the total amount of layover time had to be minimized and the total number of hours assigned per month up to the maximum hours allowed by constraint (a) had to be minimized. The reason for this latter objective is that a flight crew member's pay is the sum of a monthly base pay and an hourly rate times the number of hours flown in that month.

The solution to this problem was also written and coded for *650* and the final results, particularly in Phase II, have not been obtained. Nevertheless, certain conclusions can be made even at this stage. First, an over-all improvement over the hitherto existing way of accomplishing this task will be of the magnitude of 5 to 10 per cent, a very worthwhile achievement considering the money and costs involved. Secondly, a great deal of this improvement is a consequence of the results obtained in Phase I and to some extent, Phase III. However, there was a decided lack of technique available which would enable the computer to improve in terms of reducing costs, etc., an earlier decomposition of the workload, W. More precisely, there was no systematic way to improve an existing decomposition by an iterative procedure which was in the realm of practicality and economically feasible. The only other course of action was to produce another decomposition and select the one which was more efficient than all the previous ones made. Surely, this is nothing more than an elaborate trial and error procedure.

Concluding Remarks

The solutions to the three problems which have been discussed were to a large extent elaborate trial and error schemes. No doubt, the computer serves a very important function in this respect for the basic operations research problem is, to a large extent, initially divorced from a mathematical model. Consequently, any method which assists the analyst in deriving a mathematical model for his problem is a step in the right direction. Nevertheless, it is also true that this primitive approach is relied upon to such a large extent because of the state of the art. The problems that remain as a challenge to the computer user are much more complex than those which are presently being solved. They will require a more efficient utilization of the capabilities and capacities of the computer in use today.

On the other hand, the management of large businesses and industrial companies should realize that, in less than a decade, the computer specialist has gone from deriving mathematical tables and doing standard accounting work to solving operational and managerial problems of enormous magnitude and complexity. If management is to derive a greater utilization from its computing facilities, it must broaden its perspective and give to the computer specialist the challenge that the latter is willing to accept.

Reference

1. The Application of Operations Research and Digitalized Computational Methods to Forecasting in Baseball (abstract), **Lawrence Rosenfeld.** *Journal,* Operations Research Society of America, vol 4, no. 1, February 1955, p. 125.

A Progress Report on Computer Applications in Computer Design

S. R. CRAY R. N. KISCH

THE subject of computers designing other computers has been a popular one for several years. This subject generally brings to mind Boolean algebra reduction or generation of design logic. This is a difficult problem which the authors of this paper have investigated only superficially, and is not the subject of this paper. Another aspect of computer development work, however, lends itself to mechanization and represents the greatest portion of the time, money, and manpower consuming business of developing a new computing system. This paper summarizes the progress which has been made to date in writing, debugging, and placing in production a general purpose computer program (ERA *1103*) for handling this portion of the development work. This is a program for processing the logical design engineer's work through simulated operation of the proposed equipment to the production of detailed wiring tabulations for manufacturing purposes.

S. R. Cray and R. N. Kisch are with Remington Rand Univac, a division of the Sperry Rand Corporation, St. Paul, Minn.

The mechanization program described in this paper necessarily is based on a particular computer building block and particular type of cabinet design. It is independent, however, of any specific computer and any logical design can be processed which uses the selected building blocks and cabinet structure. The program takes into account all of the physical as well as electrical factors in planning component placement and in computing wire lengths and cable paths in tabulations for manufacture.

The Building Blocks

The particular building block chosen for the design program was a 1-microsecond magnetic switch developed at the St. Paul laboratories of Sperry Rand Corporation. This element performs 3-level "and-or-not" logic and provides one bit of temporary storage in each package.

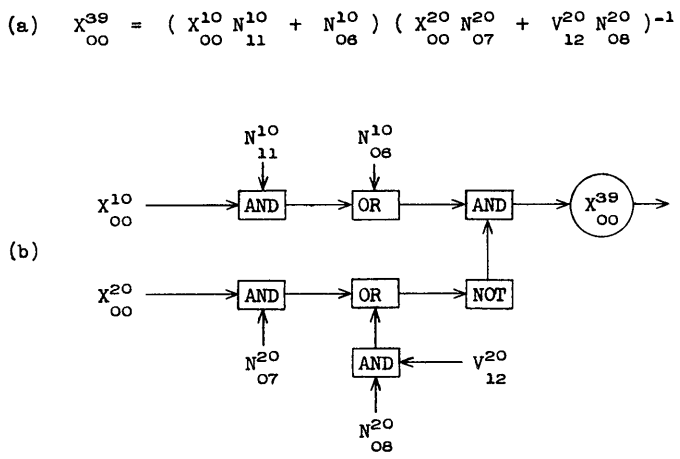


Fig. 1. Typical logical equation with diagrammatic representation

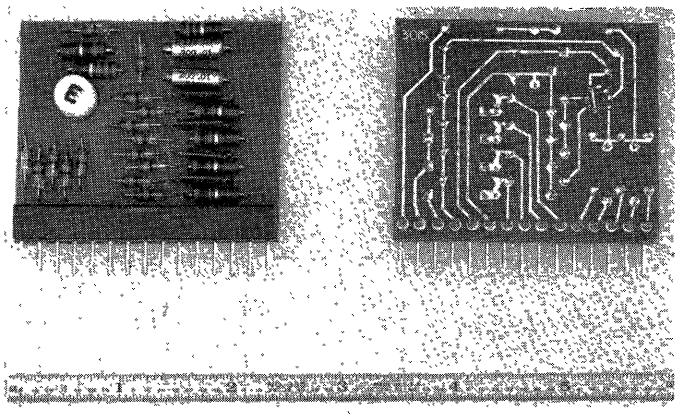


Fig. 2. Magnetic switch package

It is particularly well adapted to mechanization because of the simplicity of its logical structure and the inherent storage at each logical step.

Expression of Computer Logic in Equation Form

Communication with the machine is the first problem in utilizing a general purpose computer for design purposes. In this program an electric typewriter with attached paper tape reader and punch was chosen as the off-line communication unit. Seven-level paper tape is the medium for data transfer from printed page to computer and from computer to printed page. With this choice of input-output equipment, the design information must be reduced to symbols and combinations of symbols which can be typed on the electric typewriter.

A logical equation, representing the contribution of a single building block to the system design, is shown in Fig. 1(a). A base letter together with its two superscripts and two subscripts represents the signal from the unique building block. The equation represents the combination of signals from a number of building blocks which are combined in the indicated logical function to form the input to the building block symbolized by the left term in the equation. The same logical function is indicated by the schematic diagram of Fig. 1(b). The physical package which implements this function is shown in Fig. 2.

Communication between the building blocks is synchronized by a 4-phase clock source. To aid memory, the first superscript in each magnetic switch symbol designates the phase time at which the building block is read. This numeral, then limited to the values 0, 1, 2, and 3, is an aid in determining the timing of pulses occurring in the "and" and "or" circuits.

A Three-Phase Program

Phase One

In the first phase of a design effort using the magnetic switch elements the designer decides on an over-all general logic and then proceeds to generate a system of equations which describe how the building blocks should be connected to accomplish the desired operation. For a system of average size and complexity, from several hundred to several thousand of the building block packages are required; therefore, an equivalent number of equations must be prepared. This part of the program requires several weeks or months to perform. As yet no serious attempt has been made to mechanize the operation.

When this initial effort is completed the equipment designer is faced with a tremendous checking task. He is interested in examining the equations individually and collectively to ascertain that none of the combinatorial rules regarding numbers and relative timing of the inputs and outputs of each magnetic switch have been violated. He is interested further in learning whether or not the over-all performance of the system represented by the equations will be as planned. This task ordinarily is difficult and time consuming and the probability of detecting all of the logical and clerical errors is not much better than that of preparing all of the equations correctly in the first place. It is at this time that mechanized methods are brought into use.

A perforated paper tape copy of the equations is prepared by typing them on an electric typewriter-perforator. A typed copy is thus also obtained which can be used for reference purposes. The paper tape information is then entered into an *1103* computer, and a permanent equation file is established on one of the computer magnetic tapes. A part of the loading operation is a preliminary check of the equations to ascertain, for example, that each symbol in an equation is composed of the proper kind and number of characters. If an equation does not meet the format requirements, it is not entered in the magnetic tape file but is instead sent back out of the computer via punched paper tape. At the end of this initial loading operation the designer has a paper tape containing all of the equations having format errors. Corrections are then made, and the revised equations are entered and added to the magnetic tape file.

Next, the individual equations are subjected to more complete verification regarding the number and types of inputs and the relative timing of these inputs. It is a requirement, for example, that there shall be no more than four "or" inputs per magnetic switch and that each "or" input shall consist of no more than four "and" inputs. Furthermore all "and" inputs to each "or" must occur simultaneously, and there are restrictions regarding the timing of these inputs with respect to the read-out time of the switch. Improper equations are printed out on the monitoring typewriter attached to the computer. Equation 1, for example, would be rejected because the clock-phase numbers on the symbols X^{20}_{10} and N^{11}_{16} do not correspond. In this way many of the minor logical errors and transcription errors are detected. A number of computer programs are available for facilitating the alteration or replacement of equations in the magnetic tape file, so that the process of making the necessary corrections is made extremely simple.

$$Q^{00}_{10} = X^{20}_{10} N^{11}_{16} + Q^{30}_{11} N^{32}_{20} N^{31}_{02} \quad (1)$$

Since the number of outputs from a

Indicator Panel 002104

B^1	B^5	Q	F	O
000000000000000	000000000000000	000000000000000111111111	000000000000000	0000000
B^2	B^6	A	K C	I
000000000011100	000000000000010	000000000000001010101010	000000 00000	0000000
B^3	B^7	X		
000000000000001	000000000000000	000000000000000011111111	000010000000	0000000000
B^4	S	Z		
000000000000000	000000000000	000000000000000000000000	000000	0000000 00000000
G	P	U	D	
000000000000000	000000000000	010011000000000000000000	0000 0000	0000 0001

Fig. 3. Control panel showing register contents after simulation

single magnetic switch is limited by electrical and physical considerations a check must be made to assure that the designer has not used more outputs in his logical design than are physically available from a given switch. This check, which is again handled automatically by computer program, involves the scanning of the entire list of equations and noting where the outputs from one switch appear as inputs to other switches. The information relative to the number and destination of the outputs from each switch is recorded, for future reference, in the magnetic tape file along with the equations. Any illegal usage is immediately described on the monitoring typewriter. This process of verification is repeated, with intervening correction procedures, until a functionally sound set of equations is obtained.

Other computer programs are available which accomplish such tasks as sorting the equations by the various elements of the symbols identifying them and preparing printed copies of the equations and other information appearing in the magnetic tape file.

When a set of several thousand equations is considered it is obvious that the previously described procedures, when conducted manually, would require several weeks of effort. With the mechanized methods, several hours usually are sufficient to complete the task.

Phase Two

In the second phase of the program the designer has an opportunity for testing the logic of his design in a manner which simulates the method he would use if the equipment were actually constructed. The control panel for the proposed equipment is created, on paper, by laying out and identifying all of the push buttons and indicator lights which will appear on the actual panel. The push buttons include those used for setting and clearing the stages of arithmetic and control registers, those used for initiating the various sequences of operation, and miscellaneous operating controls. The indicator lights are those used for indicating the contents of the registers, the state of control elements, etc. The push buttons and lights which appear on this panel are incorporated into the design by adding special symbols to some of the equations of logic to represent the manual inputs and by the preparation of some additional equations to express the indication functions. For example, in equation 2a the symbol M_{56}^{-3} represents a push button which provides a manual input to stage 03 of the "X" resister. Equation 2b would be written to provide for the indication of the contents of the same register stage.

$$X_{03}^{00}=X_{03}^{20}N_{14}^{21}+A_{03}^{20}N_{16}^{23}+M_{56}^{-3} \quad (2a)$$

$$L_{56}^{-3}=X_{03}^{00} \quad (2b)$$

The superscripts and subscripts associated with the L and M symbols have a special meaning in that they represent co-ordinate locations on the control panel where the buttons and lights which they symbolize are located. Ordinarily most, if not all, of these special symbols and equations will be included from the outset, since the designer is aware of their function and probably has a fair concept of the control panel for the equipment he is designing.

To initiate the "simulation" of the various operations, the designer, using a special co-ordinate paper representing the control panel, marks the co-ordinate positions which correspond to buttons he would depress on the actual panel. He might proceed, for example, by entering certain operands into the arithmetic resisters and then initiating an operation such as "multiply." A paper tape is then prepared by typing on the electric typewriter-perforator a pattern of "1"s in which each "1" represents a depressed button, and its location, determined by the number of horizontal and vertical spaces from a printed index point, indicates which button is being depressed. The paper tape is entered into the *1103* computer, which contains the completed magnetic-tape equation file, and the information is automatically interpreted and recorded. The simulation is then initiated and involves the simultaneous solution of the system of equations for each clock cycle of the simulated operation. After a certain number of clock

WIRE TABULATION — Remington Rand Engineering Research Associates Division

DRAWING IDENTIFICATION: PREFIX | SIZE | NUMBER | REV.

TITLE
GEMINI
UNIT 10
INTRA-UNIT TABULATIONS

NOTE
THE TITLE, PREFIX, SIZE AND REVISION OF THE DRAWING APPEAR ON SHEET 1 ONLY.

SHEET 10 OF 21

ORIGIN	DESTINATION	COLOR	GAUGE	DESCRIPTION	LENGTH	CHG.
J10 H8- 2	J10 G9- 9	(56)			9-7	
	J10 L9- 1	(56)			10-2	
J10 E8- 1	J10 D9- 9	(44)			10-0	
	J10 C8- 1	(44)			10-0	
	J10 A8- 1	(44)			14-7	
J10 D9- 1	J10 L9- 9	(77)			12-4	
	E10 B-12	(77)			4-0	
J10 J8- 2	J10 L9- 2	(75)			11-6	
	J10 I9- 9	(75)			8-4	
	J10 H8- 3	(75)			9-6	
J10 L8- 2	J10 L9- 3	(05)			4-7	
	J10 K9- 9	(05)			6-7	
	J10 J8- 3	(05)			9-6	
	J10 H8- 4	(05)			14-0	
J10 N8-2	J10 M9- 9	(20)			9-7	
	J10 R9- 1	(20)			10-2	

Fig. 4. Example of automatically prepared wiring tabulation

cycles have been completed, an output is provided in the form of a paper tape. The number of cycles simulated may be predetermined or may depend on some selected criteria.

The paper tape output, when processed on an electric typewriter, provides essentially a picture of the control panel with each indicator light represented by either a "0" or a "1" depending on whether or not the light is on. Thus the contents of registers may be examined at the end of an operation and the performance of the "paper" computer evaluated. In this manner all of the operations of the computer being designed can be simulated with various combinations of operands and the complete set of equations verified. When operational errors are detected, sufficient evidence is usually present on the printed control panels to allow the designer to discover rapidly the logical error in the design. Then by using the modification facilities described under phase one he may make the necessary corrections to the equations.

It is also practical, in order to achieve added realism, to prepare some additional equations which simulate several registers of storage in the proposed computer system. Thus actual simple programs may be prepared and their execution simulated for the purpose of checking continuity of control in the new design. The added equations are, of course, removed from the file before further processing is done.

Fig. 3 shows a typical output display resulting from a simulation operation. As previously mentioned, the arrays of "0"s and "1"s represent indicators on the various arithmetic and control registers in the computer being designed. The registers are identified by the letters appearing over the indicators. Each panel is identified by a number appearing at the top, which is automatically advanced for every clock cycle of simulated operation. This number is preset to an arbitrary value at the time of entry of the paper tape containing the push button information initiating the simulation. Each type of operation performed is thus identified with a unique series of numbers. A by-product of this phase of the program is a record of each operation of the proposed system giving exact operation times and specific examples of resister contents before and after execution.

Phase Three

When the design checking process has been completed another formidable operation must be undertaken. Decisions must be made regarding the placement of the magnetic switch packages in the standard chassis assemblies, and manufacturing tabulations must be prepared which completely describe the wiring required to interconnect all of the packages. A standard chassis accommodates up to 180 of the building block packages so that in an equipment using, for example, 2,000 packages, 12 assemblies would be required. Indiscriminate or improper assignments of the various magnetic switch packages to the chassis would result in an excessive number of interconnections between chassis, intolerable lengths of wire on some of the switch outputs and possible excessive unbalance of the loads on the clock pulse driver lines. In view of the number of factors to be considered and the tremendous number of options available for placement, the assignment job is handled by the 1103 computer in a manner far more rapid than it could be done by the designer.

The same is true of the process of preparing wiring tabulations. Several hundred pages of material such as that shown in Fig. 4 are prepared in a few hours by the computer, and freedom from the various types of human errors is assured. The connection points are listed in an order such that subsequent wiring in that order will require a minimum length of wire. The length of wire required is also listed in each case and a color code is assigned. Additional manufacturing and maintenance aids such as component inventories and cross tabulations for signal tracing are also quickly obtained from the computer.

Simplicity of Utilization

In order that the use of the computer for design assistance be made as simple as possible, a method of interpretive programming for the execution of the various mechanized procedures has been worked out. A master file of all available programs is maintained on a magnetic tape which is placed on a computer tape unit at the beginning of a production run. The user is provided with a catalogue which lists an identifying code number for each program, describes its functions, and specifies prerequisite operations. Preparation for a run consists of typing on the electric typewriter a list of these code numbers in the proper sequence for accomplishing the desired task. The resulting paper tape is called the master program tape, and its contents are loaded into the computer at the beginning of the run. All of the subsequent operations are automatically controlled by the master program and the operator's attention is required only in the event of the detection of equation errors which must be corrected before proceeding.

The development of a computing system with this mechanized program requires the same logical design effort, on a system level, as any other approach. However, this method substantially reduces the time and money consuming process of detailed design and physical layout. Because of the great reduction in detailed design time, it is now practical to investigate completely a number of approaches to a system design. Several design approaches can be processed to completion so that component inventories and operating times can be compared for the completed equipments.

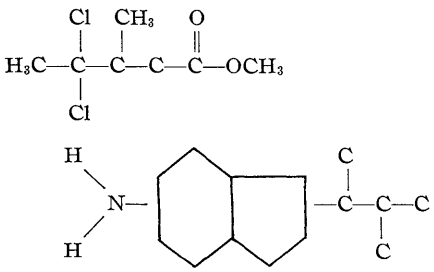
A Topological Application of Computing Machines

ASCHER OPLER

THE use of computing machines in the solution of problems arising in analysis and geometry is undergoing active exploration at the present time. These machines have been used for logical operations and have proved of considerable value. This report will describe the application of digital computers to an elementary problem in combinatorial topology.

The treatment given here will probably be of little interest to those interested in theory, but will prove to have considerable practical application. It developed as a by-product of an ostensibly unrelated research into the cataloguing of organic chemicals.

Organic chemicals, such as vitamins, insecticides, synthetic hormones, etc., are actually aggregates of complex molecules. To the organic chemist, they are more simply represented as single structures composed of atoms and bonds. The synthesis, the properties and the reactions of these compounds may be understood largely in terms of these structural diagrams. Thus, when the organic chemist wants a catalogue of thousands of chemicals, he visualizes collections of structures like:



There are over half a million of these structures known, and the list is growing at a rapid and ever increasing rate. Numerous questions arise in the daily work of chemists that make constant recourse to this catalogue a necessity. For instance: Has a certain compound ever been made before? What compounds have certain substructures in common? The members of a group have certain properties in common; what in the structures might be responsible?

Attempts have been made to devise mechanical aids to search these catalogues but have relied, in whole or in part, on chemical characteristics. Working with Dr. T. R. Norton, an organic chemist, the writer has attempted to solve the cataloguing problem by reducing it to one of topology alone and then solving it in this form.

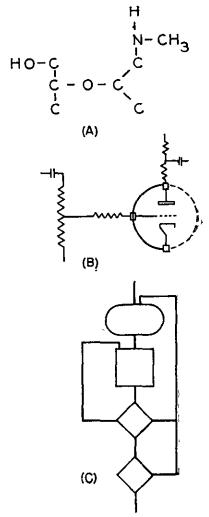
This has now been tested and proven feasible using digital computing machines. The method of transforming the chemical structures to topological networks will not be described here; only the solution of the corresponding topological problem will be covered.

The problem treated here concerns networks composed of members of various types containing a number of positions of potential attachment, some or all of which are used in the assembly of the network. The problem is to represent the topology of the network with a digital code and to search among collections of such codes to locate the networks which have given topological features. As a means of visualization, consider structures made from an Erector set constructed of flexible beams.

Practical uses, aside from chemical cataloguing, should be numerous. As an example close to the computing-machine field, the flow diagrams used to characterize computer programs fall into the category described. Computer charts may be reduced to codes and stored for later recovery. In stored form, they may be searched for common features and help the development of automatic coding and compiling systems. For those employed in electronic design of computers, the circuit diagrams also fall into this category, and the amusing prospect of computers storing their own circuit diagrams may be considered. With the storage of numerous electronic layouts, the search for common features should assist in the mechanization of production through more judicious selection of common subcircuits for automatic production of packaged plug-in assemblies. A third use in the computer field arises in the representation of connection patterns for analogue computers of the electronic differential analyzer type.

In general, the practical use of this method is indicated where large files of networks with elements used repeatedly in various arrangements are encountered. The extension to switching circuitry and communication networks is called for. Still another use might be in traffic engineering, where flow of vehicles and pedestrians can be catalogued and studied. In Fig. 1, a collection of familiar diagrams that are reducible to topological networks is shown.

Fig. 1 Typical arrangements which may be represented topologically



Elements of the System

Members

These comprise a finite set of specified types of objects representable as linear figures. For purposes of this representation, many branched and rounded figures may be converted to linear form. The choice of types to form the set of members will depend on the nature of the networks and the relative frequency of occurrence of objects of different types. In some cases, members will be only the most primitive elements; in others, simple combinations occurring more frequently may serve.

Positions

Every member must have a finite number of points of potential attachment to other members. These need not be equally spaced, but should be consecutively numbered from end to end. Numbering may be arbitrary or based on some generally accepted system.

Nodes

The conjunction of single positions in each of two different members is termed a node.

Substructures

An ensemble of two or more connected

ASCHER OPLER is with the Western Division of The Dow Chemical Company, Pittsburg, Calif.

members that form a portion of a larger network is termed a substructure.

STRUCTURES

A collection of members, properly connected, that represent an entire network is termed a structure.

Table I. Examples of Elements

Element	Organic Structure	Electronic Circuit
Member	Propyl group	1 mfd. capacitor
Position	2nd carbon atom	Grounded terminal
Node	Bond attaching a hydroxyl to 2nd carbon atom	Connection between a resistor and this capacitor
Substructure	Substituted alkyl group	Low-pass filter
Structure	Organic compound	Receiver circuit

Nomenclature

m = member
m^r = member of type r
m_i = the ith designated member
p = position
p_j = the jth position
p_{ij} = the jth position in member m_i
n = node
m-n-m' = node joining members m and m'
p_{ij}-n-$p'_{i'j'}$ = node joining members m_i and $m_{i'}$ at positions j and j'

Method of Representing Structures

OPEN STRUCTURES. [MEMBERS − (NODES + 1) = 0 (SEE FIG. 2)]

1. Designate any member as m_1, the initial member.

2. Select one of the other members attached to this initial one and designate it m_2.

3. Attach to the designation of the second member, the representation of the node attaching it to the first member $(p_{1_j}$-n-$p'_{2_j'})$ m_2.

4. Select a member attached to either of the two designated ones, number it and associate with it the node joining it to the other.

5. Continue in this manner until all members in the structure have been numbered and their member-node designation determined.

6. Use the solidus (/) to demarcate the member-node designations and a parenthesis to separate the node and member, e.g., $/m_1/(p_{1_j}$-n-$p'_{2_i'})m_2/(p_{1j}$-n-$p'_{3_i'})m_3$.

CLOSED STRUCTURES. [NODES − MEMBERS + 1 = Q; $Q \leq 1$ (SEE FIG. 3)]

1. Proceed as for open structures until it is necessary to close loops.

2. Close each loop by using a 2-node designation of the general form/(node) member (node)/.

3. Complete the structure, adding 1- and 2-node members as required. The total number of 2-node members equals q.

Discussion of Method

The principle of starting with any member might seem inefficient and calls for comparison with the alternative of starting with a specified member. This specification may be made on the basis of position (e.g., a terminal group) or on the basis of precedence (e.g., most highly connected member). In either case, multiplicities will lead to "ties" and special coding rules must be established to "break the ties." When searching is performed, these tie-breaking rules must be programmed.

After the first member has been designated, a second member must be picked. In any "systematic" scheme for selecting members, rules of precedence must establish which is to be designated next. This continues as each member is selected.

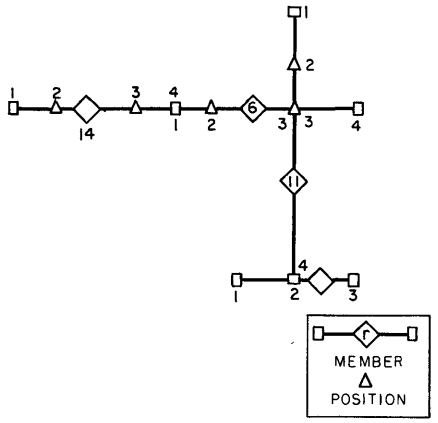


Fig. 2. Example of representation of an open structure

$$(\mathbf{m}_1^{14}/$$
$$/(\mathbf{p}_{1,4}\text{-}\mathbf{n}\text{-}\mathbf{p}_{2,1})\ \mathbf{m}_2^{6}/$$
$$/(\mathbf{p}_{2,3}\text{-}\mathbf{n}\text{-}\mathbf{p}_{3,3})\ \mathbf{m}_3^{11}/$$
$$/(\mathbf{p}_{3,4}\text{-}\mathbf{n}\text{-}\mathbf{p}_{4,2})\ \mathbf{m}_4^{2}/$$

Such a system requires that coding of structures be done by coders thoroughly experienced with the required rules of precedence. Machines to carry out elaborate searches must be programmed to use a comprehensive "rule book" to make decisions at every step.

Experience thus far with this free selection system has been favorable. For example, chemicals have been coded by approximately 50 coders without elaborate instruction. Speed of coding has been faster than for any other method studied. Even though the same structure could be coded in various ways, each chemical code has led back to only one structure.

Conversion of Representation to Digital Form

The preceding section describes the method of converting a structure into a sequence of member-node designations. In this section, the system for changing these designations into code numbers of fixed digital size is described.

Two-node member designations carry the most information, and therefore the digital code must be capable of representing them. At first glance, ten designations are present in

$$/(p_{ij}\text{-}n\text{-}p'_j)m_i^r(p_i{}^*{}_j\text{-}n\text{-}p''_{ij})/$$

but two may be eliminated as redundant (the i and i^* are identical to the i subscript attached to m) and two by the following useful convention: members which may serve in 2-node representations (closure members) must be attached only by their ends, which we may number 1 and ω. If the member is symmetrical, than 1 and ω are equivalent and we need not designate them. A special rule (1 is attached to the m_i with the lowest i, or, if $i' = i''$, at the node with the lowest j.) determines order of numbering asymmetric closure members. With these conventions, j and j^* can be eliminated. Thus it is necessary to designate only the following:

r = the member type
i = the member number
i' = the member to which m_i is attached
i'' = the other member to which m_i is attached*
j' = the position in m' at which m is attached
i = the position in m at which m' is attached†
or
i'' = the position in m'' at which m is attached*

An order of citation has been arbitrarily assigned:

x-x-x-r-x-1	(initial group)
$j'j$-x-r-$i'i$	(single-node groups)
$i'j''i''r$-$i'i$	(2-node groups)

The size of the number designation will naturally depend on the maximum number of digits that each of the six will require. This, in turn, is a property of the complexity and size of the networks under consideration. For example, if all j's range from 1 to 9, all i's from 1 to 99, and r from 1 to 999, 11-digit decimal numbers (1-1-2-3-2-2) will suffice for

*2-node members only
†single-node members only

representing any member-node combination. If these limits are 32, 32, and 1,024 respectively, a 35 binary digit number may be used.

In using these representations with conventional fixed-size word computers, the following has been found to be a useful order. Each structure makes up a block of computer words. Within each block, every member-node designation occupies one word. Each word then is divided into six parts with an arbitrary assignment of groups of digits to represent the required designations. In addition to these words, a first word identifies the block and gives the total number of words in the block. In our work, a second word is used as an internal redundancy check based on chemical principles. Thus, a typical structure appears as a block of $n + 2$ words where n is the number of members.

Details of Trials with Digital Computers

Earliest computer trials were made using the Datatron (Electrodata Corporation, Pasadena, Calif.). Here the word form was somewhat simplified and allowed 10-digit decimal words to describe the structures. These tests revealed the shortcomings of the original program and indicated that magnetic drum computers would be rather slow for the application, since nearly 3,500 logical commands were required to examine each 8-member structure studied. On the basis of these tests, the program was revised and improved, and a binary fast-access machine decided upon for further work.

The *701* Electronic Data Processing Machine of the New York International Business Machines Corporation (IBM) Data Processing Center was used for a group of more successful trials of the method. It may be of some interest to note that the programming, coding, debugging, and running of the program were all done through correspondence and wire communication. The results and methods were adjudged highly satisfactory.

The structures were all of the open type and were represented by blocks of ten full (35 - bit + sign) words, since they all had eight members. Nearly 200 of these blocks occupied the major part of the cathode-ray tube memory of the machine. In the testing

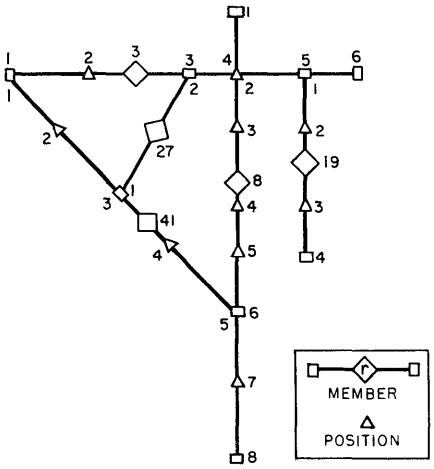


Fig. 3. Example of representation of a closed structure

$$/m_1^3/$$
$$/(p_{1,4}\text{-n-}p_{2,2})\ m_2^8/$$
$$/(p_{1,5}\text{-n-}p_{3,1})\ m_3^{19}/$$
$$/(p_{1,1}\text{-n-}p_{4,1})\ m_4^{41}(p_{4,5}\text{-n-}p_{2,6})/$$
$$/(p_{4,3}\text{-n-}p_{5,1})\ m_5^{27}(p_{5,2}\text{-n-}p_{1,3})/$$

stage, each structure was contained on a single binary card along with the necessary input, redundancy and locating words. (Later work will enter words stored in large "super blocks" on magnetic tape.) After the structures were loaded, the program described below was fed to the machine which was now in a ready status for searching for subnetworks. Each search was completely described in a single "criterion card" which was fed to the machine. Complete examination of nearly 200 blocks required 1 to 3 seconds before the printer listed the structure identities that met the criterion. Since neither the program nor the structure representations were destroyed, after this examination, the *701* was ready for more searches.

The program used in searching was designed to operate at the highest speed possible. Therefore, it called for the block-by-block examination of the structures. As soon as the first nonmatch occurred, the block was instantly rejected and the next block brought into operating position. Since there were a number of match tests, this shortened the time considerably. If a block remained in place and survived all match tests, the corresponding identity word was transferred to a block reserved for printing identities of successful structures.

Because of the latitude in the sequence of member selection, all matches had to be tried in all possible positions before nonmatch could be reported. Furthermore, since there could be considerable multiplicity, each criterion had to be tested against all nodes of a given type. Thus, if we require a member of type 13 to connect to a member 17 and there are two 13's and four 17's, we must make eight tests before we can definitely report no 13-to-17 attachment. The *701* program allowed for numerous types of multiplicity and for substructures of considerable complexity.

At the present time, work is starting to reprogram the task for a *704* computer, operating on approximately 15,000 structures stored on tapes. This program will allow for even greater complexity than the *701* program and will also handle closed structures.

Applications of Small Digital Computers in the Aircraft Industry

H. M. LIVINGSTON E. L. LYONS

THIS paper will begin by defining small computers. Just what are they?

The *650*, the Datatron, the Elecom, and others that were once regarded as small computers have now been promoted to an intermediate class by the introduction to the market of a much smaller group.

This new small class is comprised of two main types. One type is binary and has an internally stored program, and the other type is decimal with an external program. The first type is of the same philosophy as the large computers; a coded language and the same programming techniques are used.

The other type of small computer is decimal and uses an externally stored program. The second type has its instructions stated directly. The instructions do not have to be put into coded form. This is the type of small computer to which the paper will refer most frequently.

Charles Adams has classified computers as being large, small-large, and large-small. The first type just mentioned would be in the small-large category as it is similar to the large computer. The second type would be in the large-small category as it emphasizes simplicity and ease of operation.

Now, if one were going to write the specifications for a small computer for everyday work, just what would be emphasized? The computer should certainly be simple. If it is difficult to use, one might as well continue to solve one's problems with a desk calculator, or slide rule, or however it is being done now. It should also be easy to operate, so that no special training would be needed in order to use it. One also might like to have combined with this ease of programming and ease of operation, all the latest powerful features of the large computers. These are the objectives of the small computer manufacturers. Judging from the experience of the authors' firm we feel these objectives are being met.

The Burroughs Corporation had an interesting experience in its first contact with the aircraft industry. A section properties problem had been analyzed and it was felt that it was better for it to be left on their large computer because they could do in 1 minute what took us 15 minutes. They said, "The real problem is in communication and scheduling. We have to get a messenger to take the data to our computation center, then it has to be programmed, punched and verified, scheduled for the computer, run off, printed in a form that the engineer can interpret, and finally sent back by a messenger. That takes easily a day and sometimes 2 days. We're planning on having a small computer in each engineering design area, so they can do their small problems directly." That company, by the way, has ordered three more small computers since the first one was installed.

Section properties calculations present a classical problem that involves calculating centers of gravity and moments of inertia of irregular-shaped bodies. The operations to determine these on a small computer involve merely entering the successive points in the keyboard. One would go through the same motions one would go through in punching data into cards but with a small computer one would have the answers almost immediately after one had finished entering the data.

In some cases data may be supplied on punched paper tape or punched cards. For instance, telemetered data are received directly on tape. Another company with which the author's firm is working is transmitting missile test data by teletypewriter from its test station located on the East Coast all the way across the United States to its engineering center. There it will be fed into the small computer with a punched paper tape-reader attachment. The computer will reduce the data, print the results, and ultimately will punch a tape to be fed into a plotter. The equations involved in the data reduction are quite simple and a large computer would be wasting its time.

Many readers are no doubt familiar with this type of data-reduction problem, as distinct from the more complex type which involves numerous table look-ups and a large number of operations. This same company has a large amount of flight-test data reduction work being done. When the authors visited them about a year ago, there were over 100 girls carrying out the detailed steps on desk calculators. Much of the computational work is on their large computer, but the work these girls were doing is of a nonroutine nature that would require reprogramming the problem each time it was run.

The type of work sheets that is usually set up for girls to follow is familiar. There are several columns on a sheet, perhaps 10 or 20, and the operations to be carried out are shown at the top of each column. For example, one might have the numbers in columns 1 and 2 given, column 3 might be column 1 multiplied by column 2, and column 4 might be the square root of column 3, etc. This company is planning to have its computer girls, some of the same girls that are now using the desk calculators, set up their problems for the computer directly from the work sheets. The memory locations used could correspond directly to the column numbers and the operations are simply add, subtract, multiply divide—a language with which they are already familiar. When trigonometric

Fig. 1. General view of Burroughs E101

H. M. LIVINGSTON and E. L. LYONS are with the Burroughs Corporation, Philadelphia, Pa.

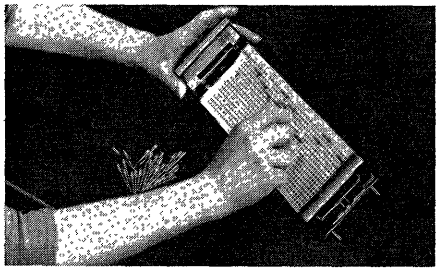


Fig. 2. The pinboard. Eight of these make up the flexible external program

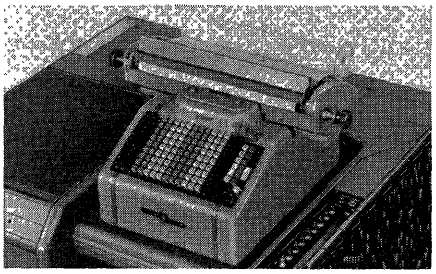


Fig. 3. The Burroughs Sensimatic flexible format input-output unit

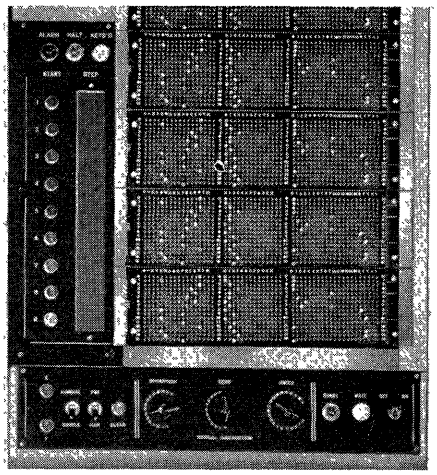


Fig. 4. The operator's control panel

functions, or any other subroutines are used, the girls would tell the computer to go to the subroutine, say sin x, after which it would automatically return to the next step in the problem.

A company near Philadelphia, Pa., inquired regarding matrix multiplications. It was set up on the pinboards in 11 steps and run. Two engineers from the company came in, learned to program and operate the computer, programmed their own problem, checked it out, and obtained a set of results in less than one day. The same thing could be done by using automatic programming routines on large computers, but it can be done on this small computer using its basic language directly.

Figs. 1–4, showing the Burroughs *E101*, will give a better idea of what is meant by small computers. This is quite typical, as far as size is concerned, of most of the small computers. As mentioned before, they all emphasize simplicity, ease of operation, and low cost.

Fig. 1 shows the entire computer not including the tape reader. The machine is about 38 inches deep and 60 inches wide, or about the size of an ordinary office desk.

The instructions or operations to be performed are set up on the pinboards on the right. There are eight separately removable boards, each with 16 steps.

The memory is a magnetic drum, and is located directly beneath the program unit. Its storage capacity is either 100 or 220 12-digit numbers plus sign.

Data are entered either through the full keyboard, or through an optional tape reader. The input-output unit shown is the Burroughs Sensimatic bookkeeping machine that has been modified for electrical input and output. With it there are all of the powerful format control features that are necessary for the bookkeeping operations under control of a separate panel that is mounted directly beneath and moves along with the carriage. It could be referred to as a format panel.

The electronic arithmetic unit and electronic control circuits are in the back section. This is one of the few machines that operates in straight decimal. For instance, the digit seven is represented by seven out of a possible nine pulses.

The power supply is on the left. It uses only about $2^1/_2$ kw and operates on the same power as most electric ranges.

The operator's control panel is on the right.

Fig. 2 is a close-up of a pinboard. The left-hand section specifies the operation and the second and third sections, the address or memory location. For instance, to perform the instruction "add the number in memory location 27 to the number in the accumulator," a pin is put under "plus" in the first section, a pin under "2" in the second section, and a pin under "3" in the third section. All of the instructions are equally as simple. A paper template is usually marked showing where the pins are to be inserted. The template is then placed over the board and the pins inserted where indicated. The paper templates can be filed for future use.

Fig. 3 shows the input unit. With the full keyboard it is as easy to enter numbers as it is in a desk calculator. There are four motor bars, all of which cause the computer to operate but have different control over the format. Printing is at the rate of 2 12-digit numbers per second. Continuous rolls of paper can be used or individual preprinted forms can be inserted from the front. Masters can also be prepared directly which simplifies reproduction of the results.

Fig. 4 shows the control section. The controls are very simple. Normally the operator presses the electronic clear button, then one of eight start buttons telling the computer where to start in the problem. The computer can also be operated one step at a time. It can also be set to manual operation and will perform that operation set up in the manual switches. The manual operations can also be under control of the push buttons on the keyboard. Indicator lights on the back of the operator's control panel are for checking circuits, also to show in what state the computer is. The power switch and associated wait and ready lights are at the right. That gives a general idea of what the small computers are like. Now some more of the applications that have been found in the aircraft industry will be discussed.

In Los Angeles, Calif., in the fall of 1955, a demonstration of conic lofting on the Burroughs small computer seemed to impress aircraft companies more than any of the other problems. The part of conic lofting demonstrated involved determining the coefficients of the equations of the curves formed by the percentage points between successive sections. It involves dividing an airfoil into percentage lines from the leading to the trailing edge. At each percentage line the co-ordinate points for the top and bottom of the airfoil are given along with various slopes and other known data. What is wanted is to obtain the coefficients of a curve representing that surface. There are 17 algebraic equations to solve and 8 keyboard entries. The total time for each point including the entry of data was 1 minute and 15 seconds. This was not too impressive because there might be from 50 to 100 per cent (%) lines per section, but the aircraft people were extremely impressed. Their delay at present is in scheduling the problem for their large computer. It takes at least a day before they get the results back from the computation center. Again, scheduling as well as communication is the big time factor. The conic lofting group, by the way, has the full support of their central computation laboratory to obtain their own computer because it is the small problems such as conic lofting that bog down the large computer. The preparation of the data alone exceeds the computation time on the large computer.

In December 1954, a session on small computers was held at the Eastern Joint

Computer Conference. Emphasis was placed on use of the small computer for checking programs for the large computers. That is actually being done now at the first installation of the Burroughs *E101*. They are doing sample calculations on the small computer to check against results from their large computers. They were previously performing the calculations by hand and found they were invariably erroneous. They found they were using the large computer for checking their hand calculations as it necessitated printing all the intermediate results from the large machine. With the simplicity of programming a person familiar with the problem can set it up for the small computer and obtain a set of check calculations in less time than it takes to perform one calculation by hand. The small computer is not being used to interpret the codes literally, as was predicted at the Eastern Joint Computer Conference, but rather to perform the check point calculations.

There is another way in which small computers are being used to aid the large computers. If, when formulating a problem one is not quite sure the equations derived will give him what he wants, one would perhaps sit down at a desk calculator, crank out the answers, make a change in the formulation, then try it again. Would not a small computer be a handy tool for this work? Many trial runs could be made with minor changes in the formulation each time by merely changing a few of the pins and pressing the buttons. Then once the problem is formulated, one could put it on a large computer and let it carry out the multitude of repetitive solutions. Of course, the same thing could be done by using automatic programming routines for the large computers, but there is still the scheduling problem. It is not economical to tie up such power equipment while one is essentially at a loss. One would certainly feel more at ease if one knew it was costing closer to $5.00 an hour than to $300.00 an hour.

The research group of a company in Philadelphia recently requested a small computer from their management. Management said, "What do you need a small computer for when we already have a large computer that will solve your problems?" The person from the research group replied, "That's just the trouble, we don't know what our problems are. We need a small inexpensive computer to help us in the formulation of the problems."

Now what the small computer can do for smaller companies as well as departments in large companies will be considered. Many small aircraft companies neither have economic justification for a large computer, nor the problems to keep it busy. Even though the large computer is more economical per unit of operation, a small computer would be better suited for these companies. There is an engineering company in Wilmington, Del., with less than 400 employees, including production and clerical staff, that is using one of our small computers for the design of arresting mechanisms on aircraft carriers. They are using it for many other engineering problems, but this is the major application. In the first week they had the computer they checked the hand calculations for a design that was ready for production. They found the plane would still be traveling at the rate of 30 miles per hour when it was supposed to be stopped. This could have been a costly error if it had not been caught until the final test. In one morning, for example, they ran off four of their design calculations. They estimated that these calculations would have taken 4 weeks by hand.

One of the first users of the small computer made a survey of the engineering problems in his organization. He found that 50 per cent could be done most efficiently on a small computer such as is being discussed, 45 per cent would be most efficiently done on an intermediate size computer, and only 5 per cent on a large computer.

In order to give a better idea of how people are planning to use these small computers, here is a breakdown by type of customer:

50% are companies that do not now have computers.

20% have large computers and are using the small computer independently.

30% are planning to use the small computer along with their large computer as an aid either for checking out problems, or doing parts of problems to be fed into a large computer, or just to keep the small problems off the large computer where they cannot be solved efficiently.

The breakdown by line of business is also very interesting.

30% are in the aircraft industry.
10% in business.
10% in banking.
8% chemical and pharmaceutical.
12% optical industry.
12% engineering.
6% research.
2% oil industry.

In summary, small computer manufacturers are emphasizing simplicity, ease of operation, and low cost. The major role of small computers in the aircraft industry appears to be:

1. To reduce the high percentage of communication and scheduling time associated with running small problems on large computers.

2. For the formulation of problems.

3. As an aid to large computers for performing check calculations and for checking programming techniques, and

4. For small companies and departments of large companies that do not have a work load for a large computer.

Traffic Simulator with a Digital Computer

S. Y. WONG

MANY mathematical models of traffic flow have been advanced in the past. Not all of these models are adaptable to situations involving complicated networks and control systems. A further difficulty is the insufficient number of specially trained traffic engineers who are capable of using these mathematical methods. In general, the design of traffic systems still involves much guesswork.

In the utility industry, as a result of the large investments involved, it is a recognized fact that expensive network analyzing equipments can be bought with full justification. Take the instance of water distribution systems, seemingly obvious guesswork solutions to increase pressure at certain points on a distribution network often turn out to be of little value when checked by a fluid network analyzer. The design of traffic systems, involving a much larger investment, is therefore in need of some form of mechanical computational aids in view of the present inadequacy of hand computing. The use of simulation methods with a digital computer was tried and presented here for the consideration by traffic and computer engineers.

Description of the Simulation System

No fixed pattern is necessary in the use of a digital computer for the purpose of traffic simulation. The following general description may serve as a broad outline and the example tested illustrates in detail a special case. The thoughts in this paper are directed toward highway and street traffic but its use can certainly be adapted to other traffic systems.

The first step is to divide the roadway into unit car-lengths (UCL). By UCL is meant the length of an average car. The space occupied by a rectangle one UCL long and one lane wide is called one unit block (UB). Each UB may contain one car or none. The cars can only move in a discrete fashion. For most purposes, it is sufficiently accurate to allow the cars to move an integral number of UB.

There is the question of simultaneity, which cannot be easily simulated with a general-purpose digital computer. It is practicable, however, to divide time also into discrete intervals and allow the cars to move one at a time, according to some previously assigned order, during that interval of time. If the time interval is short enough, an approximation of the real situation results.

The cars move according to their own behavior, subject to the constraints of the traffic control systems. Both the behavior and control system may be functions of location, time, and traffic conditions. In addition, the behavior of cars may also be functions of some random processes. In the case when both behavior and control system are functions of traffic conditions, there is a rather complicated interaction between the two.

The computing program may be divided into several more or less independent parts. In other words, each part may be changed relatively independently of the others.

1. Description of the Road System

This part can be handled similarly to an electric network,[1] once the road is divided into UB. Each UB may be considered as a junction and oriented lines join these junctions for all permissible car movements.

2. Control Systems

For a given road system, the control system can be varied independently to test the efficacy of a given control system.

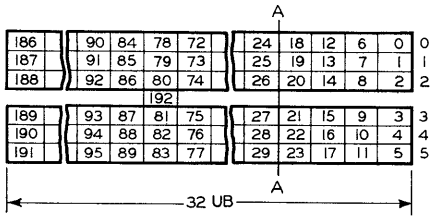


Fig. 1. Road section under study

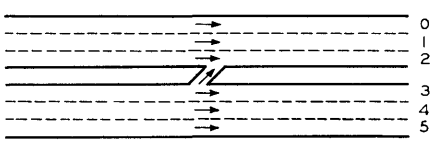


Fig. 2. Method of division and enumeration

3. Car or Driver Behavior

These can be obtained by psychological tests or by statistical methods. Since the behavior varies according to external circumstances such as weather, effect of holiday, etc., this part of the program should also be capable of being independently modified.

4. Data Handling

The final portion of the program is the main routine which sets the cars in motion and gathers the desired information. Because the desired information varies, it is obvious that this part of the program must be capable of being modified independently of the other parts.

The construction of the above parts of the program is admittedly a time-consuming affair. However, once it is available, traffic engineers can use this particular computing system similarly to a network analyzer. The advantage of a direct analogue cannot be denied.

Tests and Results

A very small example of the aforementioned system was tried on the Institute for Advanced Study machine. As a result of the limited time and effort available, the program was made for this special case, which is considerably less elegant than the clean separation described earlier.

The road system under test is a section of a 12-lane boulevard with six lanes for traffic in each direction. Fig. 1 shows the configuration under study and Fig. 2 shows how it is divided into UB and then enumerated so that it can be handled like a network. The UCL is chosen approximately 18 feet so that the speed of the cars assume discrete levels each 12.5 miles per hour (mph) apart.

From each UB, three directions of motion are permitted, these are diagonal left, straight, and diagonal right. Of course in the case of lane O, for example, diagonal left motion cannot be made because of the limit of the road.

The time step is chosen to be 1 second, so that when the velocity $V=0,1,2,3,4,\ldots$ UB, they correspond to 0, 12.5, 25, 37.5, 50, ...mph. The information about the presence or absence of a car and the behavior and records of the car is carried in a word stored at a location in the memory corresponding to the enumeration in Fig. 2. For each time step, the cars are moved one at a time in the order of enumeration, according to a set of rules. The set of rules can best be illustrated by

S. Y. Wong is with the Philco Corporation, Philadelphia, Pa.

The author is grateful for the helpful suggestions made by Lionel Rodgers and Leslie Williams, in matters of traffic engineering. William Keefe and the operating engineers at the Institute for Advanced Study (IAS) Computer Project, Princeton, N.J., greatly expedited this work. Dr. H. H. Goldstine's permission to use the IAS computing machine made the project possible. Many of the ideas in this paper were germinated as a result of the author's membership in the Citizen's Committee on Traffic Control of Philadelphia.

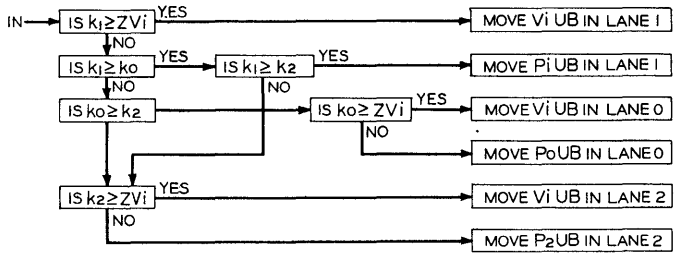


Fig. 3 (left). Program for rules governing lane 1

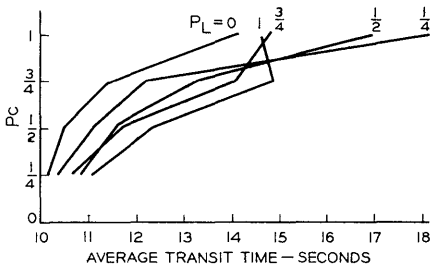


Fig. 4. Effect of left turn cars

the portion of program governing lane 1, which is shown in block form in Fig. 3.

At the end of each time step, a mechanism generates a random number x for each lane, and whether a car should appear or not at the input depends on whether x is less than a constant P_c or x is greater or eqal to P_c. Both x and P_c lie between zero and one. If a car should appear at the input of a certain lane, a similar mechanism decides what the intended speed should be according to some previously decided distribution. In this particular case, the distribution used for speeds of 25, 37.5 and 50 mph are 0.25, 0.5, and 0.25, respectively. A storage pool is provided for each lane; if a car cannot be introduced, this car will be introduced at the next time step. The waiting time in the storage pool is not counted.

The gap between lanes 2 and 3 is provided for cars in the right-hand section to be transferred to the left-hand section, and for that purpose only. It was assumed that cars intending to make the left turn at the gap would choose lane 3 initially. Therefore, as a special addition, cars that originate at the input of lane 3 go through a further decision process by comparing a random number x with a constant P_L to decide whether the car intends to make a left turn or not. When x is less than P_L, the car intends to make a left turn. For a left-turn car, no overtaking of other cars will be made, so as to make sure of reaching the gap. When a left-turn car reaches UB93, V is reduced to 12.5 mph until it reaches UB74.

Each car also carries its own clock whereby the transit time between input and the line AA is recorded. At the end of each time step, all cars that have traveled beyond line AA are counted and their clock examined so that a tally is made on a distribution plot. The cars beyond AA are then wiped out and a new time step initiated. After an arbitrary number of 512 cars has passed, the distribution plot is printed out and the average transit time calculated. In this fashion, one may plot the average transit time against P_c for various values of P_L, as shown in Fig. 4. Note that transit times for $P_c=1$ follow a different trend than other values of P_c. For the curve corresponding to $P_L=0$, that means no left turn is permitted. The forgoing study therefore gives a quantitative study of how much increase in transit time was caused by the left-turn gap.

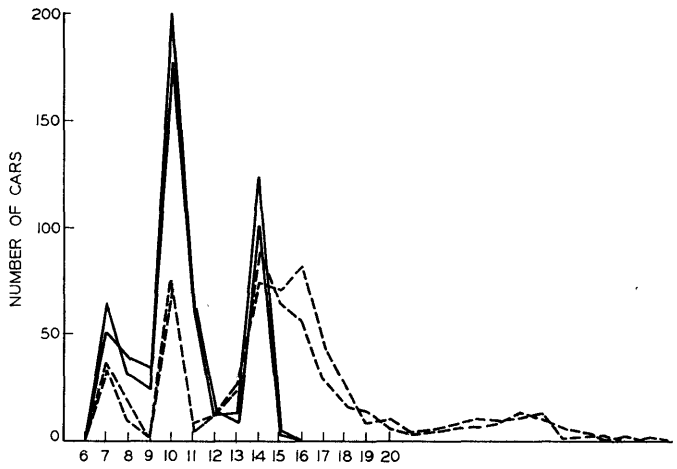


Fig. 5 (left). Comparison of independent experiments

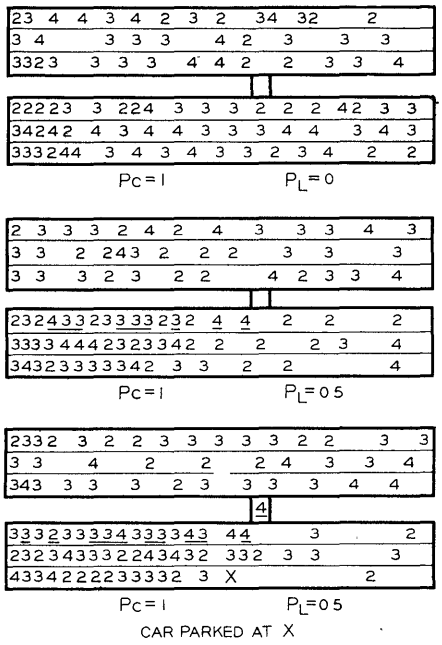


Fig. 6 (right). Typical traffic pattern

Numbers indicate intended speed
Left cars underlined

To check the variance between independent experiments in a qualitative fashion, the distributions of transit times of two sets of two experiments each are plotted in Fig. 5. It can be seen that the experimental results are not random.

Fig. 6 shows typical traffic patterns which are analogous to aerial photographs. Successive patterns then resembled a series of time elapse pictures. Figs. 7 to 11 are distribution plots of transit times for various values of P_c and P_L. As P_c increases, less and less cars can maintain their intended speed. By the use of these distribution curves, perhaps a "pleasure factor" can be derived to measure the ability of cars to maintain their intended speed on a given road.

The foregoing example is of course based on rather arbitrary rules of behavior and therefore can be viewed only in a qualita-

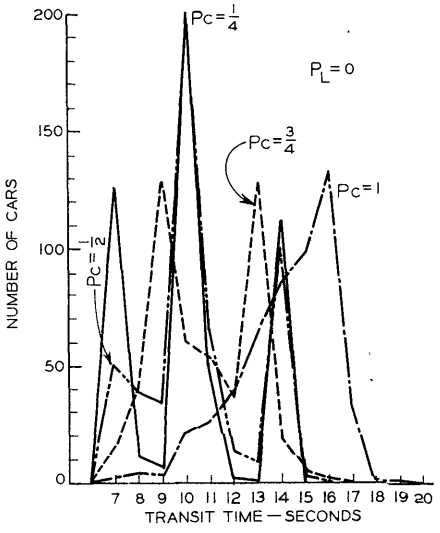


Fig. 7 (right). Transit time distributions

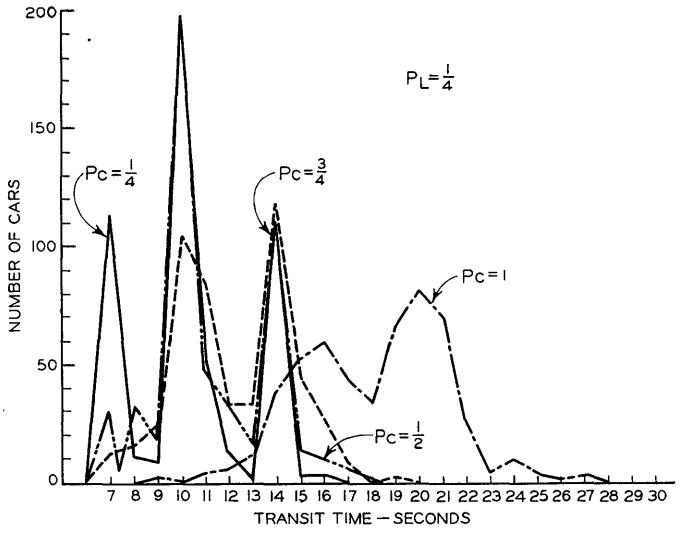


Fig. 8

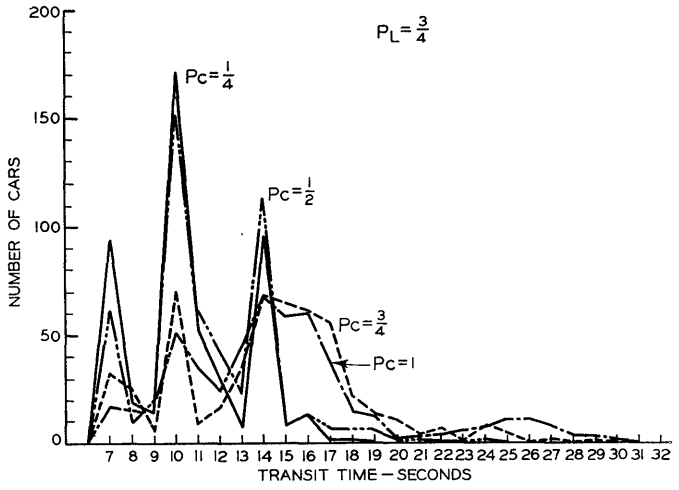


Fig. 10

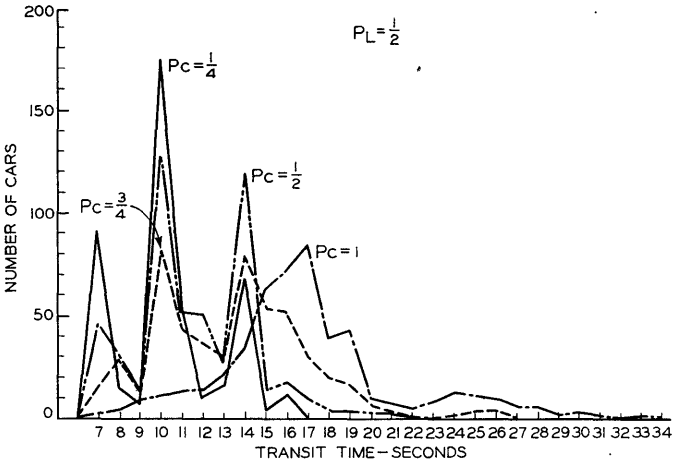


Fig. 9

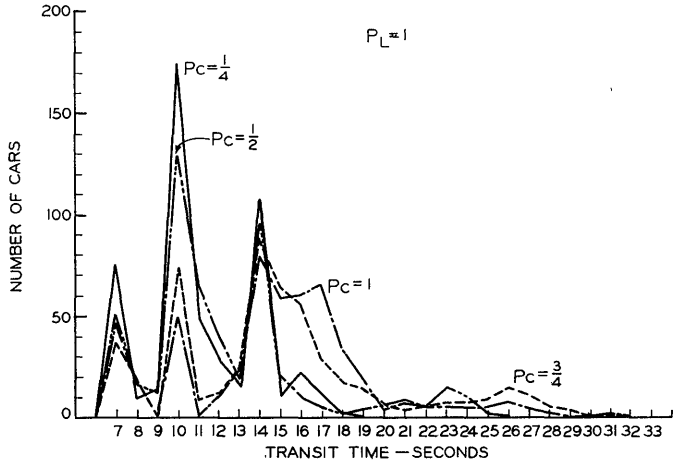


Fig. 11

Figs. 8–11. Transit time distributions

tive manner. There is no reason why a more realistic set of rules cannot be substituted. In any case, the traffic engineer can certainly make use of the type of information obtained to design a good road.

This problem was calculated with the machine running at a memory-access time of 125 microseconds, about one fifth of its normal speed. One time step (1 second) requires 1.7 seconds to complete. With the large capacity and high speed machines now coming into being, speed of computation required should not be regarded as a limiting factor.

Conclusions

This paper has shown, by way of an example, that it is practicable to program the digital computer to solve traffic problems using only elementary arithmetics. When properly programmed, the traffic engineer should be able to use a digital computer similarly to the way an electrical engineer uses the network analyzer. It is hoped that this paper will bring the computer engineer into closer co-operation with the traffic engineer for a better study of this important everyday problem of traffic.

References

1. Automatic Network Analysis with a Digital Computation System. **S. Y. Wong, M. Kochen.** *AIEE Transactions*, vol. 75, pt I, May 1956, pp. 172–75.

2. Analysis and Simulation of Vehicular Traffic Flow. *Report no 20*, Institute of Transportation and Traffic Engineering, University of California, Berkeley, Calif.

3. Esempi Numerici Processi di Evoluzione **Nils Barricelli.** *Methods*. Milan, Italy. vol. 6., no. 21-22, 1954, pp 45-68.

Integrated Data Processing with the Univac File Computer

R. P. DALY

THE Univac file computer is an intermediate computer which combines a large-capacity internally stored file of information with high-speed electronic computing. This information file is a "random access" file with sufficient speed and capacity to permit integrated data processing.

First, integrated data processing will be defined as it relates to the processing of data through a computing system. The term "integrated" means the complete or the whole function. As used in the phrase, "integrated data processing," it means the complete processing of the data involved in a business transaction to obtain the desired result and take all necessary action. In addition to obtaining the primary result and taking all necessary action, all the various business records affected by the transaction and the subsequent action taken are automatically posted and adjusted as necessary to reflect the current situation.

Now the requirements of a computing system capable of integrated data processing will be considered. As each of these requirements is looked at, the specific characteristics of the Univac file computer designed to handle each requirement will be described. To be capable of integrated data processing in business applications, the computing system must have extremely versatile input-output capabilities. Because of the diverse requirements of various business problems, it must be possible to communicate with the system on information media which best lend themselves to a specific operation. Commonly used media on which information is handled and with which the system must communicate include punch cards, magnetic tape and perforated tape. In addition to these prepared media, the system must be capable of "on-line" operation accepting data through directly connected manually operated devices such as electric typewriters and other keyboard devices. "On-line" operation of a computing system simply means the capability of the system to process instantaneously, random or sequential data from business transactions as they are passed to the system through some directly connected input device such as an electric typewriter or 10-key keyboard. On the other hand, "off-line" operation of a computing system involves the scheduled processing of batches of previously recorded data such as punched cards or magnetic tape. Information from devices such as paper tape perforators, and magnetic tape recorders, commonly used today for recording data, must all be acceptable to an integrated system.

In addition to this capability of working with any information-handling media, the input-output capabilities of the system must have common language versatility. This means that the system must be capable of communicating with any input or output device regardless of the language or coding, as it is called, of information peculiar to that device. Some examples of commonly used machine language which an on-line system must be capable of accepting are 5-, 6-, 7-, or 8-channel perforated paper tapes, 80- or 90-column punch cards. One more requirement of input-output versatility is the ability of an on-line system to work with multiple input-output devices at the same time and to make available simultaneously, on a time-shared basis, the data processing service of a central computer system to a number of input-output information-handling devices in various work centers of a business.

Each of these input-output features are necessary in a system capable of integrated data processing. Information in business applications is initiated from many sources. It is important, therefore, that no limitation be placed on the type of media on which information may be recorded. Each of the various sources of information will adapt itself best to one of the many types of recording media available today. Some will find a punch card the most readily adaptable means of initially recording a business transaction, others will find paper tape better suited as an initial recording media, and still others will find their requirements best suited by initiating business transactions directly into the computing system through directly-connected devices such as an electric typewriter or 10-key keyboard. As to common language versatility, data on business transactions may originate at remote points and be teletypewritten into the computing system; they may originate on punch cards which are forwarded to the computing center for processing, or they may originate through directly-connected keyboard devices in a third coded language. The system must be capable of accepting these various languages: teletypewriter, punched cards, or directly-connected keyboards which are peculiar to the particular communications link best suited to connecting a specific source of information on business transactions to the central computing system.

The capability of working with a number of input-output devices simultaneously on a time-shared basis is vital to successful operation of an automatic data-processing system in many business applications. Sources of business data are numerous. To avoid the time-consuming processes of data collection, data manipulation and scheduled data processing, input devices directly connected to the central computer must be available at the various locations where business data originates. In many cases the number of business transactions at a given location is such that several manually operated keyboard devices are required to process the data initially. If each of these keyboard devices is not connected on-line to the central computer, costly delays are again incurred in data collection, data manipulation and scheduled data processing. Many business operations require that the results of processing certain data be sent to a number of output locations. This may be necessary where identical business functions are being carried on in many different locations or where the results of processing certain types of business data must be continually furnished to a number of different locations to facilitate certain accounting and management controls.

The Univac file computer has common language versatility. There is no code restriction on the input or output data. The machine operates internally in Remington Rand's standard Univac Code, and a general-purpose code translator provides the common language link with various input or output devices. Some of the input-output devices which are available for use with the Univac file computer include 80- or 90-column punch card equipment, magnetic tape, perforated paper tape and key-driven devices such as electric typewriters and 10-key adding machines. As other devices now under development, such as character

R. P. DALY is with Remington Rand Univac, a division of the Sperry Rand Corporation, St. Paul, Minn.

readers and magnetic cards, become commercially available, these devices can be directly connected for operation with a file computer system. A large variety of input, input-output and output devices can be connected to the system and simultaneously time-share the computer. The file computer has ten programmable input-output demand stations. Each of these stations may have as many as 24 devices connected to it and scanned sequentially by the central computer. Multiple input-output devices can be connected to the computing system through these stations and share its computing capabilities up to the capacity of this intermediate speed computer. Six different program routines are available at each of the ten stations. The Univac file computer is a system with the complete versatility of input-output operation so necessary to accomplish integrated processing of the data involved in handling business transactions.

To accomplish integrated processing of business transactions data, a computer system must have adequate internal random-access storage with sufficient speed so that it can accept a particular transaction and completely process it, obtaining any desired end result. To process completely random business transactions, it is necessary to have available on immediate call from internal storage all of the business records which might be affected by the transaction and the action taken on it. The internal storage then must be of sufficient capacity to contain the necessary business records, and the access time to this storage must be fast enough so that it does not seriously impede the traffic on the system. It is estimated that an intermediate system such as is being considered today should be capable of processing approximately 100,000 average business transactions in an 8-hour day. The Univac file computer system meets this requirement with large-capacity random-access magnetic drum storage. Each large drum can store 180,000 alphanumeric characters, and additional drums can be added to the system as required to increase the total storage capacity to 1,800,000 characters. With special adapter equipment, this magnetic drum capacity can be expanded to almost 6,000,000 characters. The speed of these drums is 1,750 rpm, making the average access time to a selected business record approximately 17 milliseconds. This intermediate computer system can process an average business transaction in approximately 250 milliseconds, or 15,000 transactions an hour, or 100,000 transactions a day. In addition to this internal working file of business records, the system must be capable of storing large additional files of records. This additional storage is made available through magnetic tape equipment. Various blocks of records from this large external magnetic tape file can be called into the system's internal storage as required in handling various types of business transactions. These large capacity external files of business records add the necessary balance to the computing system. The magnetic tape provides an expandable storage media with tremendous capacity. As larger capacities of storage are required, magnetic tape units can be added to the basic system as one or multiple sources of blocks of business records.

Fig. 1 illustrates how data are handled by most data processing systems today. The author calls this "Off-line data processing." Information is collected as a result of business transactions and sent to the computing center. At the computing center the data is collected, batched, sorted, collated, reproduced, etc., to prepare it for processing on the computer. These operations, of course, require equipment, people, and time at the input. Periodically, as the necessary business data has been collected and manipulated, it is scheduled for computer processing and "spoon fed" to the computer. The circle marked "calculator" is the high speed counter, the computer. This is the device that must be used efficiently. Data must be collected for it, transmitted to it and scheduled for processing on it. After the computing operations are completed, the output data must again be manipulated in much the same manner as at the input, sorting, collating, reproducing, etc., as required, in order that the data can be usable for management reports, operation summaries, customer notifications, etc. These necessary operations, of course, require additional equipment, people and time at the output.

Many business data-processing operations can best be accomplished in this "off-line" manner. The sequential nature of their operation lends itself very well to off-line data processing. Many other business data-processing operations, however, are done in this manner only because equipment is not available to do them in any other way.

Fig. 2 is the same illustration as Fig. 1 with an additional box marked "large-capacity random-access storage." The addition of this one device makes it possible to consider integrated data processing for those operations where sufficient random-access storage capacity is available to store the necessary business records. Business data generated as a result of random business transactions, collected at the point of origin by an efficient information collecting device, as mentioned in the foregoing, and transmitted to the central computing system by means of capable communication facilities can now be processed in the random order in which they arrive with no collecting, batching, scheduling delays or data manipulation. Most business transactions because of their very nature are random transactions. With sufficient large-capacity random-access storage, all necessary business records can be stored

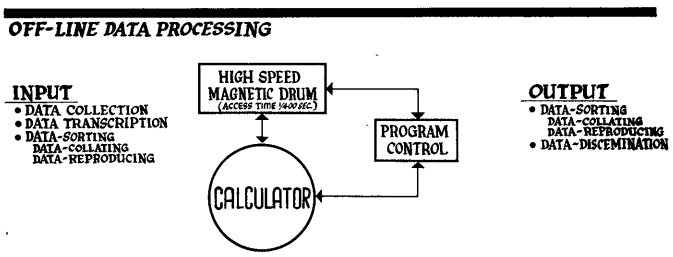


Fig. 1. Off-line data process system

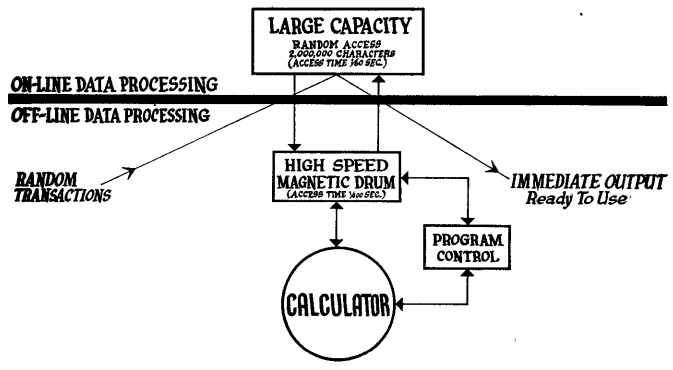


Fig. 2. Data process system showing large capacity random access storage

where they are immediately available on random call. As random business transactions are received and processed, the business records affected by the transactions are automatically up-dated.

With the ability to up-date business records automatically as a result of day-to-day transactions, the basic operation of file maintenance can be mechanized. Up to the capacities of available random-access storage devices, business record files can be maintained automatically so that they always reflect the current situation. This means that the many management reports now derived by periodic review and analysis of these files are always present. The inventory control, sales analysis, profit and loss, accounts receivable, accounts payable, etc., reports are automatically prepared, maintained, and always available on instant demand from the central computing system. As the action taken on a particular business transaction affects the ledger sheet, the inventory level, and the sales records, these records are automatically changed to reflect the current expenditures against a department budget, the current profit or loss condition, the current inventory level or the current sales volume. With complete and current report information automatically maintained and instantly available, an era of "reportless reports" can now be achieved. Management information "monitors" can be stored in the large random-access storage file and changes in various elements of operating information affected by the business data generated as a result of day to day business transactions, can be constantly and automatically "watched." Every time the inventory level of an item is changed, the new level can be automatically compared with the "monitor," the minimum reorder level. If the action taken as a result of a business transaction causes the current inventory level of any item to drop below the reorder level, an immediate alarm can be sounded. In any case, at the time an inventory report is requested, it is not necessary to prepare a complete report on every item. An exception report can be prepared automatically. The central computing system can analyze the inventory status of each item and report on just those items that have dropped below their minimum reorder level or do not meet some other qualification of the computer reporting program. This same technique can be applied to all of the filed operating data. Necessary management and operating reports are always prepared, always current and ready to use. The computing system can read out a report on any specific item or condition, a complete report or an exception report. The profit or loss condition is continually available. In fact, before a business transaction is consummated the forecasted results can be quickly reflected against the present profit and loss situation to determine if the transaction will be a profitable one or not.

The large capacity random-access storage of the Univac file computer with an access time of 1/60th of a second permits a large volume of traffic to be handled on the system. The storage capacity of these drums can be expanded in a standard system to contain up to 2,000,000 alpha-numeric characters and with special adapting equipment this "high traffic-random access" storage can be expanded to contain up to 5,000,000 characters as indicated in Fig. 2. Fig. 3 is the same as Fig. 2 with the addition of a box labeled "mass storage." This larger capacity of random-access storage is being developed for use with the Univac file computer. Mass storage units will have capacity for from 50,000,000 to 100,000,000 characters. The access time to this larger capacity storage may be 1 second or more; however, and, as such, might seriously limit the traffic on a system without the large-capacity, relatively short-access magnetic drum storage.

Notice in Fig. 3 how the three storage boxes or echelons of storage balance the flow of information to the calculator or high speed counter. The first echelon of storage must have fast enough access to feed the high-speed counter information at a rate fast enough to keep the calculator busy. The second echelon of storage can have a little slower access time but greater storage capacity to balance the flow of information. This second echelon of storage must have a fast enough access time to support the necessary traffic on the system. The third echelon of storage, of course, has very large capacity but relatively slow reference time. These three echelons of storage make up a balanced system designed to permit the necessary flow of information to use the high speed calculator efficiently.

To accomplish integrated processing of business transactions data, a computer system must have adequate program capacity and simplified programming techniques. To process data on many different types of transactions from many different sources; to edit the data at the input for computer operations and re-edit the data at output to match the requirements of any specific output devices; to co-ordinate and control the multitude of diverse operations necessary to accomplish the posting and up-dating of business records affected by any transaction, these functions require a large capacity for program instructions. The programming process itself must be simple because of the amount of day-to-day program variations generated by the many diverse data-processing operations and the daily changing requirements of business operations. Much of the information required and the data processing necessary is in the form of unscheduled demands by management. These management demands for information each require a different type of analysis of the basic data recorded in the storage file of the data-processing system. Programming techniques must be simple in order that computer operations may be easily changed and new programs readily supplied to meet the special demands of day-to-day business operating requirements.

The Univac file computer is designed to use plugboard programming. Program instructions can also be stored on a high-speed magnetic drum. It is a 3-address-logic machine and, as such, programming techniques are just an extension of punched-card logic. One program step performs the complete operation of $A+B=C$. This complete operation involves obtaining the operands A

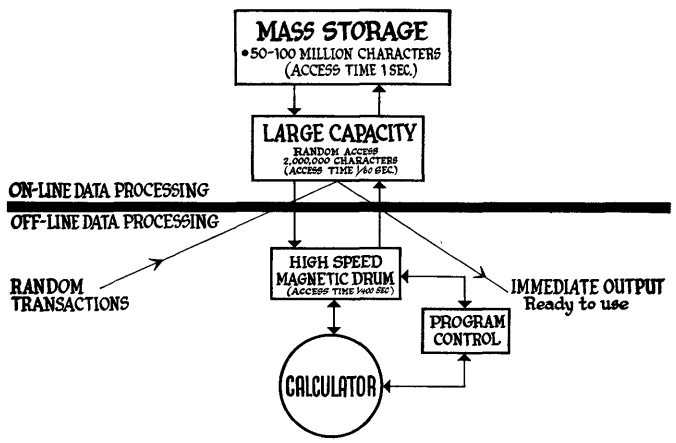


Fig. 3. Data process system showing mass storage

and *B*, performing the command function indicated and storing the result, *C*. "Plus," "minus" or "zero" branching is available on the result of any program step and "equal," "greater" or "less-than" branching is available on any comparison. Three-address logic offers the most direct and simple approach to programming business problems. The plugboard has proved through its years of use in punched card machines to be the most adaptable and flexible means of programming for business operations. Its versatility through the use of multiple boards, its ease of application, its flexibility, its ready availability for immediate use, these are a few of the reasons why a plugboard programming technique has been selected. The unscheduled non-routine demands of day-to-day business operations on automatic data-processing systems can be easily handled through plugboard programming techniques.

A plugboard of the Univac file computer has 48 program-step positions and a large complement of selectors to vary the use of these steps as a specific program requires variations. The selectors are particularly useful in handling exceptions and obtaining data from alternate storage locations. In addition to the 48 program steps on the plugboard, a small high-speed drum is available for storing instructions and data. Up to 1,000 instructions and data can be stored on this high-speed drum. Each instruction contains three 3-digit addresses, two for operands, one for the result, and a 2-digit command code for a total of 11 digits. The plugboard is used as a command coding matrix when instructions are stored on the high-speed magnetic drum. The commands necessary for specific operations are wired into the plugboard and referred to as needed by the instructions. This large capacity for program instructions can be expanded even further by storing instructions in blocks of ten on a large-capacity magnetic drum. Up to 15,000 instructions can be stored on one large drum. A "block-transfer" command is available to transfer blocks of ten instructions at a time, from the large-capacity drum to the small high-speed drum where normally, several tracks would be set aside for this purpose. The block transfer permits storing of program exception routines which must be available but are only used occasionally on the large-capacity magnetic drums. The 1,000 word positions of the small drum can thus be more efficiently used for storing primary instructions and data.

The Univac file computer is uniquely well-adapted to the integrated processing of business data primarily because of its flexibility and versatility in storing and accessing information and in controlling the movement of data in and out of the computer. Two features of the machine which particularly enhance its flexibility and versatility in these functions are a flexible command structure and variable word length of stored data.

The command structure of the Univac file computer includes three types of return jumps, an unconditional jump and five input-output control commands permanently wired in as internal instructions. All other commands to be used by the computer are selected at the programmer's option and wired into a plugboard. This philosophy gives the computer a truly general-purpose command structure with a pluggable command decoding matrix, the plugboard, capable of being wired up at the programmer's option with up to 48 command positions. Only those commands necessary to perform the data-processing functions of a given application need be wired up. This command structure flexibility permits maximum utilization of a programmer's initiative to tailor the computer control to specific applications with a minimum of the compromises normally necessary in fixed "general-purpose" command structures. A simple exchange of plugboards alters the entire command structure of the computer, quickly tailoring it to various types of applications. Such functions as interest computations and sum of the squares are easily plugged up as subroutines and referenced by a single instruction on the plugboard. Floating point addition takes approximately eight program command positions on the plugboard and can be performed by the computer as a single instruction in 40 milliseconds.

The word length of information stored on the large-capacity general storage drums of the Univac file computer is program variable within any routine in multiples of 12 up to 120 characters with the exception that the unit record length of 108 is not available. Any of the large-capacity general-storage magnetic drums or for that matter, any track on a drum, could have stored on it nine different types of information each with a different unit record length. The unit record length desired in a program routine is specified by the highest order digit of a 7-digit drum address.

In addition to variable-unit record length of stored information, the field structure of a given unit record is completely variable and specified by program control. This variable field structure applies not only to the variable size unit records of the large-capacity general-storage drums, but also to the 120-character unit record tracks of the high-speed drum. Any unit record can contain from 1 to 20 designating fields of from 1 to 12 characters in length. The field pattern of a unit record on either the large capacity general storage drums or the high speed drum is determined by block transferring a prestored field designation pattern to the field designation register of the large capacity general-storage drum or the field designation register of the high-speed drum. Automatic editing of input and output data is easily accomplished through program control of the field designation pattern of input-output storage.

The integrated processing of business data requires a complete data-processing system. Furthermore, this system must be flexible and versatile enough to be tailored easily to specific applications. The Univac file computer qualifies because it is a system of equipments the heart of which consists of intermediate-speed arithmetic equipment to which is connected control equipment made completely versatile by means of a flexible plugboard-defined command structure. To this "central computer" is connected on one hand, the particular type of storage best applied to a specific application large capacity, random-access magnetic drums, magnetic tapes, punched cards, etc., and on the other hand the input and output devices which best match the data processing requirements of the application: 80- or 90-column punch card readers and punches; 5-, 6-, 7- or 8-channel perforated paper tape readers or punches; 10-key or multiple key numeric or alpha-numeric keyboards; on-line or off-line printers; magnetic tape readers and recorders, and similar devices.

A Fixed-Program Data Processor for Banking Operations

J. GOLDBERG

ERMA is a large, special-purpose data-processing system designed by Stanford Research Institute for the Bank of America. The system includes both paper-handling and electronic data-processing functions, and its purpose is the keeping of records for commercial checking accounts. The first model will serve 32,000 active accounts and later will be extended to serve 50,000 active accounts.

In addition to keeping records and handling the paper checks and deposit slips, the system must reject overdrafts and stop-payment items, must provide the branches with a wide variety of random and scheduled information, and must proof-check the flow of data into the system. In order to meet the stringent time schedule of the system, the ERMA computer must do its work on-line. The computer was designed to fit the requirements of the system and, as a result, it is a highly specialized machine.

In its physical construction, the computer resembles modern digital machines: It stores data on magnetic drums and tapes, on keyboards, on punched paper tape, in relays, and in vacuum-tube flip-flops; its data are alphanumeric, are represented by binary-coded decimal digits, and are subject to arithmetic operations in an electronic adder.

In its logical organization, it bears little resemblance to digital computers of general purpose, either in addressing, in the structure of its central control, or in the role assigned to the arithmetic unit. The machine is permeated so deeply by the consequences of its special external requirements that it is of interest to trace the evolution of its logical design from these primary conditions.

ERMA actually contains four separate subcomputers, which operate simultaneously in the areas of magnetic drums, magnetic tapes, paper tape and a line printer. Each subcomputer has its special circumstances, and although all have their programming wired in, they differ substantially in their realization. The examples given here will be drawn only from the magnetic drum posting subcomputer, and will be used to illustrate the evolution of the following features:

1. Input items are processed using several large files stored on magnetic drums, and only a very small amount of fast-access storage is used.

2. Instead of having a single control unit performing operations serially, the work in the drum area is done by simple, special circuits, working simultaneously on different parts of the drum.

3. The programming for each mode of operation is wired into the central control circuits, and the wiring is switched as each mode is called up.

Description of Basic Operations

The primary task of the machine is to keep bank records for commercial checking accounts. The raw material for the data-processing function is a huge unsorted mass of paper checks and deposits, each addressed to a particular account. The end products are the familiar monthly statements for each account, complete with service charges, and a host of special lists and control totals for bank use.

Fig. 1 shows the major components and operations. Fig. 2 shows the drum posting area in greater detail. Five operators receive batches of customers' checks and deposit slips, and enter the items into the machine from their keyboards. At each board, the account number is read automatically from the paper to the keys and the operator depresses the dollar amount keys. The operator then presses a key instructing the machine as to the operation to be performed on the particular item on the board; for example, debit entry, stop payment posting, balance print-out, etc. ERMA automatically sweeps through the boards, processing the data on one board at a time. In the usual operation of debit entry, an incoming document, addressed to a given account, is subject to an acceptance test. If it is not rejected, it is posted; that is, it is subtracted from the "balance" file maintained for all accounts on the drum, and recorded separately on a special drum file. Later, it is sorted out for storage in the magnetic tape file kept for its account. At suitable intervals, service charges are calculated and posted, and statements are printed. Each day, special lists and balances are also issued for use. Also, the machine may be quizzed at random intervals concerning any customer's balance.

One of the basic requirements of on-line operation in ERMA is that the operators must have quick access to a large amount of information on four separate files. The "hold" and "stop-payment" files are relatively small, containing less than 1,000 17-digit words, but the file of all customers' balances contains over 50,000 10-digit words, making a total of over 500,000 decimal digits. Having these files available makes it possible for the operators to check each item immediately for overdrafts and stop-payments, and to provide instant information about the status of each account. Furthermore, having a separate record of each balance provides an excellent crosscheck on the transfer of items to permanent storage on the magnetic tape files. In order to provide a true check on overdrafts, the balance file must be continually updated by those debits and credits which are accepted for posting. Further, it is desirable to be able to post or remove hold items and stop-payment orders at any time.

All of these operations must be performed with perfect accuracy. In practice, this means that most transfers between storage media must be double-checked by repeated access. In order to keep up with the heavy flow of input data, the computer is allowed only 0.2 second to process each incoming item. This requirement of repeated access to such a large file in such a short time clearly leads to the use of magnetic drum storage. In ERMA, two large drums are used, to which the access time is relatively high (i.e., 33 microseconds maximum).

It is at this point that the ERMA computer breaks with general computer practice in that the logical design of the machine is tied very closely to a relatively slow-access memory. To the designer of a general-purpose computer, this dependence on a slow-access memory may be surprising for it is usually the character of the memory that has the greatest effect on machine structure. In a general-purpose machine, a slow drum is used only as a large back-up store, which occasionally delivers a block of data to a rapid-access working store. In this way, the slow-access memory need exercise only a secondary effect on machine design. Also, the usual general-purpose practice is to use a single arithmetic unit with the rapid access store, to perform many operations in sequence.

In ERMA, however, there is not enough time permitted to perform the repeated operations required on the "current-

J. GOLDBERG is with the Stanford Research Institute, Menlo Park, Calif.

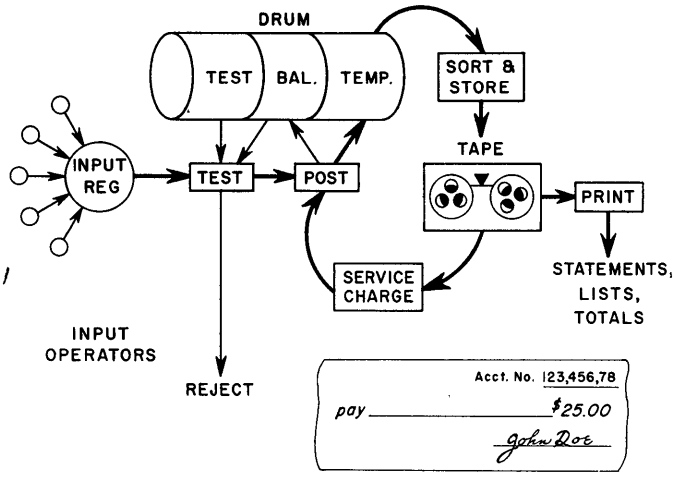


Fig. 1. Magnetic drum posting sub-computer

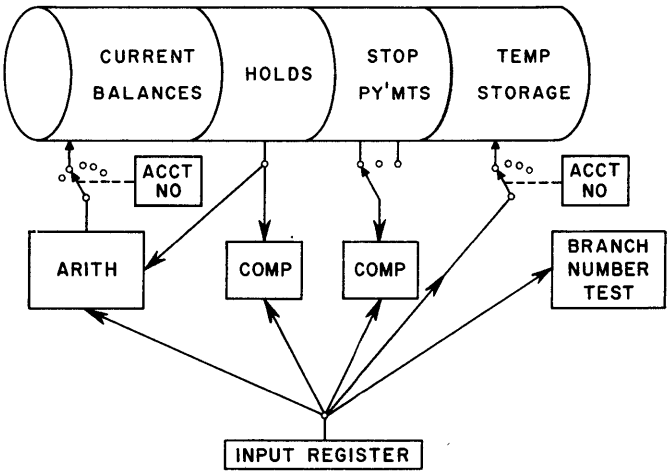


Fig. 2. Drum posting area

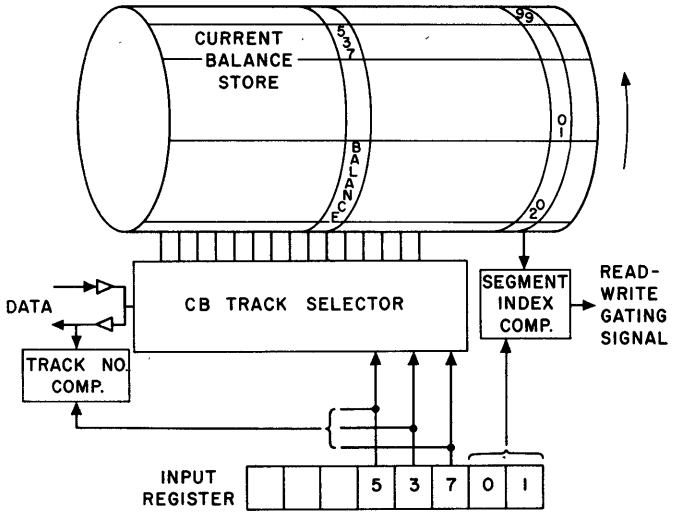


Fig. 3. Addressing method for "current balance" section of drum

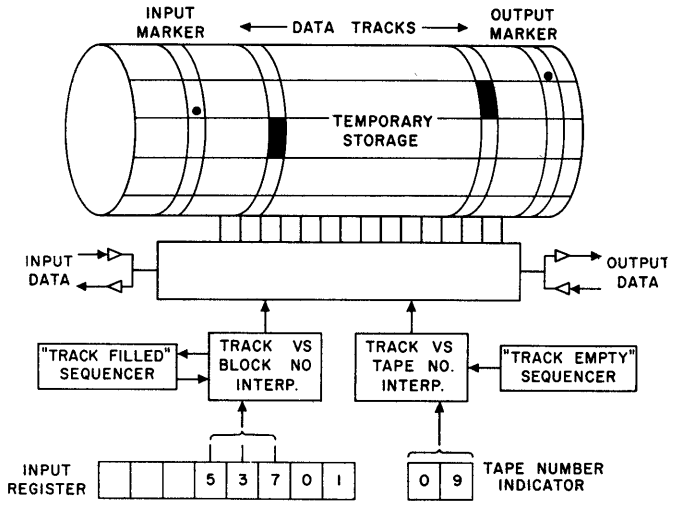


Fig. 4. Addressing method for "temporary storage" section of drum

"balance," "hold," "stop-payment," and "temporary-storage" files, all in sequence. This does not mean, however, that separate arithmetic units are needed for each file. The operations on the current-balance and temporary-storage files are quite simple, and may be accomplished by very simple circuits. On the other hand, because of the randomness of the data in the hold and stop-payment files, the operations of searching and selective extraction of items belonging to particular accounts are performed much more efficiently by a specially-wired search unit than by a programmed arithmetic unit.

The net result is that ERMA's arithmetic unit is used only to add and subtract dollar values, and the remainder of the work of the computer is performed by a number of simple, efficient circuits, operating simultaneously on the asynchronous data of the various files. Of course, each task could be done by a separate arithmetic unit and fast memory, but at prohibitive cost.

The following examples will describe some of the individual operations.

The Hold File

A word in the hold file consists of an account number, sign, symbol, and dollar amount, and every word in the file must be compared with the item in the input register. If the two words are identical, the entire process terminates. If the account number is the same, the dollar amount must be added to a running total of hold items.

These tests could be accomplished by transferring the file to a fast-access memory and applying the tests repeatedly using a programmed arithmetic unit. Even when the one revolution required to read the file from a drum into a fast-access memory is ignored, the application of a typical program of standard instructions would, by conservative estimates, require a clock rate roughly ten times that of the ERMA computer (152 kc).

In ERMA, the hold file operation is performed simultaneously on four independent drum tracks by a special comparator unit together with an arithmetic unit for the summation of "hold" values; the operations on each word are performed directly on the word as it is read on the drum, and the entire search takes one drum revolution. The comparator itself contains two flip-flops per track, with a moderate number of gate and buffer elements.

Current Balance and Temporary Storage Files

Posting a current balance consists of extracting one word from a specified address on the drum, performing three additions involving two words already in the arithmetic unit and the input register respectively, and returning the sum to the same drum address. Temporary storage

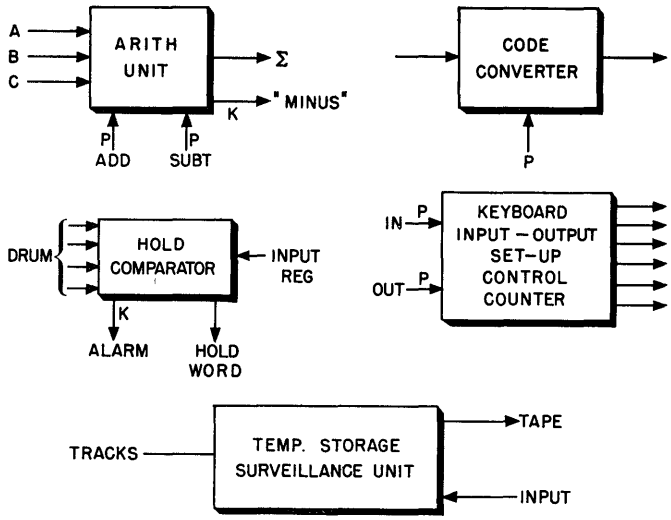


Fig. 5. Complex service units

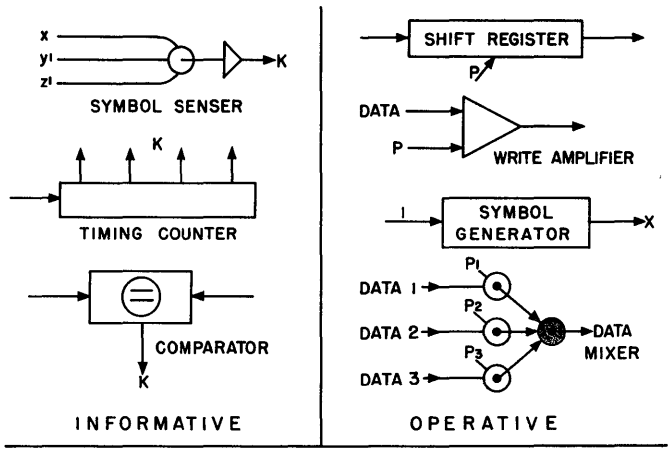


Fig. 6. Simple service units

P = control input
K = information output

posting consists of writing the input word in a specified track on the first empty space following a fixed-reference point. Both of these postings are followed by inverse operations for checking purposes.

It is true that a sequence of coded instructions could program these operations; however, for these functions the fast-access memory associated with it would be used primarily for operations on the program itself, rather than for the storage of data, and would not speed up the operation because the process is still limited by the need for random access to the files on the large drum store. Also, since the balance and temporary-storage files are asynchronous, two programmed units would be needed.

Fig. 3 illustrates the rather conventional addressing method used for access to the current balance (CB) section of the drum. The first three account number digits in the input register index a crossbar switch to connect the single "read-write" amplifier to the proper track (all drum data are serial-serial). The last two digits are compared with digits recorded on a coordinate track to produce a "read-write" gating signal for the desired balance word. The index number of each track are recorded on the track itself and is read as a check on the crossbar. Since only one track of the CB section is used for any one input process, the crossbar is set up only once in a routine.

Fig. 4 describes the addressing for the temporary storage (TS) section of the drum. The only thing of special interest temporary storage operations is addressing, since the incoming item is simply written on it its entirety, then read back for verification. In this case, the operation on the data is insignificant compared to the addressing problem. This store is the first stopping point for an item's record on the way to its particular magnetic tape file. It also serves as a medium for a sorting process.

Once the proper track is selected, the input word must be written in the first empty space found after the "band-start" point. This point is indicated by a movable pulse recorded on a special marker track. The actual track is selected by a special switching unit which simultaneously recognizes the status and demands of the input register, the drum, and the tapes. This unit assigns drum tracks to tapes on the basis of traffic load; it delivers filled tracks to permanent tape storage; and it channels incoming items to the proper drum tracks. Its internal structure is not important to an understanding of the drum-posting area.

Summary of Drum Addressing

To summarize the addressing situation in the drum area of the ERMA computer:

1. Fundamental requirements lead to the use of a large medium-access-time store.

2. Many processes must occur simultaneously. Because data appear asynchronously from different files, the access time is long, and there is not enough time allowed for sequential operation.

3. These processes may be accomplished efficiently by individual, special-purpose circuits which obviate the need for multiple arithmetic units and fast-access memories.

System Organization

Two basic facts have influenced the internal organization of the computer; first, many operations must occur in parallel, and, second, the machine must be able to switch instantly among many modes of processing data. Since the functions of control, arithmetic, and storage are included in almost all of the parallel working units, it is not too profitable to analyze the computer into these three areas. It is more instructive to divide the machine between "service" units and "program-control" units.

The service units perform the detailed tasks of transfer and transformation of data. They are specialized circuits, and quite varied, but their essential feature is that they act the same way in all routines. The function of the program-control units is to order the various service-unit operations in a manner characteristic of each specified routine. These routines contain the familiar minor cycles, the jump points, and the convergence points found in computer programs, but the program is actually wired into the control units. Since many operations are simultaneous, there are many data paths, so that a unique path must be specified by the control unit for each transfer. Also, drum data may appear with and without an account number, so that word operations may have different timing modules.

The drum-posting area has about 25 ways of processing the data in its files and input register. Examples are: debit posting, "stop-payment cancellation," "current-balance print-out," and "hold entry."

The basic operations in the typical routine are (1) transferring words among the various input and output registers, the internal electronic registers, and the various portions of the magnetic drum, (2) subjecting the data in the files and registers to tests of identity, relative size, and orders of position in files, and, (3) performing transformations on the data, such as arithmetic operations and code conversions on the whole, and on portions of a data word.

As described previously, some of these jobs are done entirely by specially wired

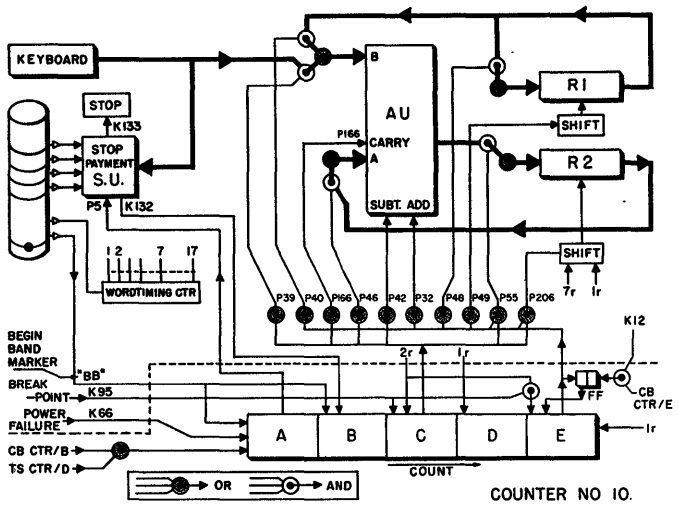


Fig. 7. Segment of drum-posting control set up for "debit entry" routine

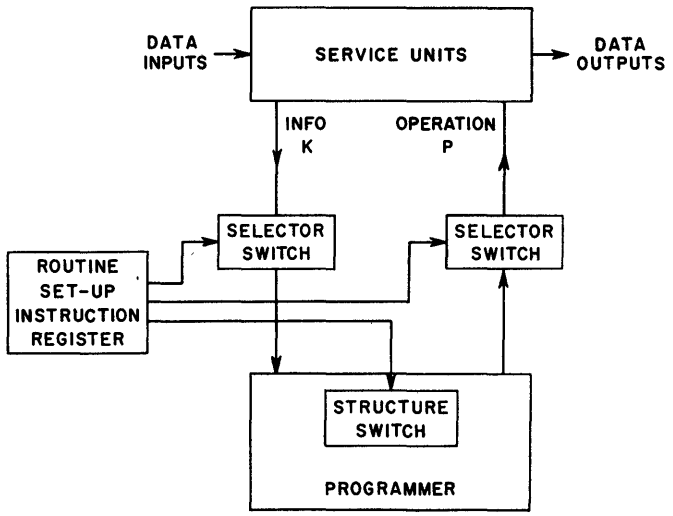


Fig. 8

subcontrol units. Examples shown in Fig. 5 are: the "hold" comparator and extractor, an arithmetic unit, a code converter, a keyboard input-output relay-control unit, and the temporary-storage surveillance unit.

Some simpler service units are shown in Fig. 6. Some, such as symbol sensers, comparators, and counters provide special information concerning the nature of signals observed; some, such as writing amplifiers, shift registers, and symbol generators, perform a special operation on data; some do both. It has been convenient to call channels carrying informative signals from service units "*K* leads," and those carrying instruction signals to the service units "*P* leads."

The Structure of the Program Control Units

The program-control unit is a sequential switching circuit consisting of flip-flops, counters, and logic elements prewired according to the prescribed program. It is driven by a set of input signals which are descriptive of the states of all of the service units and timing counters, and in response it produces a sequence of output signals which trigger one or more service units to perform their respective functions and specify data paths as needed. For operations performed by a complex service unit, the control unit acts merely to trigger or enable the operation. There are, however, various detailed operations which do not occur frequently enough to merit incorporation into a service unit. These are timed directly by the program control unit, with the result that the control has no steady rhythm.

Fig. 7 is a segment of the drum-posting control as it is set up for a debit entry routine. The following are significant:

1. The central control in this case (enclosed by dotted lines) is a ring-type counter together with a single flip-flop The arrows directed into a given counter stage are those conditions, taken in an AND sense, required to advance the counter into the indicated state. The counter is ring-like in that there is only one state on at once, and that the count proceeds in order from one state to its adjacent state, but the counter does not recycle. This counter is triggered to its first state by either of two other counter states elsewhere in the control unit. Thereafter it proceeds according to the *K* leads, timing leads, and interval control leads applied to its individual states. A state may last for from one clock cell interval (6.5 microseconds) to several drum revolutions. The timing pulses used in operating on the digits of a word are taken from a 17-state ring counter, synchronized with the drum.

2. The work of the states is done by energizing "*P*" buffers. Buffers are used to allow numerous counter states to energize the same service units. In this example, states *A*, *C*, and *E* are working states, while *B* and *D* are synchronizing states.

a. State *A* lasts for one drum revolution, by using the "begin-band" pulse as two successive advance conditions. It energizes *P5*, which gates on the "stop-payment-search" service unit to examine the SP file for the item set up in the input keyboard Search failure (*K* 132) permits the counter to advance.

b. State *B* serves only to delay the program until the word-timing counter cycles to the state suited for starting an arithmetic counter cycles to the state suited for starting an arithmetic operation.

c. State *C* energizes all "*P*" buffers required to subtract the keyboard item from the present contents of register 2 and store the difference back in register 2. Three of its loads establish the data paths used, and the remainder control the operations. The shifting of register 2 is precisely timed by its shift service unit over counts 8 to 1, inclusive, of the 17-state word-digit timing counter. The remaining control leads need merely overlap this interval roughly, so that state *C* may be triggered as early as the second count of the digit-timing counter.

d. State *D* delays the program until the digit-timing counter again reaches the second count, it provides a waiting state for the completion of some other concurrent operation as sensed by the *FF*, and permits break point operation if it is requested (*K* 95).

e. State *E* energizes all "*P*" buffers required to add the contents of register 1 and register 2, and store the sum in register 2. Note that some of the service units driven are the same used by state *C*.

3. The provision of individual advancing conditions for each state has several reasons and uses.

a. There is a need to intersperse long drum searches with single word operations of different modules in order to perform the required input item tests in the allotted time. This requires flexible timing choices.

b. Since several counters may be operating simultaneously, provision is needed for interlocking them at critical junctures.

c Break points may be assigned with great flexibility.

d. It is possible to apply feedback from the driven service unit to the driving states, so that the counter jams at the point of failure.

The program control unit has a different method of operation corresponding to each routine; each method has its own decision points, using the service units in some special order. As each new word enters the input register, the associated selection command causes a large switching to take place which essentially rewires large portions of the central control area, although many segments remain unchanged. The switching is done with relay contracts. The switching affects the control in three ways, as indicated by the three switches in Fig. 8:

1. It interconnects the components to

represent the logical structure of the operations to be performed (i.e., the decision points, the sequence of operations following each alternative, the convergence points, and the parallel operations).

2. It directs to each state the timing signals and service-unit status signals required for initiation and termination.

3. It connects each active state to drive one or more service units, with the following possibilities:

a. Several states may drive the same service unit through buffers.

b. A control state may either drive on a service unit directly, or may gate the output of one service unit to trigger another.

c. A control state may drive any number of service units; if a data transfer is required, the state may have to energize all gates required to complete the data paths when the service unit triggered is not restricted to a specific data path.

d. A control state also may do no external work, but act only as a synchronizing or delay state.

The counter shown in the foregoing example will be used in other routines. Some uses may be identical, but the counter may be pieced together with other counters in different arrangements. Some uses may be similar; for example, the work of one state may be inhibited by transferring its output bus to the zero level with a relay contact. In other uses, the majority of the input and output leads may be switched. An initial survey of the number of operations performed in consecutive groupings led to the switching of counters in blocks of four states each. The drum posting program unit handles its 26 routines with 19 counter blocks and 24 flip-flops. The longest routine uses eight counter blocks and 20 flip-flops.

Error-Checking Features

The fact that ERMA is a bank accounting machine and operates on-line means that errors are not merely inconvenient. Because all error cannot be prevented, and because ERMA must have all accounts in perfect balance by the end of the working day, errors must be found and corrected as quickly as possible. Since the best time to correct an error is when the paper document is in the hands of the operator, error checking should not be postponed. Because of the on-line nature of its operation, downtime must be minimized.

The following are some general techniques utilized in error-checking the drum-posting area:

1. Parity-check monitors are located at the output of all registers and drum-read amplifiers and may be switched onto commonly used busses. Each binary-coded decimal digit carries an even-parity redundancy bit.

2. Every addition or subtraction is followed by the opposite operation, using the sum read back from its permanent store.

3. All non-arithmetic postings, such as "temporary storage" write-on, are followed by reading back from the drum and comparing with the original items. If the source is a keyboard, the comparison is made using duplicate contacts on the key stems.

4. For certain lists, the keyboard is used as an output printer. Here, each key setup is verified by reading the key contacts back for comparison with the original source.

Summary

The ERMA computer is neither a stored program computer nor a plugboard computer, but it does contain some features of each. It is like a plugboard machine in that its program steps are wired in, and it is like a stored programmed machine in that each mode is called up by a coded instruction, which, however, is not subject to modification by the program.

It is a special purpose machine in that its detailed logical design directly reflects the external operational requirements. It needs a very large drum store to perform its operations on-line; the volume of data handled is large; many steps are required for processing each item and for accuracy checks; the data on the various files is essentially random, and the time allowed is short. Taken together, these conditions require performing numerous operations simultaneously rather than serially; each operation, however, can be performed by a simple unit operating at the moderate drum clock rate. Establishing service units which are unchanged in the various machine modes and switching the interconnections of the central control circuits to change programs provides a relatively inexpensive way of conducting the various modes of the data handling processes.

The Logical Design of a 1-Microsecond Parallel Adder Using 1-Megacycle Circuitry

A. WEINBERGER J. L. SMITH

Synopsis: The logical design of a parallel adder is developed which is capable of adding two 53-bit numbers in 1 microsecond. The design makes use of basically the same 1-megacycle circuitry which has been used successfully in the National Bureau of Standards' SEAC and DYSEAC computers. An analysis of the functional relationships of the carry digits to the augend and addend digits shows that it is feasible to form many carries simultaneously at the expense of relatively few components. The Boolean expressions for many successive carry digits can be expanded as explicit functions of some one lower-order carry, and of the relevant augend and addend digits. These somewhat complicated expressions are simplified by making substitutions for the common terms and factors they contain. These common terms and factors, called auxiliary carry functions, are implemented separately. All functional forms fit within the wide limits of gating complexity allowed by the type of circuitry to be used.

A. Weinberger and J. L. Smith are with the National Bureau of Standards, Washington, D. C.

THE development at the National Bureau of Standards of the diode capacitor memory,[1,2] which is capable of being read or written into at the rate of one word per microscecond, has made it worth while to build devices capable of processing information at comparable rates. Since the basic micro-operation common to most arithmetic processes is the adding together of two numbers, it seemed reasonable to design an adder having a cycle time no greater than 1 microsecond.

The major timing bind in an adder is in the production of carries, and in this paper the problem is attacked from the standpoint of logical organization. Although

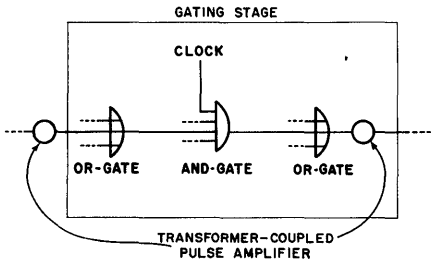


Fig. 1. One stage of SEAC-type circuitry

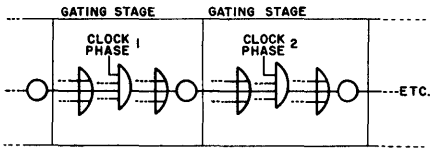


Fig. 2. Gating stages clocked with different clock phases

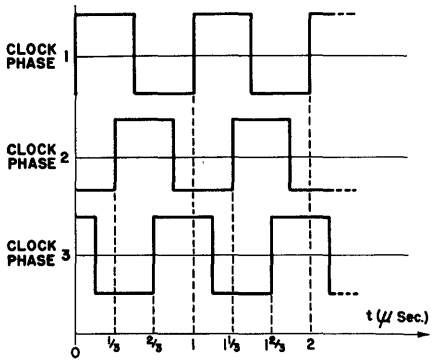


Fig. 3. Time relationships among SEAC clock phases

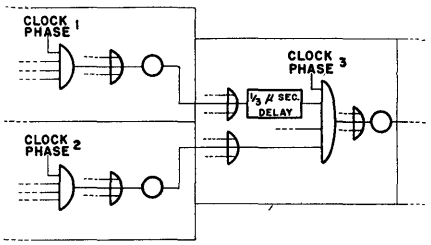


Fig. 4. Synchronizing by means of electrical delay line

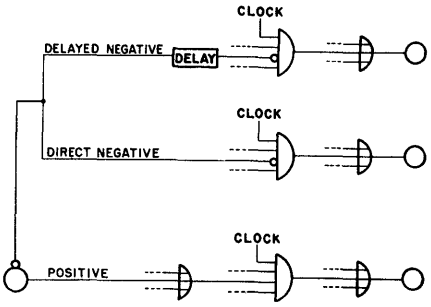


Fig. 5. Use of negative signals for inhibiting

elsewhere work is being done on this subject using newer and faster basic circuit elements, the analyses to be described show that it is both feasible and economical to achieve 1-microsecond addition times for 53-bit words using the 1-megacycle circuitry which has been successfully utilized in SEAC[3] and DYSEAC.[4,5]

The increased complexity of the logic of this adder necessitated the extensive used of Boolean algebra in arriving at the design itself. Because the procedure used in developing the final design is an interesting example of the practical application of Boolean algebra, the actual logic of the design process is described in considerable detail.

Before discussing the adder, a brief description of the logical capabilities of the SEAC circuitry[6] is in order. As shown in Fig. 1, the basic electronic unit consists essentially of three levels of diode gates in an OR-AND-OR logical array followed by a transformer-coupled pulse amplifier. The rate at which successive pulses pass through such a stage is determined by the clock frequency which is, in this case, 1 megacycle per second. The transit time of a pulse through a stage, however, is much less than 1 microsecond. For this reason, the clock pulses are made available in several phases. The way in which different stages may be controlled by clock pulses of different phases is illustrated in Fig. 2. In SEAC, for example, 1-megacycle clock pulses are available in 3 phases, 1/3 microsecond apart. In DYSEAC, 4-phase clock pulses are used, while in the adder to be described a 5-phase clock is used. Fig. 3 shows graphically these timing relationships for SEAC. Signals resulting from different stages clocked at different times must be synchronized by means of electrical delay lines before they are gated in a common stage as shown in Fig. 4. Both positive and negative signals are available from a stage, the negative signals being used for inhibiting. (See Fig 5.)

The maximum gating complexity used for a stage in the adder to be described is essentially the same as that employed in the packaged building blocks used in constructing DYSEAC, and, therefore, in the OR-AND-OR gating configuration of a stage, up to four AND-gates, and up to six inputs to an AND-gate, are permissible.

Boolean notation of the sort described by Richards[7] will be used hereafter to describe the gating configurations. In Fig. 6 is shown a typical gating stage and the corresponding Boolean expression for the output in terms of the inputs. There are three terms in the expression, each one corresponding to an AND-gate; the first term, $(A + B)\bar{C}\bar{D}EF$, corresponds to the top AND-gate, the second term, $(G + H)I$ corresponds to the middle AND-gate, and the last term, $J(K + L + M)\bar{N}$, corresponds to the bottom AND-gate. The factors of a term represent the inputs to the corresponding AND-gate. For example, the five factors of the first term $(A + B)$, $\bar{C}$, $\bar{D}$, E, and F, correspond to five inputs to the top AND-gate. Whenever a factor consists of more than one term, it is represented by an OR-gate. For example, the factor $(A + B)$ of the first term corresponds to the 2-input OR-gate of the top AND-gate. A factor could also be a negative or inhibit signal, and in this case it is denoted by a bar on top; e.g., $\bar{C}$ and $\bar{D}$ are two factors of the first term corresponding to the two negative signals which may inhibit the top AND-gate. For the sake of simplicity in the discussion of the Boolean expressions which follow, no distinction is made between delayed and undelayed signals.

Sequential Carry Propagation

Let

$$A = A_n \times 2^{n-1} + A_{n-1} \times 2^{n-2} + \ldots + A_2 \times 2^1 + A_1 \times 2^0 = \text{augend}$$

$$B = B_n \times 2^{n-1} + B_{n-1} \times 2^{n-2} + \ldots + B_2 \times 2^1 + B_1 \times 2^0 = \text{addend}$$

$$S = S_n \times 2^{n-1} + S_{n-1} \times 2^{n-2} + \ldots + S_2 \times 2^1 + S_1 \times 2^0 = \text{sum}$$

C = carry digit

The well-known rules for forming the sum and carry digits are presented in the form of a function table (Table I).

Table I. Function Table for Binary Addition

Augend	A_k	0	0	0	0	1	1	1	1
Addend	B_k	0	0	1	1	0	0	1	1
Previous Carry	C_{k-1}	0	1	0	1	0	1	0	1
Sum	S_k	0	1	1	0	1	0	0	1
Carry	C_k	0	0	0	1	0	1	1	1

From these, the binary sum and carry can be expressed in Boolean notation as follows:

$$S_k = \bar{A}_k\bar{B}_kC_{k-1} + \bar{A}_kB_k\bar{C}_{k-1} + A_k\bar{B}_k\bar{C}_{k-1} + A_kB_kC_{k-1} \quad (1)$$

$$\begin{aligned} C_k &= \bar{A}_kB_kC_{k-1} + A_k\bar{B}_kC_{k-1} + A_kB_k\bar{C}_{k-1} + A_kB_kC_{k-1} \\ &= A_kB_k + A_kC_{k-1} + B_kC_{k-1} \\ &= (A_k + B_k)(A_k + C_{k-1})(B_k + C_{k-1}) \\ &= A_kB_k + (A_k + B_k)C_{k-1} \end{aligned} \quad (2)$$

Equations 2 show how the carry function, C_k, can be reduced from four terms of three factors each (corresponding to four

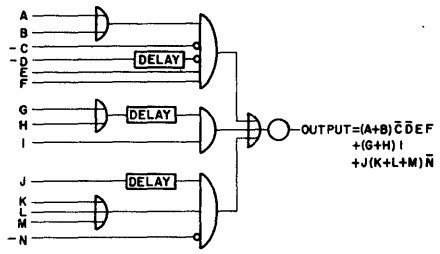


Fig. 6. Typical gating stage and corresponding Boolean expression

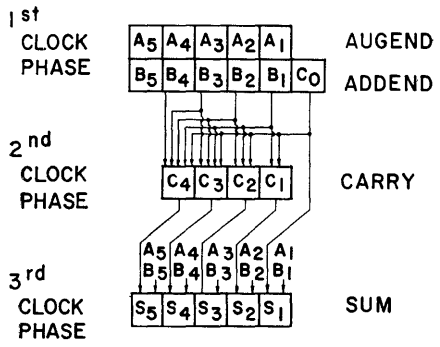


Fig. 7. 5-bit parallel binary adder

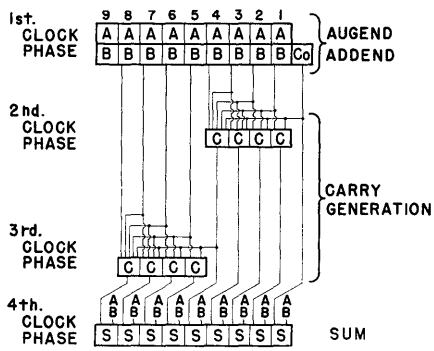


Fig. 8. 9-bit parallel binary adder

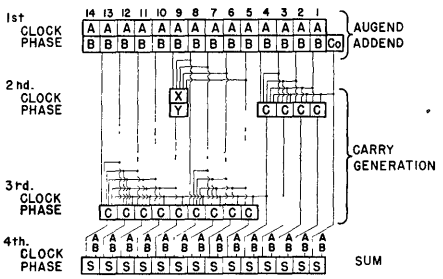


Fig. 9. 14-bit parallel binary adder

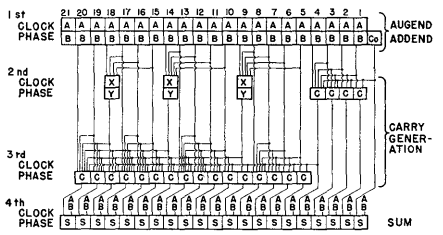


Fig. 10. 21-bit parallel binary adder

AND-gates with three inputs each), as shown in the top line of equations 2, to three alternative forms each involving fewer terms and factors.

Since the expression for S_k in equation 1 can be implemented in one gating stage, any sum digit can be made available during the clock phase immediately following the formation of the preceding carry, C_{k-1}. The speed with which successive sum digits are formed is therefore determined by the speed with which successive carries are generated.

Since each carry is explicitly dependent upon the immediately preceding one in equations 2, successive carries can be generated one clock phase apart, i.e., at the rate determined by the time interval between stages. Using any one of the equations in equations 2, if C_1 is formed during the first clock phase, C_2 can be formed during the second clock phase, C_3 during the third clock phase, etc.

Simultaneous Carry Generation

It will now be shown how C_k can be expanded so as to be independent of the previous carry. For the moment the dependence of a carry upon the appropriate augend and addend digits will be neglected. Then, from equations 2, C_k is a function of C_{k-1}, C_{k-1} is a function of C_{k-2}, etc., so that C_k can be expressed as a function of C_{k-2}. In fact, C_k can be further expanded in this fashion until it is a function of C_{k-3}, then of C_{k-4}, etc. The limit to this expansion is set by the limit to the gating complexity permitted with the circuitry.

The last of equations 2 is particularly important because of the ease with which it can be expanded so that C_k will be independent of a number of previous carries. Equations 3 show how each additional order of expansion adds only one term to the expression for C_k and only one factor to the largest of the terms.

$$\begin{aligned}C_k &= A_kB_k+(A_k+B_k)C_{k-1}\\ &= A_kB_k+(A_k+B_k)A_{k-1}B_{k-1}+\\ &\quad (A_k+B_k)(A_{k-1}+B_{k-1})\times C_{k-2}\\ &= A_kB_k+(A_k+B_k)A_{k-1}B_{k-1}+\\ &\quad (A_k+B_k)(A_{k-1}+B_{k-1})A_{k-2}B_{k-2}+\\ &\quad (A_k+B_k)(A_{k-1}+B_{k-1})\\ &\quad (A_{k-2}+B_{k-2})C_{k-3}\\ &= \text{etc.} \qquad (3)\end{aligned}$$

Applying the foregoing method for expanding the carry function, how many successive carries can be generated simultaneously is next determined. Beginning with the least significant carry, C_1, four successive carries are shown in equations 4 to be functions of C_0 and to consist of no more terms and factors than can be implemented by the physical gating structures outlined previously.

$$\begin{aligned}C_1 &= A_1B_1+(A_1+B_1)C_0\\ C_2 &= A_2B_2+(A_2+B_2)A_1B_1+\\ &\quad (A_2+B_2)(A_1+B_1)C_0\\ C_3 &= A_3B_3+(A_3+B_3)A_2B_2+\\ &\quad (A_3+B_3)(A_2+B_2)A_1B_1+\\ &\quad (A_3+B_3)(A_2+B_2)(A_1+B_1)C_0\\ C_4 &= A_4B_4+(A_4+B_4)A_3B_3+\\ &\quad (A_4+B_4)(A_3+B_3)A_2B_2+\\ &\quad (A_4+B_4)(A_3+B_3)(A_2+B_2)A_1B_1+\\ &\quad (A_4+B_4)(A_3+B_3)(A_2+B_2)\times\\ &\quad (A_1+B_1)C_0\\ &= A_4B_4+(A_4+B_4)A_3B_3+\\ &\quad (A_4+B_4)(A_3+B_3)A_2B_2+\\ &\quad (A_4+B_4)(A_3+B_3)(A_2+B_2)\times\\ &\quad (A_1+B_1)(A_1+C_0)(B_1+C_0) \qquad (4)\end{aligned}$$

It can be seen that the first expression for C_4 in equations 4 consists of five terms and therefore cannot be formed in one gating stage in this manner. However, the last two terms can be combined into one term by means of equation 5,

$$A_1B_1+(A_1+B_1)C_0=(A_1+B_1)(A_1+C_0)\times (B_1+C_0) \qquad (5)$$

thereby permitting C_4 to be formed in one gating stage as shown in the second expression for C_4 in equations 4.

(C_0 is equivalent to a digit which, together with the least significant augend and addend digits, forms the least significant sum digit, thus:

$$S_1=\bar{A}_1\bar{B}_1C_0+\bar{A}_1B_1\bar{C}_0+A_1\bar{B}_1\bar{C}_0+A_1B_1C_0. \qquad (6)$$

C_0 is used during addition cycles requiring the adding of 1 to the sum, such as the adding of negative numbers in "ones-complement" form.)

Fig. 7 shows in block-diagram form the five least significant digits of a parallel adder utilizing the above principle. Note that the carries C_1 through C_4 are obtained during the clock phase following that of the augend and addend digits. Also, the least significant sum digit, S_1, can actually be obtained during the second clock phase according to equation 6 because it is a function of A_1, B_1, and C_0, all three of which are available during the first clock phase. However, it is desirable in this parallel adder to obtain all of the sum digits at the same time. Consequently, delaying S_1 to occur at the same clock phase as the other sum digits is arranged.

The succeeding four carries, C_5 through C_8, can be formed one clock phase later in a similar fashion using C_4 as the last previous carry, as shown in equations 7. In the case of C_8, the reduction from five to

four terms is made by combining the first two terms instead of the last two as in the expression for C_4 in equations 4, in order to stay within the circuitry limitations for OR-gating delayed signals. (All signals passing through the same OR-gate must originate at the same time.)

$$C_5=A_5B_5+(A_5+B_5)C_4$$
$$C_6=A_6B_6+(A_6+B_6)A_5B_5+(A_6+B_6)(A_5+B_5)C_4$$
$$C_7=A_7B_7+(A_7+B_7)A_6B_6+(A_7+B_7)(A_6+B_6)A_5B_5+(A_7+B_7)(A_6+B_6)(A_5+B_5)C_4$$
$$C_8=A_8B_8+(A_8+B_8)A_7B_7+(A_8+B_8)(A_7+B_7)A_6B_6+(A_8+B_8)(A_7+B_7)(A_6+B_6)A_5B_5+(A_8+B_8)(A_7+B_7)(A_6+B_6)\times(A_5+B_5)C_4$$
$$=(A_8+B_8)(A_8+A_7)(A_8+B_7)(B_8+A_7)\times(B_8+B_7)+(A_8+B_8)(A_7+B_7)A_6B_6+(A_8+B_8)(A_7+B_7)(A_6+B_6)A_5B_5+(A_8+B_8)(A_7+B_7)(A_6+B_6)\times(A_5+B_5)C_4 \quad (7)$$

Fig. 8 extends the previous block diagram to include a parallel adder accommodating nine binary digits. Again, the formation of the sum digits S_1 through S_5 is delayed to coincide with the rest of the sum digits.

Use of Auxiliary Carry Functions

Of signal importance is the use made of the clock phase available between the input digits and the carries C_5 through C_8. This extra time can be used to form certain auxiliary carry functions which enable us to generate additional carries during the third clock phase simultaneously with C_5 through C_8. More specifically, C_9, C_{10}, etc., can be formed during the third clock phase as functions of C_4 if some of the terms in the expanded relations for C_9, C_{10}, etc., are combined as auxiliary carry functions in separate stages during the intervening clock phase.

For example, the expression for C_9 is expanded in equations 8 to be a function of C_4.

$$C_9=\boxed{A_9B_9}+\boxed{(A_9+B_9)A_8B_8}+\boxed{(A_9+B_9)(A_8+B_8)A_7B_7}+\boxed{(A_9+B_9)(A_8+B_8)(A_7+B_7)A_6B_6}+\boxed{(A_9+B_9)(A_8+B_8)(A_7+B_7)(A_6+B_6)A_5B_5}+\boxed{(A_9+B_9)(A_8+B_8)(A_7+B_7)(A_6+B_6)(A_5+B_5)}\boxed{C_4}$$
$$C_9=\boxed{X_9}+\boxed{Y_9}\boxed{C_4} \quad (8)$$

The outlines drawn around the various parts of equations 8 serve merely to correlate the corresponding parts. The five terms enclosed within the triangle and represented by X_9 can be reduced to four terms by combining the first two. The reduced 4-term expression can then be implemented in one gating stage during the second clock phase. Furthermore, the single factor enclosed within the rectangle, called Y_9, can also be implemented during the second clock phase in one gating stage. By means of these two auxiliary carry functions, C_9 can be formed quite easily in one gating stage during the third clock phase according to the second equation in equations 8.

Similarly, C_{10} through C_{13} can be formed during the third clock phase by utilizing these auxiliary carry functions. The most complicated of these expressions, C_{13}, is illustrated in equations 9 where further combinations must be made to bring the number of terms down to four.

$$C_{13}=A_{13}B_{13}+(A_{13}+B_{13})A_{12}B_{12}+(A_{13}+B_{13})(A_{12}+B_{12})A_{11}B_{11}+(A_{13}+B_{13})(A_{12}+B_{12})(A_{11}+B_{11})\times A_{10}B_{10}+(A_{13}+B_{13})(A_{12}+B_{12})\times(A_{11}+B_{11})(A_{10}+B_{10})X_9+(A_{13}+B_{13})(A_{12}+B_{12})(A_{11}+B_{11})\times(A_{10}+B_{10})Y_9C_4$$
$$=(A_{13}+B_{13})(A_{13}+A_{12})(A_{13}+B_{12})\times(B_{13}+A_{12})(B_{13}+B_{12})+(A_{13}+B_{13})(A_{12}+B_{12})A_{11}B_{11}+(A_{13}+B_{13})(A_{12}+B_{12})(A_{11}+B_{11})\times A_{10}B_{10}+(A_{13}+B_{13})(A_{12}+B_{12})\times(A_{11}+B_{11})(A_{10}+B_{10})(X_9+Y_9)\times(X_9+C_4) \quad (9)$$

Fig. 9 illustrates a parallel adder handling 14 binary digits using one pair of auxiliary carry functions.

By means of additional auxiliary carry functions it is possible to extend still further the sequence of carries to be formed in one clock phase. For example, as shown in equations 10, C_{14} can also be expressed as a function of C_4 with the aid of auxiliary carry functions X_{14} and Y_{14}.

$$C_{14}=\boxed{A_{14}B_{14}}+\boxed{(A_{14}+B_{14})A_{13}B_{13}}+\boxed{(A_{14}+B_{14})(A_{13}+B_{13})A_{12}B_{12}}+\boxed{(A_{14}+B_{14})(A_{13}+B_{13})(A_{12}+B_{12})A_{11}B_{11}}+\boxed{(A_{14}+B_{14})(A_{13}+B_{13})(A_{12}+B_{12})(A_{11}+B_{11})A_{10}B_{10}}+\boxed{(A_{14}+B_{14})(A_{13}+B_{13})(A_{12}+B_{12})(A_{11}+B_{11})(A_{10}+B_{10})}X_9+\boxed{(A_{14}+B_{14})(A_{13}+B_{13})(A_{12}+B_{12})(A_{11}+B_{11})(A_{10}+B_{10})}Y_9C_4$$
$$C_{14}=\boxed{X_{14}}+\boxed{Y_{14}}\boxed{X_9}+\boxed{Y_{14}}Y_9C_4 \quad (10)$$

These again represent the terms within the triangle and rectangles, respectively. C_{15}, C_{16}, and C_{17} can also be implemented in single stages as functions of C_4 using the two pairs of auxiliary carry functions. C_{18}, C_{19}, and C_{20} require still another pair of auxiliary carry functions in order for them to be generated during the third clock phase as functions of C_4.

If the number of auxiliary functions were of no concern, a total of 25 carries could be generated simultaneously as functions of C_4 during the third clock phase. However, by limiting the number of simultaneous carries to 16, only three pairs of auxiliary carry functions are required. Fig. 10 illustrates a parallel adder handling 21 binary digits utilizing this scheme.

Two Levels of Auxiliary Carry Functions

To extend the parallel adder to accommodate 53 binary digits, only one additional clock phase is necessary. During the fourth clock phase the carries C_{21} through C_{52} can all be generated as functions of C_{20}. The entire parallel array of sum digits, S_1 through S_{53}, is then formed during the fifth clock phase.

The ability to generate all of the carries C_{21} through C_{52} during the fourth clock phase stems from the fact that two clock phases are available between these carries and the input digits. This permits the formation of two levels of auxiliary carry functions. The first level consists of sets of X and Y values which are functions of only the appropriate augend and addend digits, as previously. The second level of auxiliary carry functions consists of sets of stages labeled Z and W which are functions of certain first-level functions.

Fig. 11 illustrates in block-diagram form the complete 53-bit adder which makes use of second-level auxiliary carry stages. As in the case of the preceding carries, C_{21} through C_{32} are generated as functions of the appropriate augend and addend digits, some of the first-level auxiliary carry stages, and C_{20}. For example, the most complicated of these, C_{32}, is shown in equations 11 to be reducible to four terms.

$$C_{32}=A_{32}B_{32}+(A_{32}+B_{32})A_{31}B_{31}+(A_{32}+B_{32})(A_{31}+B_{31})A_{30}B_{30}+(A_{32}+B_{32})(A_{31}+B_{31})(A_{30}+B_{30})X_{29}+(A_{32}+B_{32})(A_{31}+B_{31})(A_{30}+B_{30})\times Y_{29}X_{25}+(A_{32}+B_{32})(A_{31}+B_{31})\times(A_{30}+B_{30})Y_{29}Y_{25}C_{20}$$

$$=(A_{32}+B_{32})(A_{32}+A_{31})(A_{32}+B_{31})\times (B_{32}+A_{31})(B_{32}+B_{31})+(A_{32}+B_{32})\times (A_{31}+B_{31})A_{30}B_{30}+(A_{32}+B_{32})\times (A_{31}+B_{31})(A_{30}+B_{30})(X_{29}+Y_{29})\times (X_{29}+X_{25})+(A_{32}+B_{32})(A_{31}+B_{31})\times (A_{30}+B_{30})Y_{29}Y_{25}C_{20} \quad (11)$$

The next higher order carry, C_{33}, requires a third pair of auxiliary carry functions, X_{33} and Y_{33}, as shown in equations 12. However, at this point it becames economical to form a pair of second-level auxiliary carry functions, Z_{33}, consisting of terms within the solid-line triangle, and W_{33}, consisting of the single term within the solid-line rectangle. C_{33} can then be easily generated by means of Z_{33} and W_{33} as shown in equations 12. The elements enclosed within the broken-line triangles and rectangles correspond to the first-level auxiliary carry functions, X_{33} and Y_{33}.

The subsequent carries, C_{34}, C_{35}, etc., are similarly generated by means of these and, when necessary, other second-level auxiliary carry functions. For example, for the carries up to C_{37}, one pair of second-level functions is sufficient, as seen from equations 13. C_{38} requires the formation of another pair of first and second-level functions, as shown in equations 14.

Table II. Auxiliary Carry Functions

$X_9=F_9+R_9R_8D_7+R_9R_8R_7D_6+R_9R_8R_7R_6D_5$
$X_{14}=F_{14}+R_{14}R_{13}D_{12}+R_{14}R_{13}R_{12}D_{11}+R_{14}R_{13}R_{12}R_{11}D_{10}$
$X_{18}=D_{18}+R_{18}D_{17}+R_{18}R_{17}D_{16}+R_{18}R_{17}R_{16}D_{15}$
$X_{25}=F_{25}+R_{25}R_{24}D_{23}+R_{25}R_{24}R_{23}D_{22}+R_{25}R_{24}R_{23}R_{22}D_{21}$
$X_{29}=D_{29}+R_{29}D_{28}+R_{29}R_{28}D_{27}+R_{29}R_{28}R_{27}D_{26}$
$X_{33}=D_{33}+R_{33}D_{32}+R_{33}R_{32}D_{21}+R_{33}R_{32}R_{31}D_{30}$
$X_{38}=F_{38}+R_{38}R_{37}D_{36}+R_{38}R_{37}R_{36}D_{35}+R_{38}R_{37}R_{36}R_{35}D_{34}$
$X_{43}=F_{43}+R_{43}R_{42}D_{41}+R_{43}R_{42}R_{41}D_{40}+R_{43}R_{42}R_{41}R_{40}D_{39}$
$X_{48}=F_{48}+R_{48}R_{47}D_{46}+R_{48}R_{47}R_{46}D_{45}+R_{48}R_{47}R_{46}R_{45}D_{44}$
$Z_{33}=X_{33}+Y_{33}X_{29}+Y_{33}Y_{29}X_{25}$
$Z_{38}=X_{38}+Y_{38}X_{33}+Y_{38}Y_{33}X_{29}+Y_{38}Y_{33}Y_{29}X_{25}$
$Z_{43}=X_{43}+Y_{43}X_{38}+Y_{43}Y_{38}X_{33}+Y_{43}Y_{38}Y_{33}(X_{29}+Y_{29})(X_{29}+X_{25})$
$Z_{48}=X_{48}+Y_{48}X_{43}+Y_{48}Y_{43}(X_{38}+Y_{38})(X_{38}+X_{33})+Y_{48}Y_{43}Y_{38}Y_{33}(X_{29}+Y_{29})(X_{29}+X_{25})$

$Y_9=R_9R_8R_7R_6R_5$
$Y_{14}=R_{14}R_{13}R_{12}R_{11}R_{10}$
$Y_{18}=R_{18}R_{17}R_{16}R_{15}$
$Y_{25}=R_{25}R_{24}R_{23}R_{22}R_{21}$
$Y_{29}=R_{29}R_{28}R_{27}R_{26}$
$Y_{33}=R_{33}R_{32}R_{31}R_{30}$
$Y_{38}=R_{38}R_{37}R_{36}R_{35}R_{34}$
$Y_{43}=R_{43}R_{42}R_{41}R_{40}R_{39}$
$Y_{48}=R_{48}R_{47}R_{46}R_{45}R_{44}$
$W_{33}=Y_{33}Y_{29}Y_{25}$
$W_{38}=Y_{38}Y_{33}Y_{29}Y_{25}$
$W_{43}=Y_{43}Y_{38}Y_{33}Y_{29}Y_{25}$
$W_{48}=Y_{48}Y_{43}Y_{38}Y_{33}Y_{29}Y_{25}$

F_k represents $(A_k+B_k)(A_k+A_{k-1})(A_k+B_{k-1})(B_k+A_{k-1})(B_k+B_{k-1})$
D_k represents A_kB_k
R_k represents (A_k+B_k)

$$C_{37}=A_{37}B_{37}+(A_{37}+B_{37})A_{36}B_{36}+(A_{37}+B_{37})(A_{36}+B_{36})A_{35}B_{35}+(A_{37}+B_{37})(A_{36}+B_{36})(A_{35}+B_{35})\times A_{34}B_{34}+(A_{37}+B_{37})(A_{36}+B_{36})\times (A_{35}+B_{35})(A_{34}+B_{34})Z_{33}+(A_{37}+B_{37})(A_{36}+B_{36})(A_{35}+B_{35})\times (A_{34}+B_{34})W_{33}C_{20}$$
$$=(A_{37}+B_{37})(A_{37}+A_{36})(A_{37}+B_{36})\times (B_{37}+A_{36})(B_{37}+B_{36})+(A_{37}+B_{37})\times (A_{36}+B_{36})A_{35}B_{35}+(A_{37}+B_{37})\times (A_{36}+B_{36})(A_{35}+B_{35})A_{34}B_{34}+(A_{37}+B_{37})(A_{36}+B_{36})(A_{35}+B_{35})\times (A_{34}+B_{34})(Z_{33}+W_{33})(Z_{33}+C_{20}) \quad (13)$$

$$C_{33}=A_{33}B_{33}+(A_{33}+B_{33})A_{32}B_{32}+(A_{33}+B_{33})(A_{32}+B_{32})A_{31}B_{31}+(A_{33}+B_{33})(A_{32}+B_{32})(A_{31}+B_{31})A_{30}B_{30}+(A_{33}+B_{33})(A_{32}+B_{32})(A_{31}+B_{31})(A_{30}+B_{30})X_{29}+(A_{33}+B_{33})(A_{32}+B_{32})(A_{31}+B_{31})(A_{30}+B_{30})Y_{29}X_{25}+(A_{33}+B_{33})(A_{32}+B_{32})(A_{31}+B_{31})(A_{30}+B_{30})Y_{29}Y_{25}C_{20} \quad (12)$$

$$C_{33}=X_{33}+Y_{33}X_{29}+Y_{33}Y_{29}X_{25}+Y_{33}Y_{29}Y_{25}C_{20}$$

$$C_{33}=Z_{33}+W_{33}C_{20}$$

The last digit position where auxiliary carry functions are introduced is at 48. The carry at this position, C_{48}, is shown in equations 15 to be a simple function of the last pair of second-level auxiliary carry functions.

The number of stages required to implement the adder in this fashion can easily be determined from Fig. 11. Each box in the diagram represents one gating stage. In addition to the four registers of gating stages for the augend digits, addend digits, carry digits, and sum digits, only 26 gating stages, equivalent to one-half of a register, are required to create the auxiliary carry functions.

The expressions for the auxiliary carry functions of this particular adder as well as its final carry functions are shown in Tables II and III, respectively.

The top line of Table IV gives some statistics on the number of components required for the adder represented in Fig. 11. Two other slightly different versions have been worked out in which fewer gates need to be driven by the most heavily loaded tube. As Table IV shows, these

Table III. Carry Functions

$C_1=D_1+R_1C_0$
$C_2=D_2+R_2D_1+R_2R_1C_0$
$C_3=D_3+R_3D_2+R_3R_2D_1+R_3R_2R_1C_0$
$C_4=D_4+R_4D_3+R_4R_3D_2+R_4R_3R_2R_1(A_1+C_0)(B_1+C_0)$
$C_5=D_5+R_5C_4$
$C_6=D_6+R_6D_5+R_6R_5C_4$
$C_7=D_7+R_7D_6+R_7R_6D_5+R_7R_6R_5C_4$
$C_8=F_8+R_8R_7D_6+R_8R_7R_6D_5+R_8R_7R_6R_5C_4$
$C_9=X_9+Y_9C_4$
$C_{10}=D_{10}+R_{10}(X_9+Y_9)(X_9+C_4)$
$C_{11}=D_{11}+R_{11}D_{10}+R_{11}R_{10}(X_9+Y_9)(X_9+C_4)$
$C_{12}=D_{12}+R_{12}D_{11}+R_{12}R_{11}D_{10}+R_{12}R_{11}R_{10}(X_9+Y_9)(X_9+C_4)$
$C_{13}=F_{13}+R_{13}R_{12}D_{11}+R_{13}R_{12}R_{11}D_{10}+R_{13}R_{12}R_{11}R_{10}(X_9+X_9)(X_9+C_4)$
$C_{14}=X_{14}+Y_{14}X_9+Y_{14}Y_9C_4$
$C_{15}=D_{15}+R_{15}X_{14}+R_{15}Y_{14}(X_9+Y_9)(X_9+C_4)$
$C_{16}=D_{16}+R_{16}D_{15}+R_{16}R_{15}X_{14}+R_{16}R_{15}Y_{14}(X_9+Y_9)(X_9+C_4)$
$C_{17}=F_{17}+R_{17}R_{16}D_{15}+R_{17}R_{16}R_{15}X_{14}+R_{17}R_{16}R_{15}Y_{14}(X_9+Y_9)(X_9+C_4)$
$C_{18}=X_{18}+Y_{18}X_{14}+Y_{18}Y_{14}X_9+Y_{18}Y_{14}Y_9C_4$
$C_{19}=D_{19}+R_{19}(X_{18}+Y_{18})(X_{18}+X_{14})+R_{19}Y_{18}Y_{14}(X_9+Y_9)(X_9+C_4)$
$C_{20}=D_{20}+R_{20}D_{19}+R_{20}R_{19}(X_{18}+Y_{18})(X_{18}+X_{14})+R_{20}R_{19}Y_{18}Y_{14}(X_9+Y_9)(X_9+C_4)$
$C_{21}=D_{21}+R_{21}C_{20}$
$C_{22}=D_{22}+R_{22}D_{21}+R_{22}R_{21}C_{20}$
$C_{23}=D_{23}+R_{23}D_{22}+R_{23}R_{22}D_{21}+R_{23}R_{22}R_{21}C_{20}$
$C_{24}=F_{24}+R_{24}R_{23}D_{22}+R_{24}R_{23}R_{22}D_{21}+R_{24}R_{23}R_{22}R_{21}C_{20}$
$C_{25}=X_{25}+Y_{25}C_{20}$
$C_{26}=D_{26}+R_{26}X_{25}+R_{26}Y_{25}C_{20}$
$C_{27}=D_{27}+R_{27}D_{26}+R_{27}R_{26}X_{25}+R_{27}R_{26}Y_{25}C_{20}$
$C_{28}=F_{28}+R_{28}R_{27}D_{26}+R_{28}R_{27}R_{26}X_{25}+R_{28}R_{27}R_{26}Y_{25}C_{20}$
$C_{29}=X_{29}+Y_{29}X_{25}+Y_{29}Y_{25}C_{20}$
$C_{30}=D_{30}+R_{30}(X_{29}+Y_{29})(X_{29}+X_{25})+R_{30}Y_{29}Y_{25}C_{20}$
$C_{31}=D_{31}+R_{31}D_{30}+R_{31}R_{30}(X_{29}+Y_{29})(X_{29}+X_{25})+R_{31}R_{30}Y_{29}Y_{25}C_{20}$
$C_{32}=F_{32}+R_{32}R_{31}D_{30}+R_{32}R_{31}R_{30}(X_{29}+Y_{29})(X_{29}+X_{25})+R_{32}R_{31}R_{30}Y_{29}Y_{25}C_{20}$
$C_{33}=Z_{33}+W_{33}C_{20}$
$C_{34}=D_{34}+R_{34}(Z_{33}+W_{33})(Z_{33}+C_{20})$
$C_{35}=D_{35}+R_{35}D_{34}+R_{35}R_{34}(Z_{33}+W_{33})(Z_{33}+C_{20})$
$C_{36}=D_{36}+R_{36}D_{35}+R_{36}R_{35}D_{34}+R_{36}R_{35}R_{34}(Z_{33}+W_{33})(Z_{33}+C_{20})$
$C_{37}=F_{37}+R_{37}R_{36}D_{35}+R_{37}R_{36}R_{35}D_{34}+R_{37}R_{36}R_{35}R_{34}(Z_{33}+W_{33})(Z_{33}+C_{20})$
$C_{38}=Z_{38}+W_{38}C_{20}$
$C_{39}=D_{39}+R_{39}(Z_{38}+W_{38})(Z_{38}+C_{20})$
$C_{40}=D_{40}+R_{40}D_{39}+R_{40}R_{39}(Z_{38}+W_{38})(Z_{38}+C_{20})$
$C_{41}=D_{41}+R_{41}D_{40}+R_{41}R_{40}D_{39}+R_{41}R_{40}R_{39}(Z_{38}+W_{38})(Z_{38}+C_{20})$
$C_{42}=F_{42}+R_{42}R_{41}D_{40}+R_{42}R_{41}R_{40}D_{39}+R_{42}R_{41}R_{40}R_{39}(Z_{38}+W_{38})(Z_{38}+X_{20})$
$C_{43}=Z_{43}+W_{43}C_{20}$
$C_{44}=D_{44}+R_{44}(Z_{43}+W_{43})(Z_{43}+C_{20})$
$C_{45}=D_{45}+R_{45}D_{44}+R_{45}R_{44}(Z_{43}+W_{43})(Z_{43}+C_{20})$
$C_{46}=D_{46}+R_{46}D_{45}+R_{46}R_{45}D_{44}+R_{46}R_{45}R_{44}(Z_{43}+W_{43})(Z_{43}+C_{20})$
$C_{47}=F_{47}+R_{47}R_{46}D_{45}+R_{47}R_{46}R_{45}D_{44}+R_{47}R_{46}R_{45}R_{44}(Z_{43}+W_{43})(Z_{43}+C_{20})$
$C_{48}=Z_{48}+W_{48}C_{20}$
$C_{49}=D_{49}+R_{49}(Z_{48}+W_{48})(Z_{48}+C_{20})$
$C_{50}=D_{50}+R_{50}D_{49}+R_{50}R_{49}(Z_{48}+W_{48})(Z_{48}+C_{20})$
$C_{51}=D_{51}+R_{51}D_{50}+R_{51}R_{50}D_{49}+R_{51}R_{50}R_{49}(Z_{48}+W_{48})(Z_{48}+C_{20})$
$C_{52}=F_{52}+R_{52}R_{51}D_{50}+R_{52}R_{51}R_{50}D_{49}+R_{52}R_{51}R_{50}R_{49}(Z_{48}+W_{48})(Z_{48}+C_{20})$

F_k represents $(A_k+B_k)(A_k+A_{k-1})(A_k+B_{k-1})(B_k+A_{k-1})(B_k+B_{k-1})$
D_k represents A_kB_k
R_k represents (A_k+B_k)

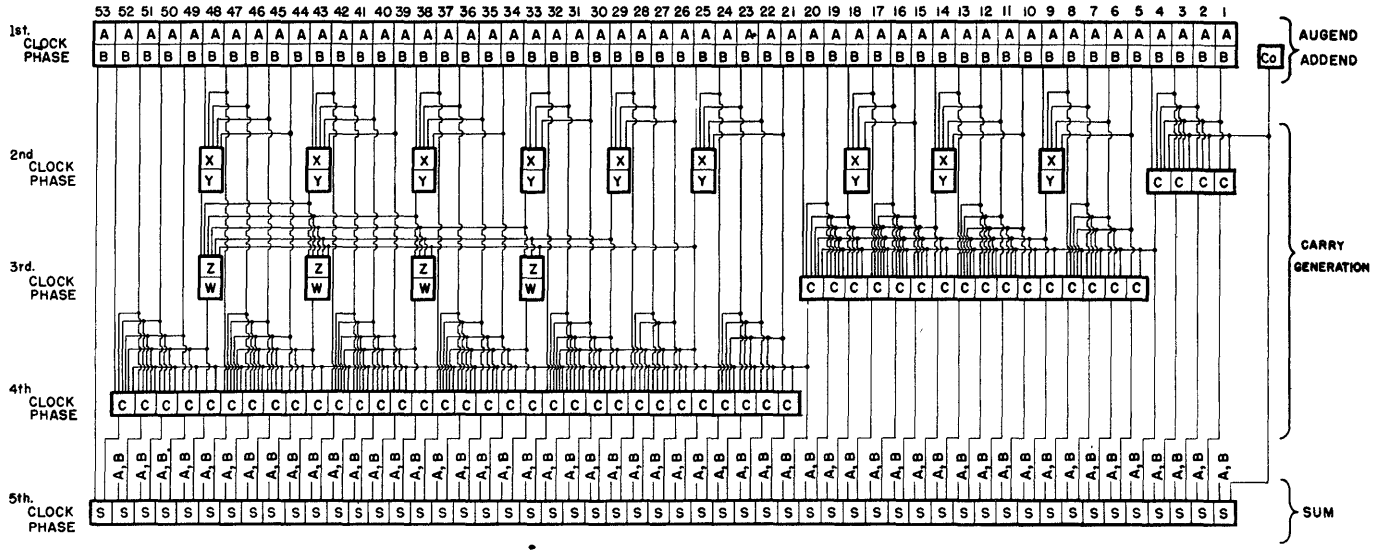


Fig. 11. 53-bit parallel binary adder

Table IV. Component Requirements for a 53-bit Parallel Binary Adder

Max. Load*	No. of Stages	Tubes	Delay Lines (in μsec.)	Germanium Diodes
25	238	238	300	10,000
19	253	253	250	10,000
14	285	285	150	10,000

* Unit of load = one gate-load.

versions also require different proportions of components.

If the adder is to be used for multiplication, division, and other operations requiring the recirculation of the sum digits back into one of the inputs, the clock must be available in five phases in order to complete the addition as well as the recirculation in 1 microsecond.

$$
\begin{aligned}
C_{38} = {} & A_{38}B_{38} \\
& + (A_{38}+B_{38})A_{37}B_{37} \\
& + (A_{38}+B_{38})(A_{37}+B_{37})A_{36}B_{36} \\
& + (A_{38}+B_{38})(A_{37}+B_{37})(A_{36}+B_{36})A_{35}B_{35} \\
& + (A_{38}+B_{38})(A_{37}+B_{37})(A_{36}+B_{36})(A_{35}+B_{35})A_{34}B_{34} \\
& + (A_{38}+B_{38})(A_{37}+B_{37})(A_{36}+B_{36})(A_{35}+B_{35})(A_{34}+B_{34})\,X_{33} \\
& + (A_{38}+B_{38})(A_{37}+B_{37})(A_{36}+B_{36})(A_{35}+B_{35})(A_{34}+B_{34})\,Y_{33}X_{29} \\
& + (A_{38}+B_{38})(A_{37}+B_{37})(A_{36}+B_{36})(A_{35}+B_{35})(A_{34}+B_{34})\,Y_{33}Y_{29}X_{25} \\
& + (A_{38}+B_{38})(A_{37}+B_{37})(A_{36}+B_{36})(A_{35}+B_{35})(A_{34}+B_{34})\,Y_{33}Y_{29}Y_{25}\,C_{20}
\end{aligned}
\qquad (14)
$$

$$
\begin{aligned}
C_{38} = {} & X_{38} \\
& + Y_{38}X_{33} \\
& + Y_{38}Y_{33}X_{29} \\
& + Y_{38}Y_{33}Y_{29}X_{25} \\
& + Y_{38}Y_{33}Y_{29}Y_{25}\,C_{20}
\end{aligned}
\qquad
\begin{aligned}
C_{38} = {} & Z_{38} \\
& + W_{38}\,C_{20}
\end{aligned}
$$

$$
\begin{aligned}
C_{48} = {} & X_{48} \\
& + Y_{48}X_{43} \\
& + Y_{48}Y_{43}X_{38} \\
& + Y_{48}Y_{43}Y_{38}X_{33} \\
& + Y_{48}Y_{43}Y_{38}Y_{33}X_{29} \\
& + Y_{48}Y_{43}Y_{38}Y_{33}Y_{29}X_{25} \\
& + Y_{48}Y_{43}Y_{38}Y_{33}Y_{29}Y_{25}\,C_{20}
\end{aligned}
\qquad (15)
$$

$$
\begin{aligned}
C_{48} = {} & X_{48} \\
& + Y_{48}X_{43} \\
& + Y_{48}Y_{43}(X_{38}+Y_{38})(X_{38}+X_{33}) \\
& + Y_{48}Y_{43}Y_{38}Y_{33}(X_{29}+Y_{29})(X_{29}+X_{25}) \\
& + Y_{48}Y_{43}Y_{38}Y_{33}Y_{29}Y_{25}\,C_{20}
\end{aligned}
\qquad
\begin{aligned}
C_{48} = {} & Z_{48} \\
& + W_{48}\,C_{20}
\end{aligned}
$$

References

1 An Experimental Rapid Access Memory Using Diodes and Capacitors, **A. W. Holt.** *Proceedings*, Association for Computing Machinery, New York, N. Y., December 1952.

2. High-Speed Memory Developments at the National Bureau of Standards, **R. J. Slutz, A. W. Holt, R. P. Witt, D. C. Friedman.** *Circular 551*, National Bureau of Standards, Washington, D. C., January 1955, pp. 93–108.

3. SEAC, **S. Greenwald, R. C. Haeuter, S. N. Alexander.** *Proceedings*, Institute of Radio Engineers, New York, N. Y., vol. 41, October 1953, pp. 1300–13. Also published in *Circular 551*, National Bureau of Standards, Washington, D. C., January 1955, pp. 5–26.

4. System Organization of the DYSEAC, **A. L. Leiner, S. N. Alexander.** *Transactions*, Institute of Radio Engineers Professional Group on Electronic Computers, New York, N. Y., vol. EC-3, no. 1, March 1954.

5. System Design of the SEAC and DYSEAC, **A. L. Leiner, W. A. Notz, J. L. Smith, A. Weinberger.** *Ibid.*, no. 2, June 1954.

6. Dynamic Circuit Techniques Used in SEAC and DYSEAC, **R. D. Elbourn, R. P. Witt.** *Proceedings*, Institute of Radio Engineers, New York, N. Y., vol. 41, October 1953, pp. 1380–87. Also published in *Transactions*, Institute of Radio Engineers Professional Group on Electronic Computers, New York, N. Y., vol. EC-2, no. 1, March 1953, and in *Circular 551*, National Bureau of Standards, Washington, D. C., January 1955, pp. 27–38.

7. Arithmetic Operations in Digital Computers (book), **R. K. Richards.** D. Van Nostrand Company, Inc., New York, N. Y., 1955, pp. 26–50.

The Transfluxor

J. A. RAJCHMAN A. W. LO

IN a recent paper[1] the authors have announced a novel device showing that completely new switching and storing functions can be performed by employing magnetic cores with two or more apertures,[2] (Magnetic circuits using multiaperature cores have also been reported elsewhere.[2,3]) instead of the conventional single-aperture cores, thereby creating a number of distinct flux paths via the legs of the core. The new device operates by the controlled transfer of flux from leg to leg in the magnetic circuit and was consequently named "transfluxor."

One of the most important properties of the transfluxor is its ability to store a level of control established by a single electric pulse. An energizing a-c drive will or will not produce an a-c output depending upon the nature of the last setting pulse to which the transfluxor was subjected. Furthermore, intermediate setting is possible for which an output of any desired level in a continuous range between almost zero and a maximum level will be produced according to the amplitude of a single setting pulse.

In the present paper, after a short review of the principle of the new device, its operation is illustrated in detail by the characteristics of a typical 2-apertured transfluxor and some of its applications. Several examples of logical functions attainable with multiapertured transfluxors are also included.

Principle of the 2-Aperture Transfluxor

Consider a core made of magnetic material such as a molded ceramic "ferrite" which has a nearly rectangular hysteresis loop and consequently a remanent induction B_r substantially equal to the saturated induction B_s. Let there be two circular apertures of unequal diameter which form three distinct legs, *1*, *2*, and *3*, in the magnetic circuit, as illustrated in Fig. 1. The areas of the cross sections of the legs *2* and *3* are equal and the cross section of leg *1* is equal to, or greater than, the sum of those of legs *2* and *3*.

The operation of the device previously explained[1] is reviewed here for convenience. Assume that at first an intense current pulse is sent through winding W_1 on leg *1* in a direction to produce a clockwise flux flow which saturates legs *2* and *3*. This is possible since the larger leg *1* provides the necessary return path. These legs will remain saturated after the termination of the pulse since remanent and saturated inductions are almost equal. Consider now the effect of an energizing alternating current in winding W_3 linking leg *3*, producing an alternating magnetomotive force along a path surrounding the smaller aperture, as shown by the shaded area in Fig. 1. When this magnetomotive force has a clockwise sense, it tends to produce an increase in flux in leg *3*, and a decrease in leg *2*. But no increase of flux is possible in leg *3* because it is saturated; consequently, there can be no flux flow at all, since magnetic flux flow is necessarily in close paths. Similarly, during the opposite phase of the alternating current, the magnetomotive force is in a counterclockwise sense and tends to produce an increase of flux in leg *2*, but in the case leg *2* is saturated. Consequently, flux flow is blocked as the result of the direction of saturation of either leg *2* or *3*. The transfluxor is in its blocked state and no voltage is induced in an output winding W_0 linking leg *3*.

Consider now the effect of a current pulse through winding W_1 in a direction producing a counterclockwise magnetomotive force. Let this pulse be intense enough to produce a magnetizing force in the closer leg *2* larger than the coercive force H_c, but not large enough to allow the magnetizing force in the more distant leg *3* to exceed the critical value. This pulse, called hereafter the "setting pulse", will cause the saturation of leg *2* to reverse and become directed upwards (Fig. 1), but will not affect leg *3* which will remain saturated downward. In this condition, the alternating magnetomotive force around the small aperture resulting from the alternating current in winding W_3 will produce a corresponding flux flow around the small aperture. The first counterclockwise phase of the alternating current will reverse the flux, the next clockwise phase will reverse it again, etc., indefinitely. This flow may be thought of as a back-and-forth transfer of flux between legs *2* and *3*. The alternating flux flow will induce a voltage in the output winding W_0. This is the unblocked or "maximum-set" state of the transfluxor.

It is seen that the transfluxor is blocked when the directions of remanent induction of the legs surrounding the smaller aperture are the same, and unblocked when they are opposite. In the blocked state the magnetic material around the small aperture provides essentially no coupling between the primary (W_3) and secondary (W_0) windings, while it provides a relatively large coupling between these two windings in the unblocked state. It is interesting to note that the information as to whether the transfluxor is blocked or unblocked can be thought of as being stored in terms of the flux through leg *1*, and that this stored flux does not change when output is produced by interchange of flux between legs *2* and *3*.

The transfluxor can also be set to any level in a continuous range in response to the amplitude of a single setting current pulse. Once set, it will deliver indefinitely an output proportional to the setting. This may be explained in a simplified manner as follows: Consider first the transfluxor in its blocked condition. Let there be a setting current pulse through winding W_1 of a chosen amplitude and, of course, of a polarity opposite that of the original blocking pulse. A magnetizing force H, proportional to this current, is produced around the large hole. This force in the magnetic material is greatest at the periphery of the hole and diminishes gradually with distance. In the case of a circular aperture it is inversely proportional to the radius. Therefore, for the given selected amplitude of the setting current pulse, there will be a critical circle separating an inner zone, in which the magnetizing force is larger than the threshold magnetizing force H_c required to reverse the sense o flux flow, and an outer zone, where this field is smaller than the threshold value. These two zones are shown in Fig. 2. Consider now the alternating magnetomotive force on leg *3* produced by an indefinitely long sequence of pulses of alternating polarity. The first pulse, applied to leg *3* in a direction to produce downward magnetization in leg *2*, can change only that part of the flux in leg *2* which is directed upwards, namely that part which has been set or trapped into that leg by the setting pulse. This changing part of the flux will flow through

J. A. RAJCHMAN and A. W. LO are with the Radio Corporation of America, Princeton, N. J.

George R. Briggs has greatly contributed to the understanding of the operation of the transfluxor and helped in many experimental phases of the work.

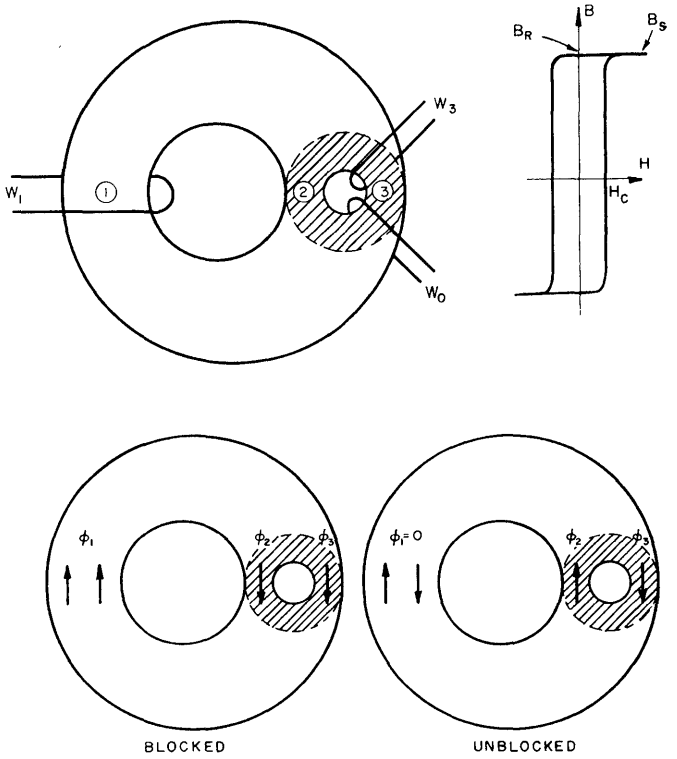


Fig. 1 (above). Principle of transfluxor

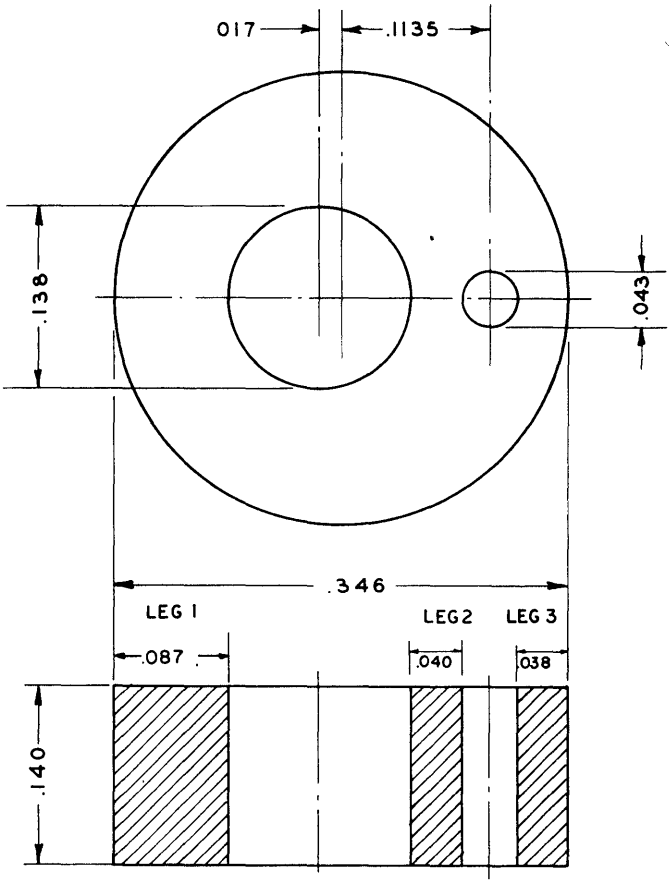


Fig. 3 (right). The prototype transfluxor

leg *3* until leg *2* reaches its original downward saturation. The amount of flux set into leg *2* is therefore transferred to leg *3*. The next pulse, applied to leg *3*, will saturate it to its original downward direction and thereby retransfer the trapped amount of flux back to leg *2*. There is no danger of any transfer of flux to leg *1*, since the magnetizing path is much longer through that leg, and, once leg *3* is saturated, no further flux flow is possible however intense this second pulse. It is apparent, therefore, that the succession of pulses of alternating polarity on leg *3* will cause an interchange between legs *2* and *3* of an amount of flux just equal to that initially set into leg *2*. This will produce across the output winding W_0 an indefinitely long train of voltage pulses. The magnitude of the output or the analogue information can be thought of as stored in terms of the flux in leg *1* (or algebraic sum of fluxes in legs *2* and *3*), just as was the case for the on-off information. This stored flux is not affected by the output-producing interchange of flux between legs *2* and *3*.

There is a possibility that, in the blocked condition, or any intermediary set condition, a sufficiently large alternating current in the phase tending to produce counterclockwise flux flow could in fact change the flux in leg *3* by transferring flux to leg *1*. There is, therefore, a limit to the permissible amplitude of the energizing alternating current because of the possibility of spurious unblocking. The limiting amplitude is increased by the use of unequal hole diameters rendering the flux path via legs *1* and *3* much longer than via legs

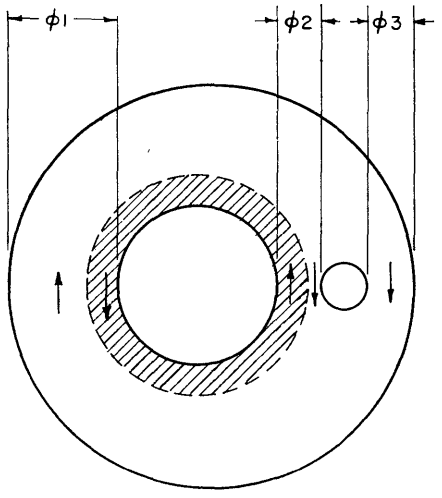


Fig. 2 (left). Setting the transfluxor to a level in a continuous range

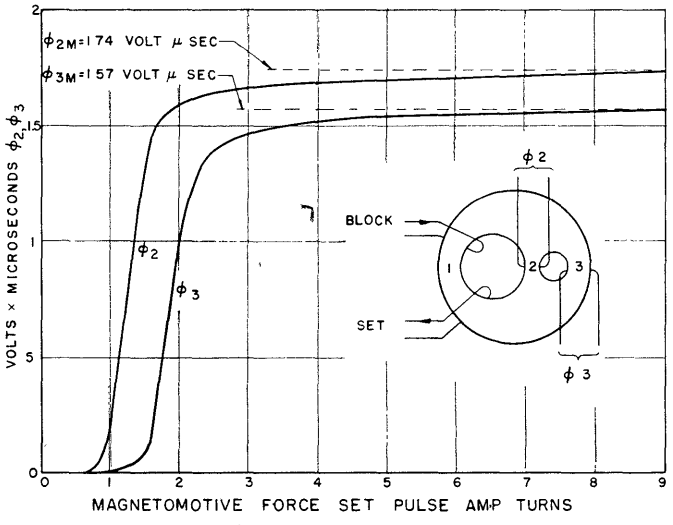


Fig. 4 (right). Inherent setting characteristic

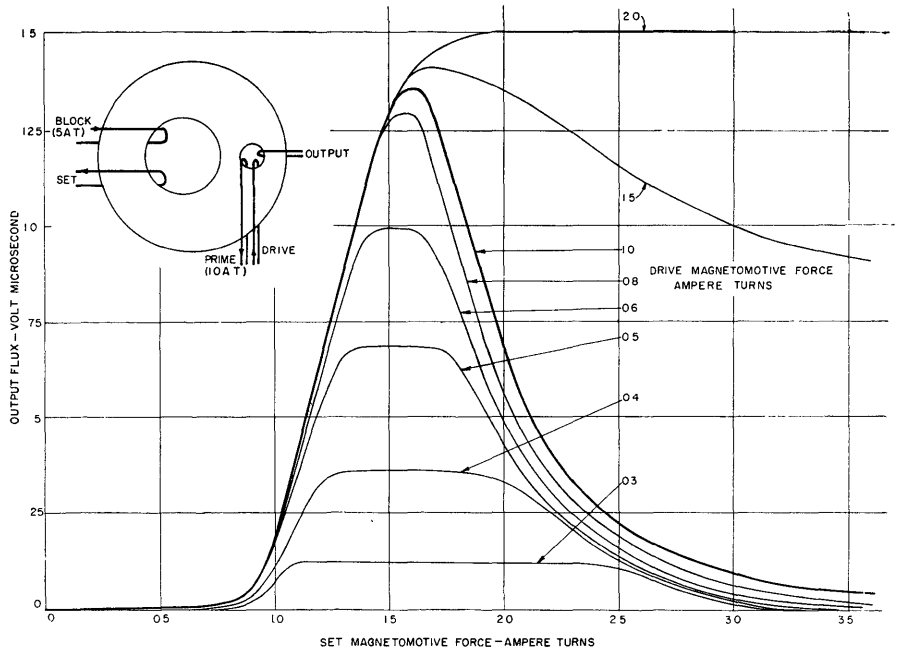


Fig. 5. Setting characteristic—asymmetrical energization

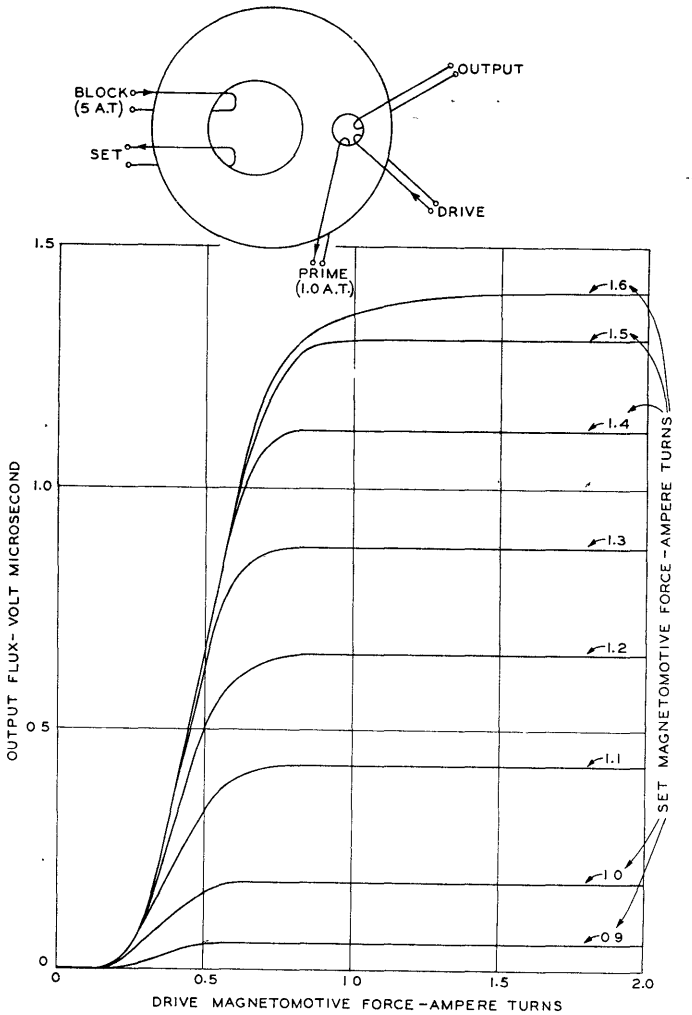


Fig. 6. Driving characteristic—asymmetrical energization

2 and *3*. There is no danger of spurious unblocking by the alternating current in the phase tending to produce a clockwise flux flow, since in this phase leg *3* is being magnetized in the direction in which it is already saturated. Therefore, there is a considerable advantage in using asymmetric-energizing alternating current or a train of interlaced relatively large driving pulses (clockwise) and relatively small priming pulses (counterclockwise). The driving pulses, which cannot possibly spuriously unblock a blocked transfluxor, can be arbitrarily large, with the result that, when the transfluxor is unblocked by proper setting, these pulses may not only provide the required minimal reversing magnetizing force around the small aperture, but also provide substantial power to deliver large output currents. The priming pulses must be of sufficient magnitude to provide the required magnetizing force around the small aperture, but insufficient to provide it around both apertures.

The Prototype 2-Aperture Transfluxor

Inherent Characteristics

The core of the transfluxor for which the characteristics are given in the following was made of magnetic ceramic material (manganese-magnesium ferrospinal; 30 per cent MnO, 30 per cent MgO, 40 per cent Fe_2O_3), of composition and processing identical to those used for memory cores. The dimensions are shown in Fig. 3.

The knowledge of the actual amounts of flux set in leg *2* and leg *3* by a pulsed magnetomotive force applied to leg *1* can be used to evaluate the idealized explanation of the operation given in the foregoing as well as to predict the detailed operation. These inherent setting properties, illustrated in Fig. 4, were obtained as follows: Leg *1* was subjected first to a relatively large blocking pulse of 5 ampere-turns magnetomotive force, and then to a setting pulse the amplitude of which is the abscissa of the plot. The instantaneous voltages on 1-turn windings linking legs *2* and *3* were integrated throughout the duration of the setting pulse, including the rise and decay of the pulse. These integrated values are the irreversible, or net, flux changes ϕ_2 and ϕ_3 produced by the setting, and are shown as the ordinates of the plot.

Ideally, the variations of the fluxes ϕ_2 and ϕ_3 can be predicted by considering the location of the closed line of magnetic induction passing through leg *1*

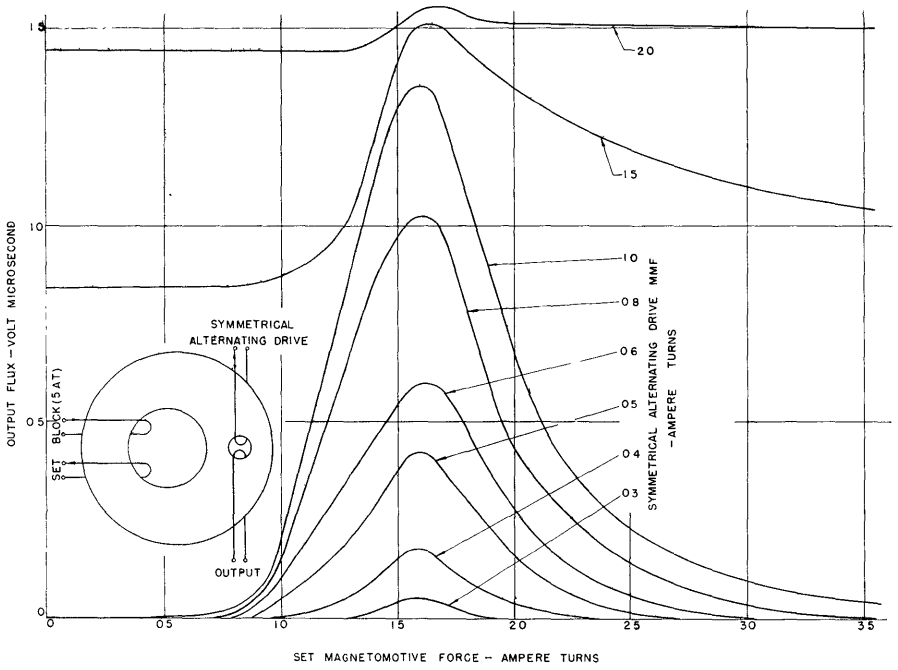


Fig. 7. Setting characteristic—symmetrical energization

along which the magnetizing force, because of the set current, is just equal to the threshold magnetizing force required to reverse the sense of flux flow, as was explained in the foregoing. This boundary between reversed and nonreversed senses of flux flow is approximately a circle the radius of which increases linearly with setting current. No flux change occurs until this circle reaches the edge of the large aperture ($\phi_2=\phi_3=0$). After this first threshold is exceeded, the flux in leg *2* increases linearly with the radius of the limiting circle as it sweeps across the width of the leg, while the flux in leg *3* remains zero. When the limiting circle has reached the smaller aperture, the direction of flux across the entire width of leg *2* is reversed. A further increase of setting current will increase the length of the boundary, which may then deviate somewhat from a circle, until it reaches the inner edge of leg *3*. At this second threshold value, the flux in leg *3* will start to increase linearly until the boundary has swept across the entire width of that leg.

The curves of Fig. 4 only approximate the sectionally linear curves predicted by the above idealization, because the real material is characterized by a family of hysteresis loops deviating appreciably from the ideal rectangular shape. The rounding-off of the corners and the asymptotic saturation characteristics are, therefore, not surprising. It is worth noting that leg *3* exhibits a substantial flux setting before leg *2* reaches saturation and also that its saturated value is smaller because the width of this leg happens to be somewhat narrower than that of leg *2* (see Fig. 3).

Operating Characteristics

The amount of flux which is interchanged between legs *2* and *3* for a given setting by priming and driving pulses, determines the output from the transfluxor and will be referred to as the output flux. The setting pulse will determine the maximum interchangeable amount of flux while the priming and driving pulses will determine what part of that flux is actually interchanged. It is convenient to consider as setting characteristics the plots of the output flux as a function of the setting magnetomotive force for given priming and driving pulses, and as driving characteristics the plots of the output flux as a function of the driving pulse for given setting and priming pulses. The main mode of operation consists of blocking and setting on leg *1*, priming and driving on leg *3*, and deriving the output from leg *3*. There are two cases of interest: (1) asymmetrical energization consisting of unequal drive and prime pulses useful when efficient loading is desired, and (2) symmetrical energization consisting of equal drive and prime pulse or sinusoidal current encountered in small signal a-c transmission.

The setting characteristics for asymmetrical energization shown in Fig. 5, can be explained as follows: the flux change ϕ_{1S} in leg *1*, set into it by the setting pulse, will divide itself, in general, between legs *2* and *3* where the changes of flux ϕ_{2S} and ϕ_{3S} will be produced ($\phi_{1S}=\phi_{2S}+\phi_{3S}$). When the set flux ϕ_{1S} is smaller than the maximum flux containable in the narrowest leg, (in this case leg *3*, which is slightly narrower than leg *2*) the setting is referred to as normal. For normal setting the first prime pulse on leg *3* will saturate leg *2* in its original blocked direction of saturation, transferring the flux ϕ_{2S} set in leg *2* to leg *3*, thereby concentrating the amount of flux equal to ϕ_{1S} in leg *3*. When the drive pulse is of an amplitude just sufficient to saturate the entire width of leg *3* (1 ampere-turn), the flux concentrated in that leg by the prime pulse will be retransferred to leg *2*. The steady-state interchanged output flux between legs *2* and *3* will, therefore, be precisely that set initially in leg *1*, which in turn is equal to the sum of fluxes ϕ_2 and ϕ_3 shown on the inherent set curves of Fig. 4. This flux increases linearly up to the limit of normal setting which is about 1.5 ampere-turns.

When the drive is not sufficient to transfer all the flux of leg *3* back to leg *2*, only a part of the interchangeable flux set in leg *3* will be, in general, actually interchanged. This explains the shape of the setting characteristic for lower drives (less than 1 ampere-turn). These curves follow the main characteristic only up to the value of drive for which the whole set-in flux can be transferred.

When the flux ϕ_{1S} set in leg *1* exceeds that containable in the wider of the two legs, *2* or *3*, it oversets the transfluxor by starting to reverse the fluxes in all legs, producing blocking. Therefore, the amount of output flux interchanged between legs *2* and *3* diminishes. The rate of decrease of the output flux with increasing setting current is smaller in the overset region than the rate of increase in normal region because of the asymptotic nature of the curve near saturation (see Fig. 4). The loss of output flux due to oversetting may be corrected by overdriving. A large enough driving pulse on leg *3* will saturate it completely in the original blocked downward direction by transferring flux first to leg *2* until it is saturated and then the excess to leg *1*. Therefore an overset and overdriven transfluxor will produce the maximum output, as shown by the characteristic curves of Fig. 5.

The driving characteristics of Fig. 6. for the same asymmetrical energization, show that arbitrarily large driving pulses may be used without disturbing normal setting as was explained before. There

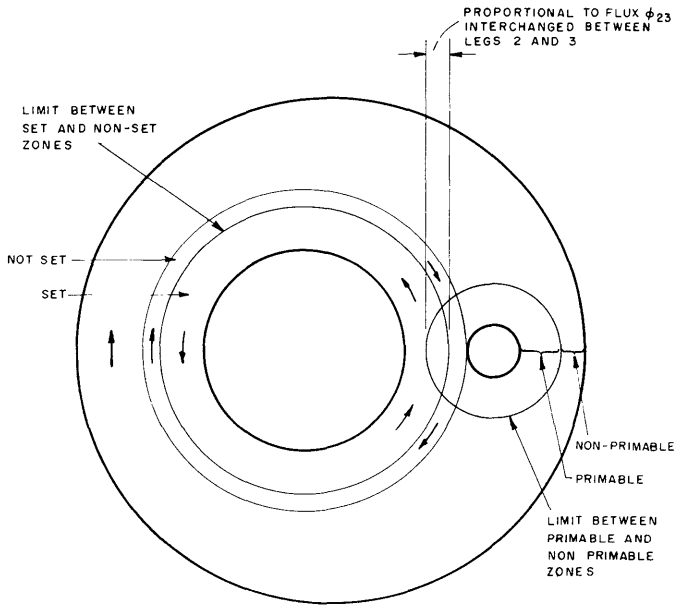


Fig. 8. Zones of setting and primability in the transfluxor

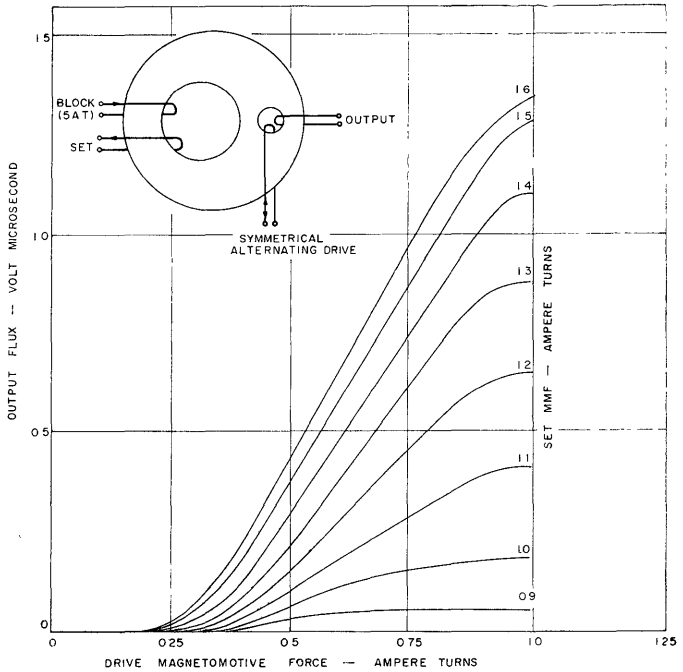


Fig. 9. Driving characteristic—symmetrical energization

is a threshold value of 0.2 ampere-turn, corresponding to the value of magnetomotive force producing a magnetizing force just equal to the coercive force at the periphery of the small aperture, below which there is no interchange of flux between legs *2* and *3*. Each curve exhibits a sharp threshold followed by an approximately linearly rising curve.

The setting characteristics for symmetrical energization with equal drive and prime pulses, shown on Fig. 7, may be explained by considering first the idealized situation for a normal setting in which two zones are created in leg *2*; one near the large aperture where the flux is reversed, and one near the smaller aperture where it remains unaffected (see Fig. 9). When the prime pulse on leg *3* is insufficient to produce a magnetizing force greater than the coercive force at the location of the boundary between the two zones, no flux will be transferred to leg *3*. There will be a definite value of prime for a given setting, for which transfer will start to occur. The amount of transferred flux will be proportional to that portion of the cross section of leg *2* which is included between the boundary separating the set and nonset zones around the larger aperture, and the boundary between the primable and nonprimable zones around the small aperture, as shown in Fig. 9. This flux increases with setting, for a given prime, at a rate which is independent of the priming value. When the priming pulse becomes large enough to produce interchange of flux between legs *3* and *1*, it will effectively produce a setting of leg *1* even when no previous setting on leg *1* was applied. This is the spurious unblocking due to symmetrical drive and prime mentioned earlier. The characteristics of Fig. 7 have approximately the shape to be expected from these considerations.

The driving characteristics for the symmetrical energization case are shown in Fig. 8. The range of the priming pulses, and consequently also of the driving pulses, is restricted to the values (up to 1 ampere-turn) which do not produce spurious settings. For a given setting, the amount of interchangeable output flux is proportional to the flux in leg *2* which is both set and primable. The threshold value of prime, below which no flux is interchanged, is smaller the greater the set, since the boundary between set and nonset zones is closer to the small aperture in this case.

In the main mode, with either asymmetrical or symmetrical energization, the input control (setting) circuits are completely separated from the output circuits, since the block and set winding is on leg *1* and the priming, driving, and output windings are on leg *3*. There is negligible coupling between the control and output circuits because there is practically no interchange of flux between legs *1* and *3* in normal operation. For example, in the prototype transfluxor set to maximum, the back-and-forth interchange of flux between legs *2* and *3* is 1.5 volt-microseconds, and this is accompanied by only 0.01 volt-microsecond of flux change in leg *1*, or less than 1 per cent. Furthermore, normal setting pulses produce only slight changes of flux in leg *3*, of only about 5 per cent of the flux set into leg *2* by maximum setting. If blocking occurs after driving, rather than priming, it produces a negligible flux change in leg *3* since the drive pulse has already saturated that leg in the direction of blocking.

An arbitrarily large priming pulse, rather than one needing to be of a prescribed amplitude, can be used by priming on leg *2* rather than on leg *3*. After normal setting, the prime pulse will saturate leg *2* downward to its original blocked direction and transfer flux to leg *3* rather than leg *1* because the flux path is shorter. A further increase of the prime pulse on leg *2* produces no further flux change because leg *2* is saturated. When the transfluxor is overset, the prime pulse on leg *2* transfers to leg *3* that part of its flux that leg *3* can accept and forces the excess flux to leg *1*, where it readjusts the setting. Oversetting is thus corrected for by the first prime pulse which causes the transfluxor to be set for maximum output. In this mode of priming on leg *2*, the setting pulse induces a voltage in the prime winding. Thus, there is an interaction between setting and priming circuits. This interaction can be tolerated in the many practical cases in which the priming winding is in the plate circuit of a vacuum tube.

In some on-off applications of the transfluxor, it is convenient to set on leg *2* rather than on leg *1* to avoid the possi-

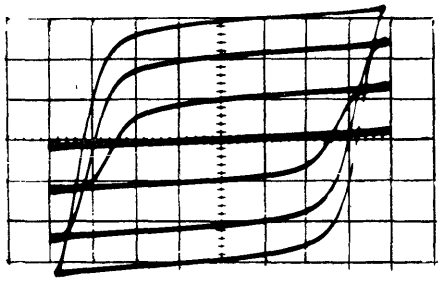


Fig. 10. Hysteresis loops of the output magnetic circuit for various settings

bility of oversetting. The setting amplitude is not critical as long as it is greater than some required minimum (about 1.3 ampere-turns). In this case the large setting pulse producing an upward magnetomotive force on leg *2* forces the flux of leg *2* to reverse its blocked direction. Since flux flow through leg *3* is impossible, necessary continuity of flux is satisfied by an interchange of flux between legs *2* and *1*, which leaves leg *1* with practically zero flux and leg *2* with an upward saturation. This is the unblocked state of the transfluxor. The pulses on legs *1* and *2* serve respectively as the blocking and unblocking pulses and close or open the transfluxor gate. In this mode there is direct coupling between the setting winding on leg *2* and the output, since the output flux is precisely the interchangeable flux between legs *2* and *3*. Here again this coupling is tolerable in many practical cases, e.g., when the setting winding is in the high impedance circuit of a vacuum tube. It is possible also to use leg *2* both for setting and priming.

Output From the Transfluxor

The transfluxor exercises control by means of the amount of flux which can be transferred for an indefinitely long time between legs *2* and *3*, and which amount can be set by a single pulse to any desired value in a continuous range. This back-and-forth transfer of flux between legs *2* and *3* can be considered also as a back-and-forth reversal of flux around the small aperture along a path which is effectively the output magnetic circuit and may be characterized by a conventional hysteresis loop relating the flux flow and the magnetomotive force on leg *3* producing it. Fig. 10 shows oscilloscope traces of a family of such loops, each obtained for a different setting, including the blocked and maximum settings.

It is apparent that the transfluxor operates as if the output magnetic circuit consisted of a conventional 1-aperture core with the essential property that the effective cross-sectional area of that core can be adjusted by a single set pulse to any desired value from practically zero to a maximum value equal to the physical cross-sectional area of its smallest leg.

The relations which exist between the primary and secondary circuits of a pulse transformer apply equally well to the output circuit of the transfluxor, provided account is taken of the definite set cross-sectional area of the equivalent core and the properties of the material of the core. These include the shape of the hysteresis loops and the intrinsic possible rates of flux reversal. The salient properties of the output circuit can be illustrated by the cases of very high and very low impedance loading.

Output voltage wave forms are shown in the oscilloscope traces of Figs. 11a and 11b for the case of an open-circuited output winding, i.e., a very high impedance load. The traces are for the various values of setting, as indicated. For the relatively large (5 ampere-turn) and fast rising (3.3 ampere-turns per microsecond) drive of Fig 11a, and for the relatively small (1 ampere-turn) and slow rising (0.7 ampere-turn per microsecond) drive of Fig. 11b, the voltage peaks vary linearly with the current settings. The unloaded transfluxor may therefore be considered as a device furnishing a controllable voltage. The short flux-reversal time of 0.1 microsecond for large drive is noteworthy.

The ratio of voltage output at maximum setting (1.5 ampere-turn) to that at zero setting or blocked condition, is 15 to 1 for the large drive of Fig. 11a, and 50 to 1 for the smaller drive of Fig. 11b. The slight voltage output in the blocked condition results from the elastic or reversible flux excursion because of the lack of perfect saturation at remanence of the material composing the transfluxor's core. There is actually no measurable irreversible output flux for zero setting. (See Figs. 5, 6, 7, and 8.)

The output voltage wave forms, developed across a very low resistance load, for different settings, are shown in Fig. 12a. The setting is seen to control the duration of the pulses rather than the voltage maximums which are very flat and almost the same for all settings. This results from the fact that the counter-magnetomotive force due to the secondary current tends to keep the rate of change of flux constant. The output current in a low resistance load for a transfluxor set to maximum is shown in Fig. 12b for different drive currents. The output current increases linearly with drive current; the heavily loaded transfluxor

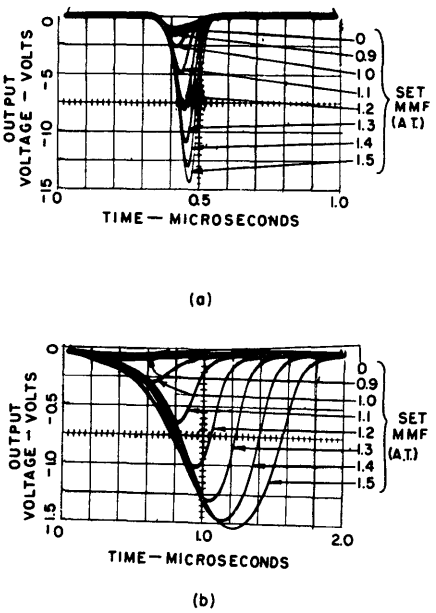


Fig. 11. Voltage output wave forms for open circuit

(a) Large drive
(b) Small drive

may therefore be considered to be a current transformer. The efficiency of power transmission is high for large drives since only a small part of the drive is wasted in magnetizing the output magnetic circuit and the major part neutralizes the secondary countermagnetomotive force and produces the output current. Efficiencies of 75 per cent have been obtained. This important property of the transfluxor reflects the advantage of using unsymmetrical output circuit energization, i.e., small and slowly rising priming current pulses producing negligible output and no spurious setting, and fast-rising large-amplitude drive pulses producing the useful output.

In the foregoing illustrations the transfluxor was used as an adjustable transformer with a primary winding energized by a current generator and a secondary winding carrying the load. In many applications it is convenient to use it as an adjustable inductance with a single winding (on leg *3*) in series with a voltage generator and the load. In that case a high output is obtained when the transfluxor is blocked and a low output when it is unblocked, but besides this inversion, all operations as an inductance or as a transformer are similar.

The material composing the core of the prototype transfluxor permits short switchover times. Conventional memory cores, made of this material, switch in about 1.5 microseconds when driven by a current giving optimum discrimination in a 2-to-1 driving system. In the

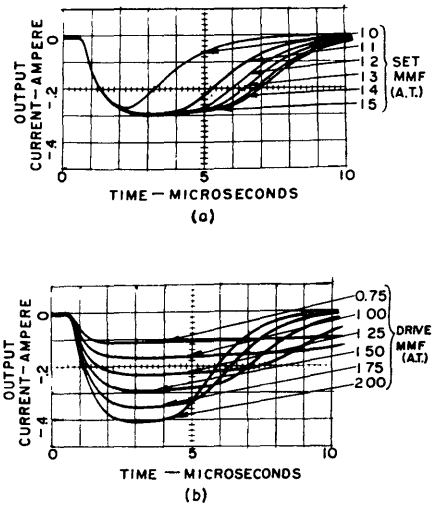


Fig. 12. Voltage output wave forms for low resistance load

(a) Various settings
(b) Various drives

range of setting pulse amplitudes shown in the operating characteristics, the setting requires about 2 microseconds. As was noted, very much shorter output pulses are possible when strong drives are used. The repetition rates of the driving and priming pulses or the frequency of sinusoidal a-c energization can be 1 megacycle or more without any appreciable heating of the core of the transfluxor. There is no lower limit to the frequencies used in the output circuit, other than that imposed by the practical use of the low voltages resulting therefrom.

Applications of the Transfluxor

Information Registers for Digital Computers

The transfluxor can be operated as an on-off gate utilizing only the blocked and the set-to-maximum or unblocked settings. The amplitude of the blocking pulse on leg *1* is not critical provided it is greater than some minimum. Similarly, the unblocking pulse can be rendered uncritical if applied to leg *2*, as was mentioned in the foregoing, or if a third setting aperture is provided, as will be described. In this operation the transfluxor performs the same function in digital computing applications as does a gate controlled by a conventional tube or transistor flip flop. Once set, the gate remains open or closed without requiring any holding power. A bank of transfluxors constitutes a register for storing a number of binary signals from which read-out signals can be obtained any number of times without destroying the stored information. These signals can be conveniently generated from a common source gated by the individual transfluxors. No signals in the setting circuits result from the interrogation when proper arrangements are made.

Random-Access Memory with Non-Destructive Read-Out

An array of transfluxors can be used as a random-access memory with so-called nondestructive read-out, i.e., where the read-out is obtained without changing at any time the physical state representing the information. The two stored states are the blocked and unblocked remanent conditions of the transfluxor. Current coincidence can be utilized for selection, as is done in conventional core memories. Consider two selecting windings on leg *1* and two selecting windings on leg *3* of each transfluxor. Let these windings be connected in series by rows and columns as shown for simplicity by single linking wires on Fig. 13. Address-selecting writing pulses are applied simultaneously to one row and one column winding linking leg *1*. The additive effect of these pulses on the selected transfluxor at the intersection of the row and column windings is sufficient to produce a setting, but the amplitude of the pulses is insufficient to affect the transfluxors on which they act singly. The direction of the writing pulses determines whether the selected transfluxor is set to the blocked or unblocked condition. Reading, based also on current coincidence selection, is obtained by applying pulses of the proper amplitude to the selected row and column winding linking leg *3*. A read-out is obtained by a pair of pulses on each selecting line, one in the prime and one in the drive direction. As a result, fluxes in legs *2* and *3* reverse back and forth and return to their initial state, if the transfluxor is unblocked, or remain in that initial state, if the transfluxor is blocked. These flux reversals can be detected as induced voltages on a common read-out winding linking leg *3* of all transfluxors. (See Fig. 13.)

Coincident current selections for write-in and read-out are possible because there are thresholds below which pulses linking leg *1* and leg *3* produce negligible flux changes. Satisfactory operation is obtained with the prototype transfluxor when the selecting currents are approximately equal to the threshold values of setting and prime drive shown on the characteristic of Fig. 7. Somewhat better results are found when the blocking write-in pulses are larger than the setting write-in pulses. The coincident current

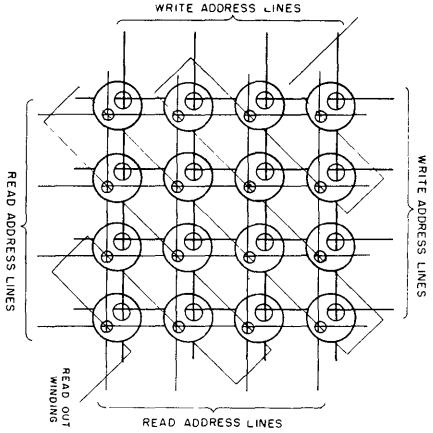


Fig. 13. Array of transfluxors used as random access memory with nondestructive read-out

selection principle can be extended to the simultaneous addressing of many plane arrays used in parallel by using selective inhibiting of the planes, as is customary with core memories.

Read-out from a magnetic storage device must necessarily be dynamic, since induced voltages are possible only by changing flux. Nevertheless, in the transfluxor memory, the read-out may be considered to be nondestructive, because the flux in leg *1* is not altered by the interrogating pulses, retains at all times the stored information, and yet its value determines whether or not flux in legs *2* and *3* will be interchanged as a result of interrogation. The second interrogating pulse is necessary to restore the altered states of legs *2* and *3* due to the first, in a manner similar to the rewrite pulse in destructive read-out memories required when a read-out signal is obtained, but with the essential difference that its unconditional occurrence at every read-out is not dependent on the presence of the read-out signal.

In the transfluxor coincident current memory the read-out circuits are very simple since no feedback into the write-in circuits are required. A further, perhaps more important, advantage, is the possibility of simultaneously writing-in and reading-out on two unrelated addresses, by energizing the proper lines of the independent write-in and read-out selecting grids. This possibility may speed up the operation and simplify the logic of some types of computing machines.

Channel Selector

Transfluxors may be used with advantage to select one channel out of many for transmission of modulated signals. The transmission from, or to, a common channel to, or from, each one of a

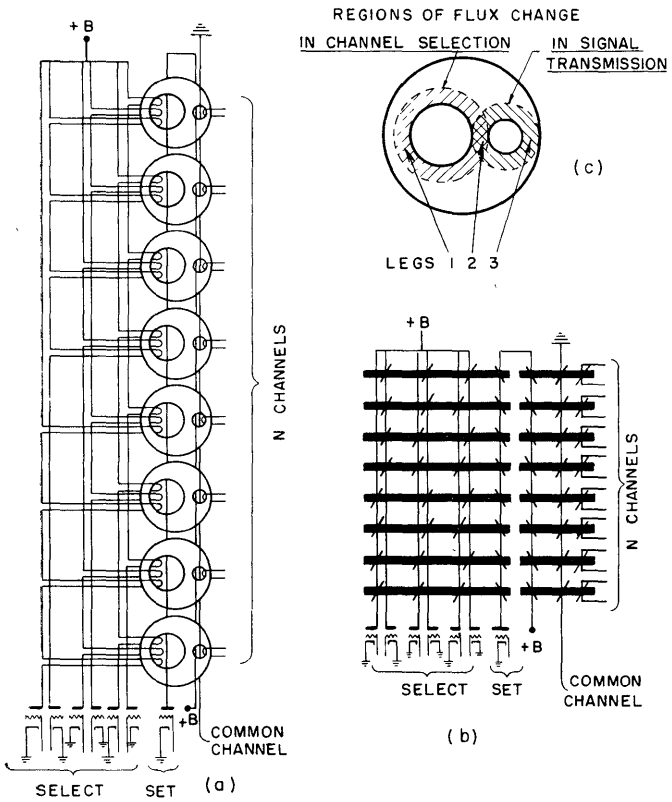


Fig. 14 (left). Transfluxor channel selector

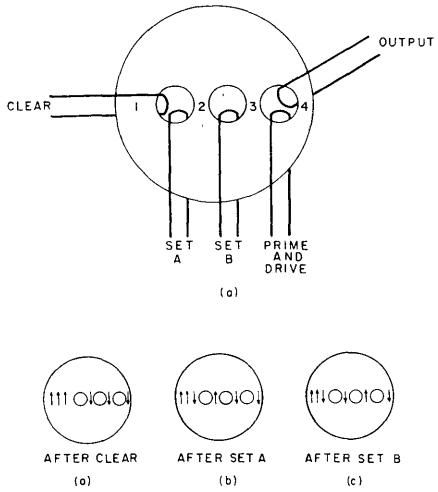


Fig. 15 (right). 3-aperture transfluxor sequential gate

number of selectable channels is controlled by a transfluxor. All transfluxors are blocked except one which is set and which determines the selected channel.

A channel selector for selecting one out of $N=2^n$ channels in response to n binary pulsed signals is illustrated in Fig. 14 for the case of eight channels and three binary inputs. The selection system is similar to that used in combinatorial decoding switches[3] using conventional cores. For each binary input there is a pair of conductors, one of which links leg *1* of half the transfluxors and the other, the other half. The division of transfluxors in halves by the various input pairs is by juxtaposed halves, interlaced quarters, interlaced eighths, etc., so that the pattern of linkages is according to a binary code. To select a channel, current is sent through one or the other conductor in each pair in a direction tending to block the transfluxors which are linked by these conductors. For every combination of inputs there will be one transfluxor, and one only, which is not linked by conductors carrying the blocking currents. The transfluxor, thus selected, is set by a current pulse sent through a winding which links leg *2* of all transfluxors. The setting pulse, occurring while the selecting current pulses are on, has insufficient amplitude to overcome the blocking effect of even a single one of these blocking currents and therefore sets the selected noninhibited transfluxor only. The setting is to maximum, but there is no danger of oversetting, since leg *2* is used for setting, and the region of flux change during selection is around the large aperture only. The selected transfluxor remains set until a different combination of input pulses is applied to the selector, at which time a new transfluxor is set, and the previously selected one is automatically blocked.

Flux changes around the small aperture are possible only in the selected unblocked transfluxor of which the output magnetic circuit may be considered to be a regular transformer. Since a transformer transmits in either direction, the selector can be used either to transmit from the common channel to a selected channel or vice versa, depending on which channel corresponds to the primary winding. The primary winding can be energized by symmetric alternating current, provided that the resulting magnetization around the small aperture exceeds a certain threshold, but does not exceed a value producing spurious unblocking. In the prototype transfluxor the permissible range is from 0.25 to 1 ampere-turn. With asymmetric primary excitation a greater efficiency of power transmission is possible, as was explained in the foregoing. In this case it may be convenient to prime the transfluxor by a common winding linking leg *2* of all units (not shown on Fig. 14), to use a fixed amplitude prime pulse (1 ampere-turn for the prototype) and to modulate by the amplitude of the drive pulse.

The selector of Fig. 14 is shown with tube drivers to illustrate fully a concrete example. For simplicity, single turn windings are shown. Fig. 14b is a simplified representation of the selector, shown conventionally in Fig. 14a. The apertures of the transfluxors are represented by heavy horizontal lines which are assumed to be linked through by the vertical conductors when a 45 degree line is drawn at the intersection, the direction of inclination of the line denoting the direction of linkage.

It is possible to use the channel selector not only to gate sustained modulated signals, but also to control the amplitude of the transmitted signal in the selected channel according to the amplitude of a single-control current pulse. To accomplish this, the control pulse is sent through the set winding linking leg *2* of all transfluxors simultaneously with the selecting pulses, and sets the noninhibited selected transfluxor to a particular level in a continuous range dictated by its amplitude. Constant amplitude energization of the output circuits of all transfluxors produces through the selected one a sustained output of an amplitude determined by this level. The energization can be symmetric or asymmetric, a-c or by pulses.

The arbitrary selection, in any desired order, of the transmission channel in the channel selector described earlier is accomplished by considering the large aperture of the transfluxor as if it were a simple core and linking it through systems of selecting windings known in conventional core decoding switches. Similarly, a commutator switch for establishing transmission through the channels in ordered succession can be made by coupling the large aperture of successive transfluxors by the circuits used in con-

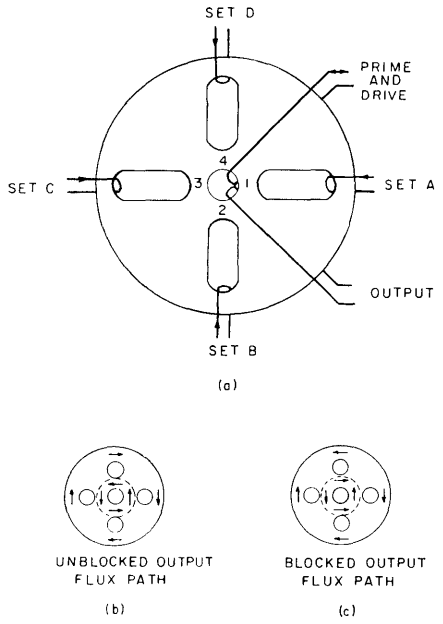


Fig. 16. 5-aperture "flower" transfluxor

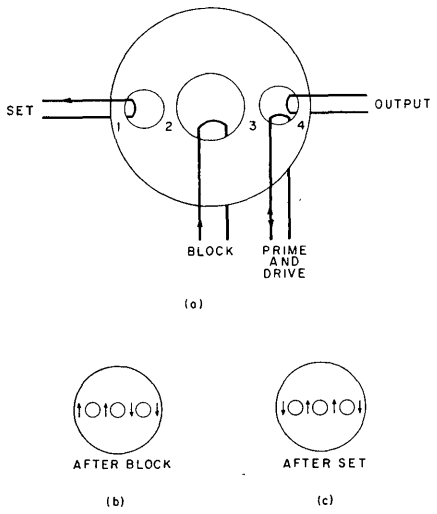


Fig. 17. Elimination of oversetting with an additional aperture

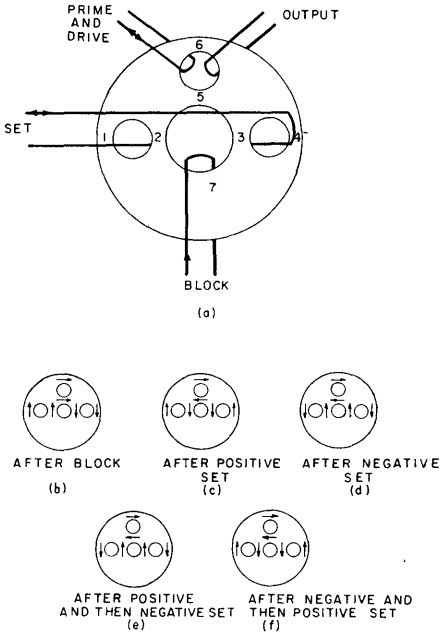


Fig. 18. Setting with a pulse of either polarity

ventional magnetic shift registers. In the core switches the selected core produces a transient output at the instant of selection only, while in the analogous transfluxor switches, the selected transfluxor opens for as long as desired a channel for bidirectional transmission of a-c or pulsed signals.

Such channel selector switches for routing modulated signals, either in an order depending on a code, or else in a predetermined succession, can be useful in a variety of applications, for example: multiplexing system for telegraphy or telephony communications, multiplexing for telemetering or automatic controls, switching of heads of a magnetic drum, central exchange in large systems handling digital information, telephone exchange systems, and so forth.

Other Applications

In digital computers, the transfluxor can also be used as a logical element as well as an auxiliary to energize incandescent lights for providing bright visual indicators with negligible loading on the pulse circuits.

The transfluxor is inherently an indicator of the peak pulse in an arbitrarily long pulse train. This property can be useful, for example, in nuclear instruments. In pulse-modulation systems, in addition to channel selection, the transfluxor can convert pulse into amplitude information or convert randomly occurring pulses into regularly spaced pulses. The control of automatic mechanisms by single command pulses, often a prerequisite for automation systems operated from a central computer, may easily be obtained with transfluxors.

These examples are cited to illustrate the variety of applications of the transfluxor.

Multiaperture Transfluxors

The 2-aperture transfluxor has been discussed in detail to illustrate the principles of operation and the general properties of the device in one of its simplest forms. The use of more than two apertures creates many new modes of flux transfer and broadens the kind and number of switching and storing functions, making possible many novel applications. A few illustrative examples of multiaperture transfluxors are described in the following.

A transfluxor with three apertures in a row, as shown in Fig. 15, can be operated as a 2-input sequential gate. An output is produced if the two inputs A and B are applied to it in the order AB, and no output is produced if either input is missing or if the two inputs are applied in the order BA. The operation is illustrated by the symbolic diagrams, Figs. 15b, 15c, and 15d. After a clear pulse the legs *2*, *3*, and *4* are saturated downward. The output flux path around the last aperture via legs *3* and *4* is blocked and neither the prime nor the drive pulse can produce any flux change. The flux path around the second aperture via legs *2* and *3* is also blocked so that the signal B cannot produce any flux change. However, the flux path around the first aperture, via legs *1* and *2*, is not blocked and the signal A can reverse the direction of flux in leg *2* by transfer of flux to leg *1*. If A was present and leg *2* was reversed, the flux path via legs *2* and *3* is unblocked, with the result that the occurrence of B can now reverse the flux of legs *2* and *3*. This returns leg *2* to its original downward direction, reverses leg *3* and unblocks the flux path via legs *3* and *4*. Consequently the output flux path is now unblocked, and a succession of priming and driving pulses will produce an output for as long as desired. This 3-aperture transfluxor can also be operated with intermediate setting. The analogue intelligence can be carried by either signal A or B or both.

The transfluxor with five apertures arranged in a flower-like pattern, Fig. 16, can be operated as a 4-input AND gate. The occurrence, in any order, of all four input signals A, B, C, and D is required to open the gate. The principle of operation depends upon the fact that the output flux, via legs *1*, *2*, *3*, and *4*, around the central aperture can be blocked by any one of the four legs and is unblocked only when the senses of flux saturation around the central hole in all legs are the same. There are two unblocked states corresponding to the two senses of flux rotation around the small aperture. In many applications it is convenient to use one leg as a reference, yielding a 3-input gate, and to eliminate one of these states.

A third aperture added to the two of the transfluxor described earlier, in detail, can eliminate any possibility of oversetting. The four legs *1*, *2*, *3*, and *4* of the transfluxor illustrated in Fig. 17 are of equal cross section. The amount of flux that can be set is limited by the width of leg *1* to precisely the amount

which leg *3* can accept. Therefore, when the setting pulse on leg *1* becomes larger than required to transfer flux to the first filled closer leg *3*, no further flux is available for transfer to leg *4*. In this 4-legged transfluxor, leg *2* is a dummy which remains always saturated in the same direction and provides the necessary return path for continuity of flux flow.

A transfluxor which can be set by either polarity of setting pulse can be obtained by using four apertures, as shown in Fig. 18. It is apparent that either a positive or a negative set pulse will cause leg *5* to reverse its flux. For a positive set pulse this will occur with corresponding reversals of legs *2* and *4*, and for a negative set pulse with reversals of legs *1* and *3*. The output flux path via legs *5* and *6* is unblocked by the setting of leg *5*. Blocking can also be of either polarity.

Conclusion

The transfluxor has the unique property of being able to control the transmission of electric power according to a stored level established by a setting pulse. In contrast to the magnetic amplifier in which the input command is not stored and must be present at all times, the transfluxor requires only a single setting. In contrast to the conventional memory core, the transfluxor is not only capable of storing a given amount of set-in flux, but also is capable of furnishing on demand, and for an indefinite length of time, an output according to the stored setting without affecting that setting in the least. In a sense, the transfluxor combines the functions of a magnetic amplifier and a memory core.

The multiaperture transfluxor core, made of square hysteresis-loop material, used at present for ring-shaped cores, is simple to manufacture. Like other magnetic elements it is a solid-state passive element which is rugged, stable in operation, and immune to permanent deterioration due to accidental overdriving of its associated circuits.

For these reasons it is believed that there is a great future for the transfluxors described in this paper, and similar ones that can be made with artifices based on manipulating the flux distribution in cores of square-loop-magnetic material having a number of apertures in various geometrical configurations.

References

1. The Transfluxor—A Magnetic Gate With Stored Variable Setting, **Jan A. Rajchman, Arthur W. Lo.** *RCA Review*, Radio Corporation of America, New York, N. Y., vol. XVI, no. 2, June 1955, pp. 303–11.

2. Magnistor Circuits, **R. L. Snyder.** *Electronic Design*, New York, N. Y., vol. 3, no. 8, August 1955.

3. A New Non-destructive Read for Magnetic Cores, **R. Thorensen, W. R. Arsenault.** *Proceedings of the Western Joint Computer Conference, March 1955*, Institute of Radio Engineers, New York, N. Y., 1955, pp. 111–16.

4. Static Magnetic Matrix Memory and Switching Circuits, **J. A. Rajchman.** *RCA Review*, Radio Corporation of America, New York, N. Y., vol. XIII, no. 2, June 1952, pp. 183–201.

Bilateral Magnetic Selection Systems for Large-Scale Computers

A. H. SEPAHBAN

SELECTIVE writing of information on a chosen channel of a large memory system (e.g., a magnetic drum memory) and selective reading of information from one out of many such memory channels can be accomplished by use of a single 2-way magnetic pyramid made solely of high quality magnetic saturable cores. A description is given of a working magnetic selection unit used in a large inventory control system with a few thousand magnetic drum channels.

A. H. Sepahban is with the Philco Corporation, Philadelphia, Pa.

The Megacycle Ferractor

T. H. BONN

THE ferractor is a magnetic amplifier designed to replace vacuum tubes in digital computer pulse circuits. Operation at information rates as high as $2^1/_2$ megacycles with moderate power gains and power levels has been achieved. This development represents an increase in the operating gain-bandwidth of magnetic amplifiers between one and two orders of magnitude over what has previously been reported.

This large step forward was due to a number of factors:

1. Improved circuits.
2. Improved methods of analysis of high-frequency magnetic amplifiers.
3. Improved magnetic materials and new developments in core construction.

Ferractors are readily adaptable to modular construction. In using them as building blocks the control and arithmetic sections of computers can be economically constructed with a minimum number of circuit types. All typical computer circuits: flip flops, binary counters, shift registers, etc., can be readily synthesized.

The construction of a special-purpose all-magnetic computer for military application, the Univac Magnetic Computer, is

T. H. Bonn is with Remington Rand Univac, Philadelphia, Pa.

Part of the work reported here was supported by the Air Force Cambridge Research Center, Air Research and Development Command.

nearing completion. This machine uses approximately 1,500 ferractors, 9,000 germanium diodes, and several dozen transistors. Only a few vacuum tubes are used in the circuits which generate the carrier essential to the operation of the magnetic amplifier. The information rate of the machine is 660 kilocycles. This was chosen as a conservative figure for the first attempt at a large system.

The basic circuit is the series magnetic amplifier in which the output circuit is fed from a low impedance source. This type of amplifier has the following advantages:

1. Power is distributed from a constant-voltage low-impedance pulse power supply instead of a constant-current high-impedance power supply as heretofore used in magnetic circuits. It is much easier to generate and distribute high frequency pulse power at a low-impedance level than at a high-impedance level.
2. Each amplifier performs the functions of pulse shaping and timing, in addition to power amplification.
3. The zero output signal is easy to handle by straightforward amplitude clipping techniques.
4. The circuits are simple and require a minimum of components.

The analysis of the high frequency operation of magnetic amplifiers includes consideration of all core and circuit tolerances as well as the high frequency properties of magnetic materials and the associated diodes. In order that system reliability be realized a standard minimum output signal from an amplifier with maximum load is assumed. The design is such that the minimum output signal which is obtained under worst operating conditions is sufficiently large to operate the intended circuits in the computer. The zero signal is effectively suppressed by the amplitude clipper referred to in the foregoing. Furthermore, the magnetic cores have the property of integrating the effects of signals applied to them. Therefore, they are not as sensitive to high frequency noise as other types of circuits.

An exhaustive study of the characteristics of all commercial magnetic materials and some specially made in the laboratory showed the superiority of *4-79* molybdenum permalloy for magnetic amplifier applications. The analysis of the gain of magnetic amplifiers shows clearly the detrimental effect of space factor on magnetic amplifier gain. The gain goes down as the ratio of magnetic material cross section to air cross section in the amplifier output winding decreases. A realization of the importance of space factor led to the use of a stainless steel bobbin as a support for the magnetic material and wire. It is possible to fabricate stainless steel with much thinner walls than the ceramics which are customarily used as bobbins for magnetic amplifiers. Difficulty is experienced in machining ceramics to a thickness smaller than 0.015 inch. As this thickness is approached the ceramic loses strength and becomes as fragile as an egg shell. On the other hand, stainless steel can be fabricated into bobbins with walls of thicknesses as small as 0.003 inch. These bobbins have adequate strength if they are handled with care.

A typical design of a core for the computer referred to above has 13 wraps of 1/8-mil *4-79* molybdenum permalloy tape, 1/32 inch wide, wound on a stainless steel bobbin, 0.1 inch diameter. The cores are machine wound and potted in hermetic seal headers. The headers are then mounted on printed circuit boards which also contain associated diodes and resistors. These printed circuit boards are plugged into the computer frame work.

Laboratory experience with the computer has shown the magnetic circuits to be very reliable. In low frequency applications magnetic amplifiers have been unsurpassed in reliability when properly designed and packaged. Now that high speeds have been achieved with magnetic amplifiers, speeds that are comparable to those which can be obtained with any other available method of power amplification, the magnetic amplifier will find wide use in high-speed pulse handling systems.

Purpose and Application of the RCA BIZMAC System

W. K. HALSTEAD J. W. LEAS J. N. MARSHALL E. E. MINETT

THIS presentation before the 1956 Western Joint Computer Conference is the first complete public presentation of the RCA (Radio Corporation of America) BIZMAC system. In November 1955, the system passed acceptance tests and was delivered in December to the Ordnance Tank-Automotive Command (OTAC) of the U. S. Army Ordnance Corps in Detroit, Mich.

This set of papers covers the system philosophy embodied in the equipment. It attempts to set forth the reasoning and planning behind the development of the data-handling equipment.

The RCA BIZMAC system is an accounting system in the broad sense that it aims to mechanize all the functions of record keeping, handling of data, calculation and decision, summarization, and prediction which are the basic functions of industrial paper work. It is a system developed to meet the requirements of such work, and not an attempt to apply an available component to the extent that it can be useful. The measure of its success is cost reduction to its user. By making a machine do routine "mental" work, it will help to free many people from tasks which are dull and repetitious, to be available for more creative and challenging work.

In the Ordnance application, the job to be performed is supply and stock control of tank and automotive parts. It is a task of some magnitude. The system must be able to keep inventory records on 250,000 items. Each item contains data of stock on hand at a number of depots, shipments made, goods received, back orders, stock-leveling action taken, various condition codes signifying such things as whether an item is new, or used, earmarked for overseas shipment, etc., as

---

W. K. HALSTEAD, J. W. LEAS, J. N. MARSHALL, and E. E. MINETT are with the Radio Corporation of America, Camden, N. J.

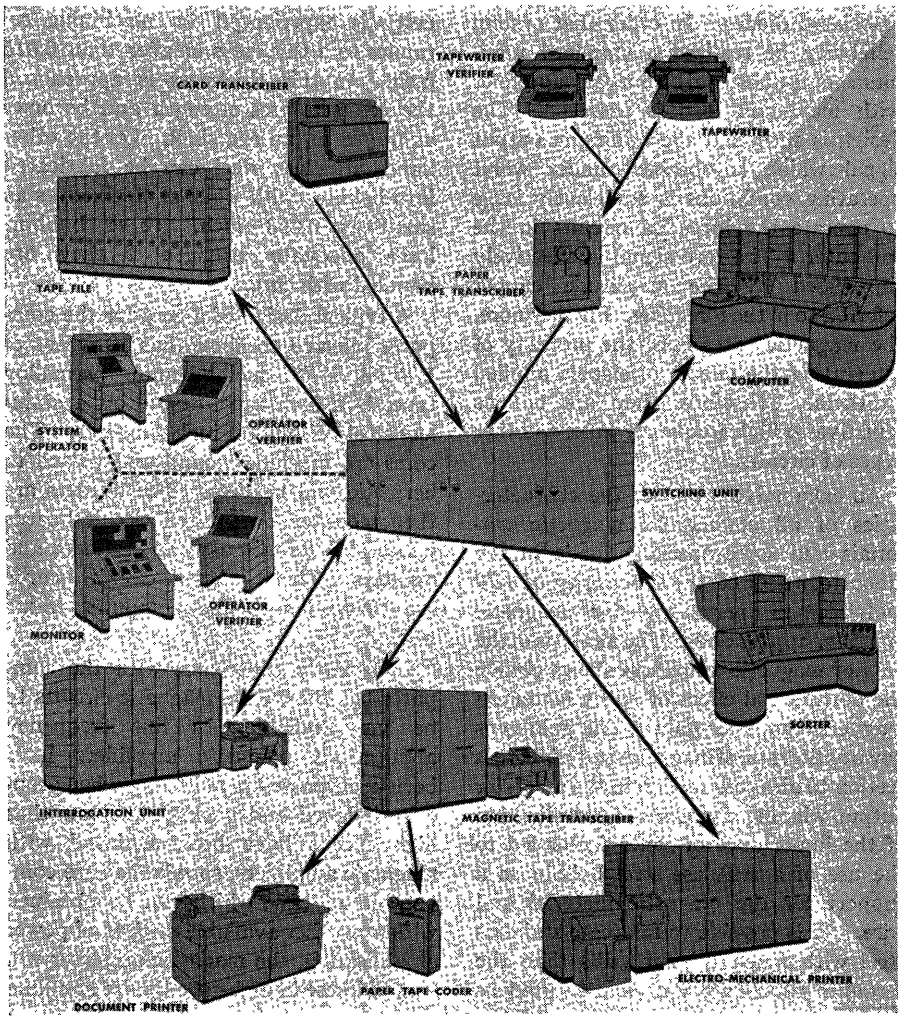


Fig. 1. RCA BIZMAC system

well as the stock number, noun description, and total national inventory. The average number of characters used to describe a stock item is 300, but in some cases runs as high as 1,500. Each day up to 65,000 transactions must be handled, the inventory corrected, the reordering and stock leveling action taken reported, action on special cases indicated, and a daily transaction record printed out. In fact up to 254,000 lines of printing must be put out each day.

A business machine system which employs a serial file is particularly well adapted to cyclic operation. Therefore, routine operations tend to be handled in a cyclic manner. Yet in any business data-processing task there are many times when data must be examined singly without regard to the main procedure which may be in process. In data-processing systems where reference data are contained in some sort of readable form, handling of the look-ups can be accomplished manually with relatively little cost. With the necessity to record data in an invisible form on magnetic tape in order to process them rapidly, the problem of handling look-ups must be faced. Obviously, duplicate files in readable form can be maintained. The cost of keeping such files up to date can soon outweigh their value. If, on the other hand, it is possible to interrogate a magnetic-tape file, the look-ups which must be handled quickly and yet specifically can be accomplished. Without such means, look-ups can only be handled in a batch after accumulation, which, of course, delays actions considerably. To cover the need for rapid random look-up, another special-purpose machine called the interrogation unit was designed. In the Ordnance application this unit will perform some 100 demand look-ups in an 8-hour day.

Now how these new machines and concepts fit into the over-all data-processing system will be examined. This can perhaps be done best by tracing the flow of data as it is processed through the RCA BIZMAC system.

RCA BIZMAC Equipment

In looking at the RCA BIZMAC input equipment, there are two basic types: manual and automatic. Manual input is through what is termed a tapewriter, which is a keyboard device producing punched paper tape and hard copy. A similar machine can be used to verify the input of the first one. This is called a tapewriter-verifier. The operation is similar to key verification of punched cards. Not only does the punched paper-tape output from the tapewriter provide a ready means of verification, but it provides an economic means of transcription from readable data to magnetic tape. This is accomplished through the unit known as the paper-tape transcriber. It can actually transcribe the normal output of about 35 tapewriters. The paper-tape transcriber reads the code punched in the paper tape and directly records it on magnetic tape as the paper tape is punched in the RCA BIZMAC machine language. It transcribes data at 200 characters per second. To save key stroke operations, the intermessage space on the magnetic tape is introduced automatically in the paper-tape transcriber rather than having this put on the paper tape by the tapewriter operator.

Automatic input is handled through the unit called the card transcriber. It reads punched cards at the rate of 400 per minute, and has the ability not only to split columns and add more data prior to recording on magnetic tape, but also permits rearrangements of data before recording on magnetic tape. While this rearrangement can be accomplished by a computer pass, it is less costly to do this at the card transcriber by means of plugboard connections.

The data are recorded in machine language on plastic-based magnetic tape. Each reel of tape is mounted at a tape station, and a pool of such equipment is called the tapefile. This is the reference and working file for the system.

The data are recorded 125 characters to the inch on a 5/8-inch-wide tape, and the tape is run at 80 inches per second giving the rate of data transfer of 10,000 characters per second. It may be of interest to point out that all data are recorded twice on the tape. Read-back of either recording will give satisfactory operation. This is accomplished by having 14 channels across the 5/8 inch width of the tape, thereby recording each of the seven channels twice. No more electronics is required to accomplish this, as the seven pairs of head coils are joined immediately outside the magnetic head. The dupli-

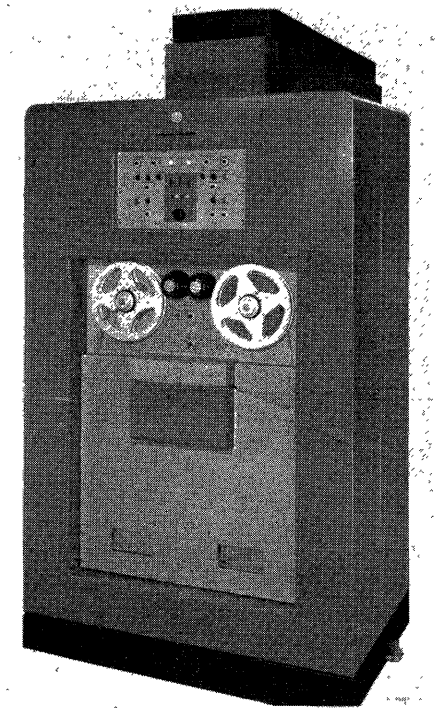


Fig. 2. Paper tape transcriber

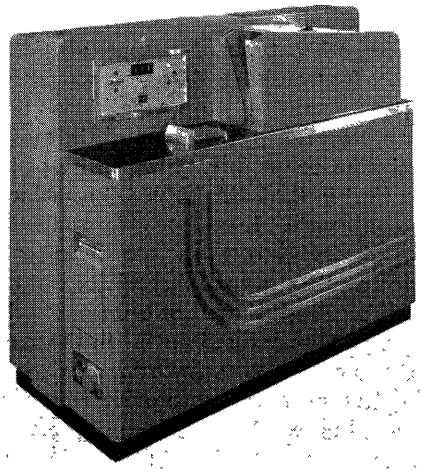
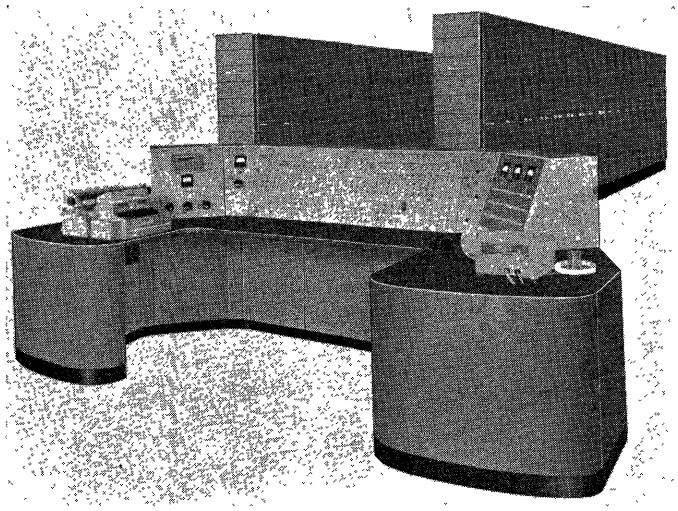


Fig. 3. Card transcriber

Fig. 4. Computer



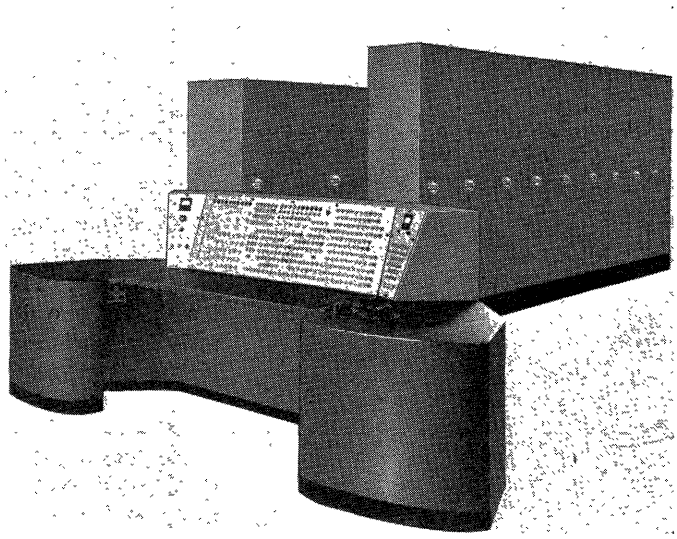


Fig. 5. Sorter

cate recording is displaced half the width of the tape and 1/8 inch along the length of the tape. As only one or the other of the dual recordings need be read to obtain satisfactory operation, dual recording is a very effective means of avoiding data loss due to any imperfection in the magnetic tape.

As mentioned before, all tapes requiring frequent access are mounted on tape stations which can be electrically connected to other equipments. These trunk connections are made remotely through a switching unit containing telephone-type relays.

For reasons of flexibility and ability to alter schedules rapidly, the system central is manually controlled. With this flexibility comes the necessity for accuracy control. Consequently each setup of a trunk line connection, each instruction to a particular machine, is either machine-checked or independently operator-checked. The system central concept and design has been human engineered for more efficient and accurate operation.

The RCA BIZMAC sorter will not be described in detail in this paper, and hence a few words of explanation may be appropriate. While it is called a sorter, actually its major use is in extracting by list or class, and merging. It actually is a wired program machine which has some 13 separate modes of operation including a simultaneous merge, substitute, extract, and delete mode. This is the method used to update the reference file and to extract from it the active accounts for the next processing cycle. The sorting method used builds up progressively longer strings of messages arranged in ordered sequence. It takes advantage of any inherent order in the list of messages to be sorted.

In practice, business data often contain much ordered data. Depending on its modes of operation, the sorter may have anywhere from two to six tapes connected to it. Once the appropriate tapes are connected to the sorter for sorting, the successive passes through the data on the tapes are accomplished automatically by the sorter without system central intervention. It is only when the sorting is completed and the data are recorded on the output tape that the system central switches that one tape to the next operation. It is possible on the sorter to sort, merge, or extract on a criterion consisting of up to 32 characters in the first 50 characters of the message. This criterion may be a composite one consisting of more than one data item. To give an idea of speed, assuming random order of messages, the sorter can sort 16,000 100-character messages in approximately 90 minutes. It can merge 150 such messages into 16,000 in $4^1/_4$ minutes. Similarly, it can extract the same number of messages in $4^1/_4$ minutes.

As can be seen, the sorter can relieve

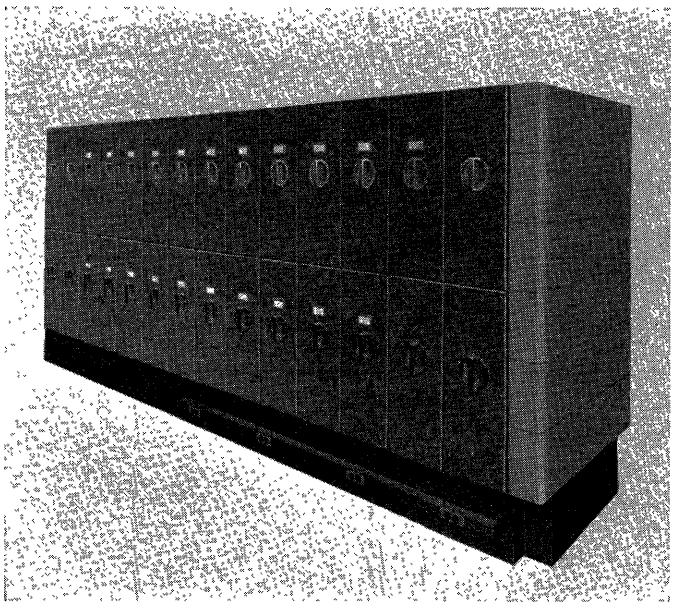


Fig. 6. Tapefile

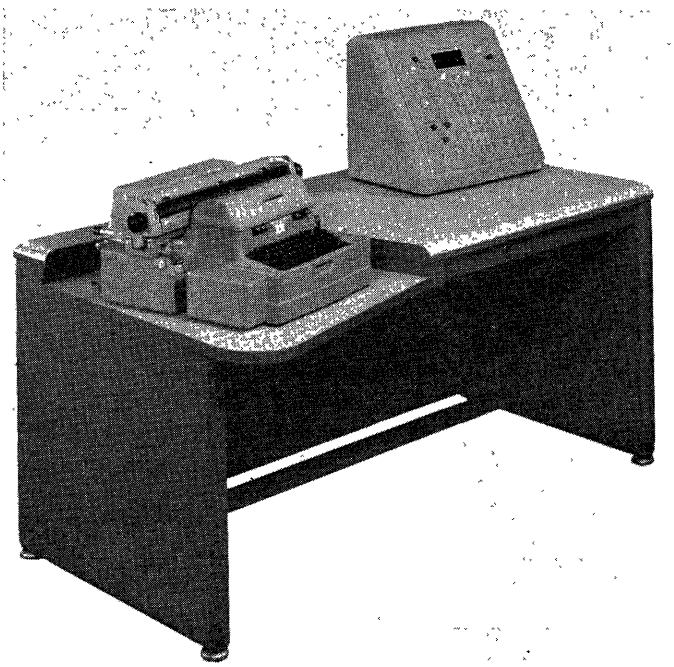


Fig. 8. Interrogation unit

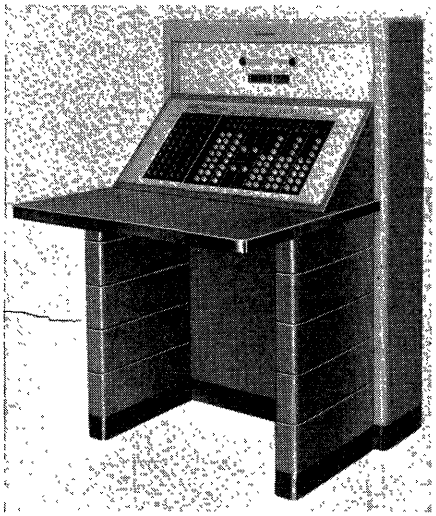


Fig. 7. Operator-verifier console

the computer of many of the data-shuffling processes and, of course, perform these simultaneously with computer operations. The saving in elapsed time in the OTAC application is considerable. There are three sorters in the Ordnance installation to handle the required amount of data shuffling.

The RCA BIZMAC computer is shown on the chart (Fig. 4) and will be discussed later in greater detail. One of the characteristics to point out at this time is that up to five tapes may be connected to the input of the computer, and up to ten tapes to the output of the computer. When so connected through the system central, the use of these tapes is entirely under the control of the computer, and no system central intervention is required until the computer is finished with the use of a tape. It is a binary-coded alphanumeric machine, handling data serially by character with a 3-address program. The rate of computation is such that two decimal digits may be added in 40 microseconds. It has both a high-speed magnetic-core memory and auxiliary internal storage, primarily for program instructions.

The interrogation unit can have any tape station within the tapefile connected to it, and any particular message recorded thereon can be extracted. To accomplish this, an operator connects the appropriate tapefile from the control console of the interrogation unit by referring to an index. Incidentally, the interrogation unit can prepare its own tape station index. Once the tape station is selected, the operator then types in the identification of the particular data desired and presses the "interrogate" button. The tape is run and comparisons made until equality is obtained, at which time the data are read from the magnetic tape, and then are used to actuate the same tapewriter on which the operator typed the identification number. Provision is also made for printing the interrogation output at remote points. The over-all time for a complete interrogation averages approximately 4 minutes.

Looking at the output side of the RCA BIZMAC system, two types of outputs are indicated. The first, the electromechanical printer, is a multiple-wheel rotating-shaft printer which will print at the rate of 600 lines per minute up to 120 characters per line. Editing for horizontal and vertical tabulation is accomplished at the printer and up to three carbon copies can be obtained.

The other type of output provides means for translating data from magnetic tape into punched paper tape. The unit for accomplishing this is called the magnetic-tape transcriber. The paper tape is punched in the RCA BIZMAC machine-language code. It can be fed into an output device known as the document printer. Besides providing suitably tabulated hard copy output, it can print both lower and upper case letters. In the particular application at the Ordnance Corps, this feature is used to provide copy for typesetting catalogues where upper and lower case letters are desired. In addition there is a 7-hole to 5-hole paper-tape converter known as the paper tape coder, which can transform the data into code suitable for teletype winter transmission or for input to tape-to-card converters.

A final point is the important system consideration of accuracy control. Its implementation must be related to the accuracy requirements of business data processing. These, we believe, do not call for either the extreme of complete duplication of hardware or the opposite extreme of complete dependence on programmed or procedural checks. Where uniform procedures of accuracy control can be applied repetitively, automatic hardware checks should be employed, consistent with the cost of such hardware. Such hardware can usually save heavy programming and processing costs. On the other hand, where flexibility of checking is desired to suit varying conditions,

programming and processing checks are usually best. The philosophy in the system has been reflected in various ways, which will be covered in the four following papers.

About 1,300,000 messages or 230,000,000 characters of reference information must be processed daily. An additional 7,500,000 characters are handled to process the day's transactions. Keeping up-to-date information on all open orders: vendor's name, material ordered, shipping dates, destinations, performance against purchase orders, etc., calls for record keeping on 120,000 ordered messages.

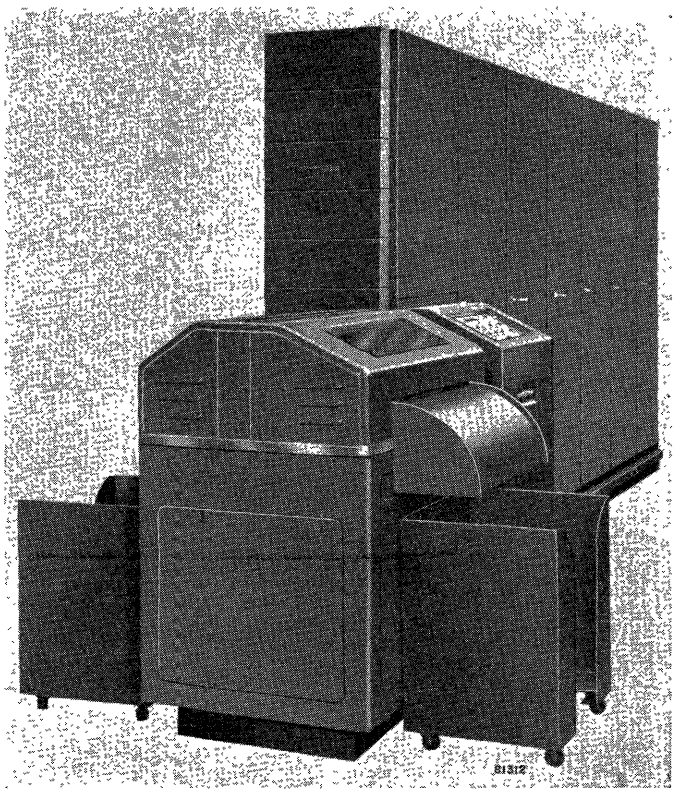


Fig. 9. Electromechanical printer

System Innovations

To accomplish these mass data-handling tasks, certain new concepts were indicated.

It was realized that there were several special characteristics in business data handling for which special system and equipment provision were then made.

First of all it was realized that business data themselves were highly variable in content as well as length of data items. This can be an asset if handled in the right manner. If one makes provision in the recording of the data for actual length of data, rather than maximum possible length, very significant savings are realized in length of magnetic tape used and processing time. This approach was pursued and the entire RCA BIZMAC system was planned to handle variable item and message lengths of data. (An item is a single piece of data such as a stock number of quantity. A message is all the information relating to a particular stock number or transaction.) In the Ordnance application, the ratio of maximum to average length of inventory messages is 5 to 1. Since a basic limitation on speed of data processing is the rate at which useful data can be read from the file, the advantage of having data recorded in the compressed form indicated is obvious. With the same instantaneous rate of data flow, a fixed word organization of data would require approximately five times as long to process as variable-data-length organization. Similarly, the data compression has made it economically feasible, for the first time, to leave tapes mounted at their tape stations and to provide enough tape stations so that all tapes which are to be used regularly can be mounted simultaneously. The design of the tape station has been considerably simplified and reduced in cost. For instance, one amplifier control unit can service up to five tape stations on a time-shared basis, and one power supply can handle seven amplifier control units.

Leaving tapes at tape stations requires a form of switching to connect a tape station first to one processing machine and then to another. This permits the imposition of centralized control over the entire system operation. The switching unit and control consoles are called the system "central." In basic concept this is similar to a telephone exchange. Trunk connections can be made between any tape station and any operating equipment. To handle a large flow of data, such as in the Ordnance application, this was found to be the only method to accomplish the task in the available time. There are over 1,000 trunk connections to be made daily. In addition, machine setups as required are made by the system central.

Another important characteristic of business data processing is the relatively large amount of rearrangement of data that takes place. When data are recorded serially on magnetic tape, it is necessary to arrange these data in different ways for various parts of the job. For instance, inventory messages need to be extracted from a reference file according to a list of stock numbers, or according to a classification such as new or used. The RCA BIZMAC system has termed these operations "extract by list" and "extract by class." Two message sequences need to be merged together, such as updated inventory messages being merged into a master file, either together with the previous data or replacing them. These operations are called "merge" and "merge-substitute." Incoming transaction messages may require sequencing in an order different from the order in which they are transcribed into the system. This is known as sorting. These various operations must not only be performed rapidly, but they must be performed using varying criteria, i.e., sort on stock number, then on depot location. In the particular inventory control job of the Ordnance Corps, this kind of data shuffling operation is approximately three times the amount of computational operations. While it is true that the various functions of sorting, extracting, and merging can be accomplished in most computers, these operations are not strictly computing.

In order to handle these operations more efficiently, a special purpose machine has been designed which is called a sorter.

System Flexibility

The innovations in the RCA BIZMAC system bring to the large-scale data-processing field a flexibility of operation which has not heretofore been available. This ranges all the way from the detailed but important structure of information as it is recorded on tape, through a highly flexible programming system for the computer, to the manner in which the system as a whole is operated from a central location. These subjects are described in detail in following papers (pp. 124–42).

Functional Organization of Data in the RCA BIZMAC System

A. D. BEARD W. K. HALSTEAD J. F. PAGE

BASIC to the data-handling pattern in the RCA (Radio Corporation of America) BIZMAC system is the highly flexible arrangement of data on magnetic tape. This arrangement was established after careful study of actual business data and processes to be handled by the system. It was found that business data have two major characteristics:

(a). Various parts of the data composing a record differed from each other in their maximum length.

(b). Each part of the data might vary from time to time from nothing up to the maximum length called for by the characteristics of the data.

The usefulness of a business machine system in a given application ultimately can be expressed in terms of cost of performing the job. The costs of entering information into the system, storing it, and retrieving the information represent important items in the over-all cost. Whatever can be done to reduce the number of manual key strokes at the input, to reduce the amount of magnetic tape used to store a given volume of information, and to reduce the time of passing through this information in processing will produce large dividends in holding down the cost of a job.

Arrangement of Data on Tape

Data in the RCA BIZMAC system are represented by binary-coded characters consisting of 7 bits each, 6 information bits plus a parity bit. Inclusion of the parity bit is one part of the RCA BIZMAC accuracy control system. It is used as a means of insuring accuracy in the transfer of data. It is chosen to be a one or a zero so as to make an even number of binary ones in every character.

Characters include the decimal digits, the letters of the alphabet, punctuation and certain other special marks, and a group of control symbols used in organizing data in the system. The code assignments permit alphabetic, numeric or alphanumeric sorting.

Data to be processed by the system are inscribed through one of several input devices and are recorded on magnetic tape mounted at tape stations. The 7 bits of each character are recorded simultaneously onto 14 parallel channels on the tape. Each bit of the character is recorded on two separate channels providing an added increase in the reliability of the tape recording.

Groups of one or more characters having some particular significance, such as a numerical quantity, an alphabetic name, a stock number, etc., are called items. An item is always preceded by a control symbol called an item separator.

All characters are recorded on tape, and read from tape serially so that the characters making up an item follow one another in sequence from the most significant to the least significant. Character spacing on magnetic tape is 125 characters per inch. The tape speed is 80 inches per second, giving a data rate of 10,000 characters per second.

One or more related items are called a message in the RCA BIZMAC system. A message is delineated by always having a "start message" symbol as the first character and an "end message" symbol as the last character. Each message usually consists of a number of discrete items of information, some of which may or may not be present each time. Each of these items can vary in character length from one to some sensible maximum. The maximum for one item may be different from the maximum of another. The use of item-separator symbols allows the use of variable item lengths, but does not preclude the use of fixed item lengths.

It is important to distinguish between two separate concepts included in the term "variable item length." First, this implies that different items of information may have different maximum lengths. This may be referred to as the nonstandard maximum item length. Each item will have some maximum possible length. Secondly, variable item length implies that each item of information may have a varying number of characters from zero (not present) up to this assigned maximum length in any particular message.

The items of a message follow one another in sequence, each being preceded by an item-separator symbol. In every message of a given type the n^{th} item always has the identical connotation, e.g., date. Therefore, a count of the item-separator symbols, starting with the first which follows immediately after the start-message symbol, permits identification of the meaning of any item.

If any particular item of information should not occur in certain messages, its item-separator symbol only is recorded on the tape in its proper sequence, except that it too may be omitted if the items that follow it are missing also. In this case, the end-message symbol may follow immediately after the last active item. This avoids writing an unnecessary number of consecutive item-separator symbols at the end of a message. Thus, with such a system, a further gain is achieved by arranging the most frequently used items at the front of the message.

Fig. 1 illustrates the layout of a typical message on magnetic tape. The message consists of a start-message symbol, followed by an item-separator symbol, and the characters of the first item, the succeeding items (each preceded by an item-separator symbol) and an end-message symbol in that order.

It is frequently desired to read messages individually from magnetic tape and to record messages individually on tape, stopping for a period of time between messages. Therefore, sufficient blank tape must be provided between successive messages to allow stopping and starting the tape between messages without missing characters. The design of the tape station is such that 5/8 inch is adequate for this purpose.

Fig. 2 illustrates the difference between systems which use either the fixed word, the nonstandard maximum, or the variable word used in the BIZMAC system. In the fixed-word system each word must be equal in length to the maximum word expected to be processed. In the example, this is 12 digits. In the second example (nonstandard maximum) a saving of digits is realized by requiring each item or word of a message to be equal only to the maximum length expected for that particular word. In the example chosen this is seven digits. The variable word used in the BIZMAC system however, requires only the meaningful digits for any item.

Influence of Data Pattern on System Design

It should be clear from the preceding sections that the variable item and mes-

A. D. Beard, W. K. Halstead, and J. F. Page are with the Radio Corporation of America, Camden, N. J.

sage length concepts lead to extraordinary savings in tape space, and hence in tape-file processing time. In a large inventory-control operation, ratios of maximum to average message lengths have been encountered which have exceeded 5 to 1. This ratio, in fact, applies to the main inventory file composed of approximately 250,000 messages.

Obviously, these concepts of data pattern influence the design of the data-processing equipment. The special symbols which indicate start message, end message, and item separator are used as sentinals to convey to the processing equipment the meaning of the accompanying information, and to control equipment action in accord with previously established programs or machine settings. In this sense, the special symbols replace character counts so commonly used in fixed-word-length computing systems.

In the sorter, for instance, the selection of the sorting (or merging, or extracting) criterion is based on a recognition and counting of item-separator symbols, rather than characters, beginning with the item separator immediately following the start-message symbols. (Under certain circumstances, the criterion can be a part of an item, in which case it is necessary to count characters within the designated item to arrive at the start of the criterion.)

In the computer, the high-speed memory is laid out on the basis of the maximum number of items in the incoming, and outgoing, messages, and of the maximum assigned character count for each item. Since that much memory must be available anyhow to handle the maximum-maximum case, fixed memory assignments are made, since they simplify programming and speed up processing.

The computer "read-in" instruction has associated with it a series of addresses giving the starting (leftmost) positions in the memory for consecutive items on the input tape. In this manner, the compressed information on the tape can be spread out to permit fixed-address programming for the remainder of the computer processing job. Since the series of read-in addresses does not have to be monotonic, the read-in process can be used simultaneously for editorial rearrangements, preparing for subsequent printing, for example.

For the "write-out" operation, the reverse requirement exists, at least in those instances where a file or intermediate storage tape is created. In these instances, it was found most expeditious to precede the write-out instruction by a "compress" instruction. This shifts all meaningful information close together in one area of the high-speed memory, eliminating all unnecessary space characters and, if so programmed, also suppressing all zeros which may have built up to the left of the most significant character in some results of arithmetic operations.

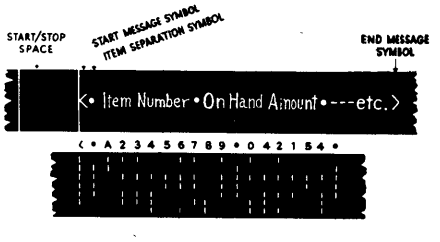


Fig. 1. Organization of data on magnetic tape

Another peculiarity of the compression of information on tape is that left-justified and right-justified numbers are not readily distinguished from each other. An example of left-justified numbers exists in the Dewey Decimal System of book cataloguing where places of equal significance are measured from the left end of the number. Alphabetic lists (telephone books) are also usually left justified. Right-justified numbers, typically, are dollars-and-cents amounts, or other integral numbers in columns which are to be added. Few, if any right-justified alphabetic items exist in practice.

In the computer, right justification becomes important in connection with arithmetic operations. All items are read left-justified from tape to high-speed memory, i.e. with the most significant digit immediately to the right of the item separator. Arithmetic operations, on the other hand, proceed from right to left, and the memory for the numerical items involved is addressed at the rightmost location, where the least significant digit would appear if the item were long enough to occupy all allocated memory positions. If the actual item is shorter, space characters appear in the remaining memory positions to the right. In carrying out an arithmetic operation, the computer skips over any such unused position, and then works with the numerically significant digits, automatically "lining up" the least significant digits of the two operands. Since the computer only works with the significant digits of a word, considerable savings in computer time are realized. This concept of

Item Quantity on Hand • 478
Maximum Length of Item • 7 Digits
Average Length of Item • 4 Digits

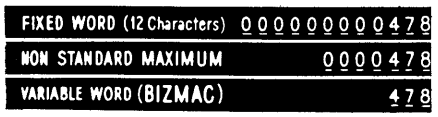


Fig. 2. Variable word data organization on tape

The use of variable word organization on tape:
- Minimizes tape storage requirements
- Minimizes the number of tape handling mechanisms required
- Minimizes the time required to pass information to and from magnetic tape

variable operation time in the computer is discussed in a later paper (Bensky *et al.*).

The foregoing examples show the extent to which the major equipments have been specially designed to accommodate the basic data organization. This was thought to be particularly important in connection with the computer.

In some of the peripheral devices, compromises have been struck in favor of equipment simplification. In the electromechanical printer, for instance, item-separator symbols are used to cause horizontal tabulating of information in a line, but right justification of an item can be obtained only by previously inserting on the tape the required space symbols to the left of the numeric digits. This is done in the computer. Similarly, in the card transcriber, some unnecessary space symbols, derived from blank card columns, are recorded on the magnetic tape.

Although the system is designed to take full advantage of variable item and message lengths, with the restrictions described above, this does not preclude the use of fixed lengths where convenient. The card transcriber for example, produces fixed item and message lengths, since data read from punched cards are arranged in fixed fields. After transcription, this information may be read directly into the computer memory and compressed as desired.

Summary

In summary, the variable item and message lengths concepts lead to a highly efficient tape-storage system.

Extremely significant savings in tape space and hence tape-file-processing time are achieved. The quantity of manual input operations is held to a minimum.

The RCA BIZMAC System Central

J. L. OWINGS

THE RCA BIZMAC system is a product line of fully-integrated electronic data-processing machines, which has been designed to meet the large volume requirements of commercial applications. A typical large installation may have about 200 separate RCA BIZMAC units. To operate such a large assemblage of machines, a means must be provided for integrating them into an operating unity and for controlling their operation at all times. For the RCA BIZMAC system the "system central" is the integrating element and the focus of all operating activity.

The design of a system central was based on the concept of centralized control of all elements in the system. This concept is well known in other fields for obtaining optimum performance from assemblages of equipments and operators, and is well suited to the field of electronic data-processing machines. A review of the design of the system central that was provided for the RCA BIZMAC system which was supplied to the Ordnance Tank and Automotive Command (OTAC) will show how it is made and how it is used to control the operation of the system.

The RCA BIZMAC Machines

Before looking into the data-processing task itself, a knowledge of the function of the machines in the system and what system central has to do to operate them is necessary. Fig. 1 shows the RCA BIZMAC equipments. They consist of input machines for transcribing data into the system, a magnetic-tape storage file for both long-term and short-term storage of data in the system, processing machines for computing on and rearranging data, output machines for transcribing data from the system after processing, and auxiliary machines which fulfill special system functions.

System Central Functions

To transcribe data into the system it is necessary to select one of the input machines and one of the magnetic-tape-storage units, connect them together, set up the input machine, and operate the two machines. To reduce both operating and equipment costs, the tape units have been designed for automatic operation under the control of the data-processing machine to which they are connected. This is an important system feature, because it means that there need be no occasion for an operator to attend any one of the tape stations during periods of normal operation. This considerably reduces the cost of operating an RCA BIZMAC system.

To process data on one of the major machines in the system, it is necessary to select the proper kind of machine for the job and the particular tape stations which contain either the data to be processed or the reels of magnetic tape which will be used to carry the processed data to the next operation, connect the tape stations to the correct input and output trunks of the data-processing machine, instruct the machine, and proceed with the operation. During the operation it may be necessary to supply more than one reel of data (on magnetic tape) to one or several information trunks of the machine. When one tape is exhausted, the major machine automatically initiates rewind of the tape and requests the next tape. It is then necessary to select the proper tape station from the storage file and the machine trunk which needs the tape, connect the two, and continue with the operation.

To transcribe data from the system it is necessary to select an output machine and the tape station which contains the data to be transcribed, connect the two, set up the output machine, and transcribe the data.

These functions, i.e., selecting, connecting, instructing, and operating, are fundamental in the operation of an RCA BIZMAC system and, as such, must be provided in the design of system central. Another essential function is over-all control of the operation of the system. The principal areas for control which system central is responsible for are: the orderly execution of data-processing operations, the location of all data which are on magnetic tape, the setup and operation of all machines in the system, and the work which operators do. With these functions in mind, a typical example of a particular job is given, and the resulting design of a system central is presented.

The Task for System Central

The task for the RCA BIZMAC system encompassed all the bookkeeping for the OTAC depot located at Detroit, Mich. This consisted of stock control, supply control, catalogue changes, cross-reference filing, etc., on some 250,000 catalogue items. The work load for the system involved processing approximately 80,000 transactions per day. This required about 100 machine operations and 1,000 individual tape connections. To do this amount of work each day required a highly integrated and controlled system.

First, the task must be stated. This is done without reference to any form of implementation. What must be obtained from the system, what available data there are to work with, and what must be done with these data to get the desired results is determined. The next step is to determine how to do this with the RCA BIZMAC machines. This leads to a chart of the form shown in Fig. 2. Here the flow of data is shown in a general way and only machine types are referred to. However, before the job can be given to the system it must be reduced to much greater detail. In Fig. 3, a block diagram of the task in the form needed by system central is shown. Each block represents a single machine operation, and the lines which interconnect the blocks show the flow of data from trunk to trunk of each machine operation. This diagram is prepared by a planning group and shows all the data-processing operations which are to be done on one batch of input data. In addition to this chart, the planning group supplies system central with detailed instructions for all machine operations. These consist of data loads on the input and output of each machine operation, machine setup instructions for all operations, the approximate operating times for each machine operation, and the priorities and prerequisites among all operations. This information is supplied to the system central in the form of charts and decks of operation cards. An example of an operation card for the sorter is shown in Fig. 4. The operation number, machine type, and priority number are located on the card tab. The setup instructions for initiating the operation are located in the solid color area of the card. The connections needed to supply additional data to the machine during an operation are shown in the area titled "subruns." This card, when it has been filled out completely, contains all the information needed by the system operators to set up the system and do the

J. L. OWINGS is with the Radio Corporation of America, Camden, N. J.

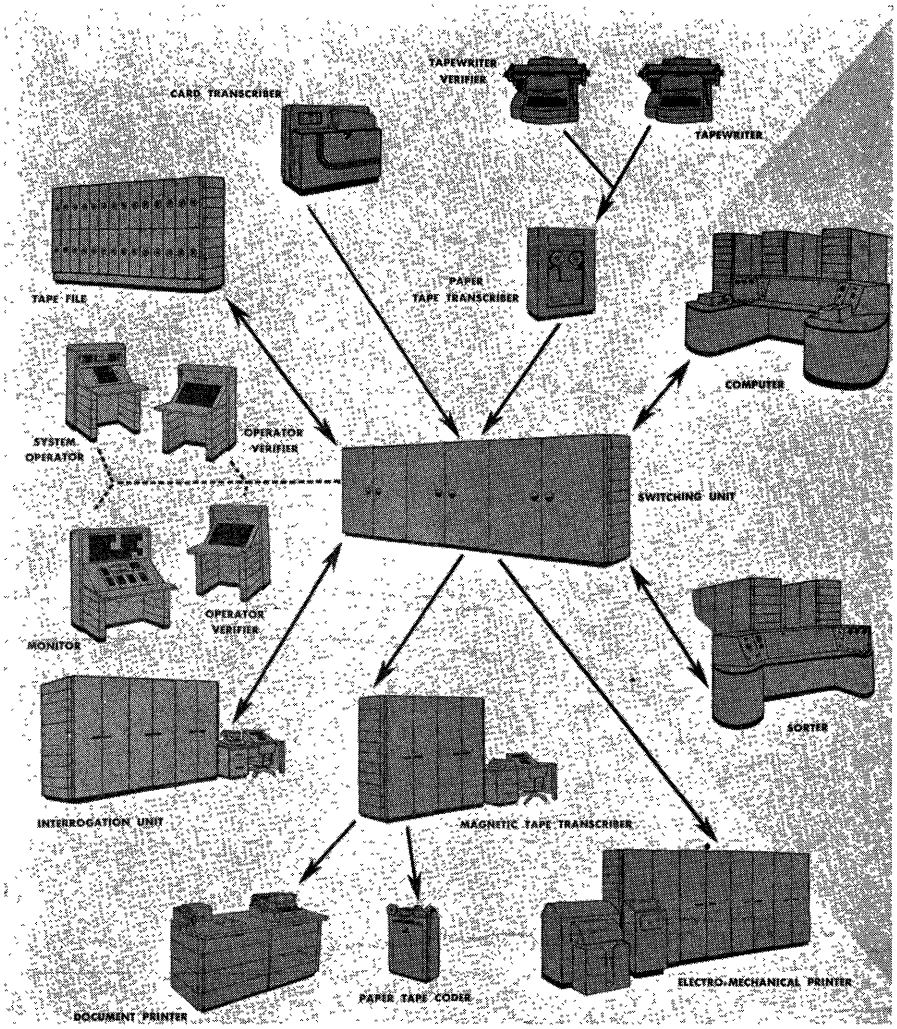


Fig. 1. The RCA BIZMAC electronic accounting system

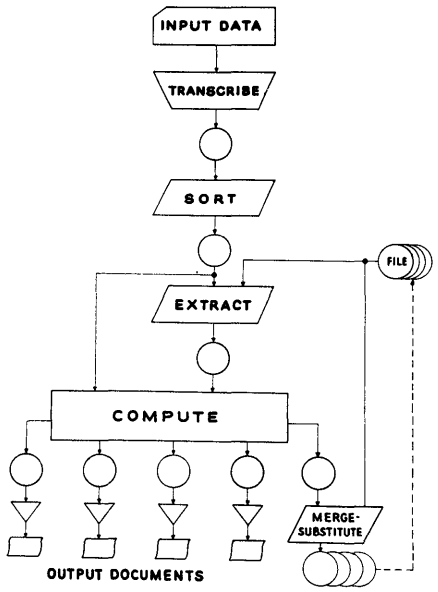


Fig. 2. RCA BIZMAC OTAC acceptance test

operation. When this card reaches system central, it contains all the information needed for operation except the numbers of the tape stations which will be connected to the machine trunks during the operation.

As mentioned previously, data are processed through the system in batches. Each batch may be released to the system as soon as the detailed plans for operation have been completed. Therefore, there may be more than one batch of data in process on the equipments at one time. Fig. 5 shows how, in a typical operating situation, cycles may overlap during normal system operation. Cycles are introduced into the system at irregular times. The time needed to complete a cycle is only roughly known at the outset and may be radically changed by unforeseen or unanticipated operating conditions. This means that detailed scheduling of the use of data-processing machines and tape stations is not practical or desirable. The most desirable situation is to use the machines and tape stations when they are available, and control the order in which things are done instead of the time at which they are done. This then, is the situation which system central was designed to accomplish.

The System Central

Fig. 6 is a diagram of the system central which was designed for this system. This diagram shows system operation under control of system central. The lightning strokes show the system operator's control over both the operating team and the system machines. The broad arrows show the flow of operating instructions from the operating team into the system, and the feedback from the system to the operators. In addition to these operators there is another group of operators located at the various input and output machines which require manual setup and operation. The work of these operators is strictly controlled and co-ordinated with the activity in the system central control room through procedures and the use of the operation cards.

To assist the operators in performing the functions of selection, connection, instruction, and operation, system central contains a switching unit, an operation control unit, and a set of consoles and control panels for the operators. The switching unit consists of electromagnetic relays of a special design to provide reliable operation for long periods of time. Fig. 7 is an example of the kind of relay which was the basic element used in the switching unit to connect tape stations to machine trunks for operation. The operation control unit supplies the checking, lockout, transfer, and other functions needed to set up elements of the system for operation, guard the system against improper operating routines and operator mistakes, and return the monitoring information needed by the system operators. The consoles, as shown in Figs. 8 through 10, provide the work space and control panels needed by the system central operators to perform their duties. These consoles were human-engineered to make operation simple and straightforward for the operators, and to be comfortable to work at for long periods of time. The operating functions of direction, setup, and monitoring were separated and provided on separate consoles on the basis of the amount of operating load which was required for each of these functions in this system.

Human Engineering

The system central provisions for selecting, connecting, instructing, and operating machines have been reviewed. Now the provisions for controlling the proces-

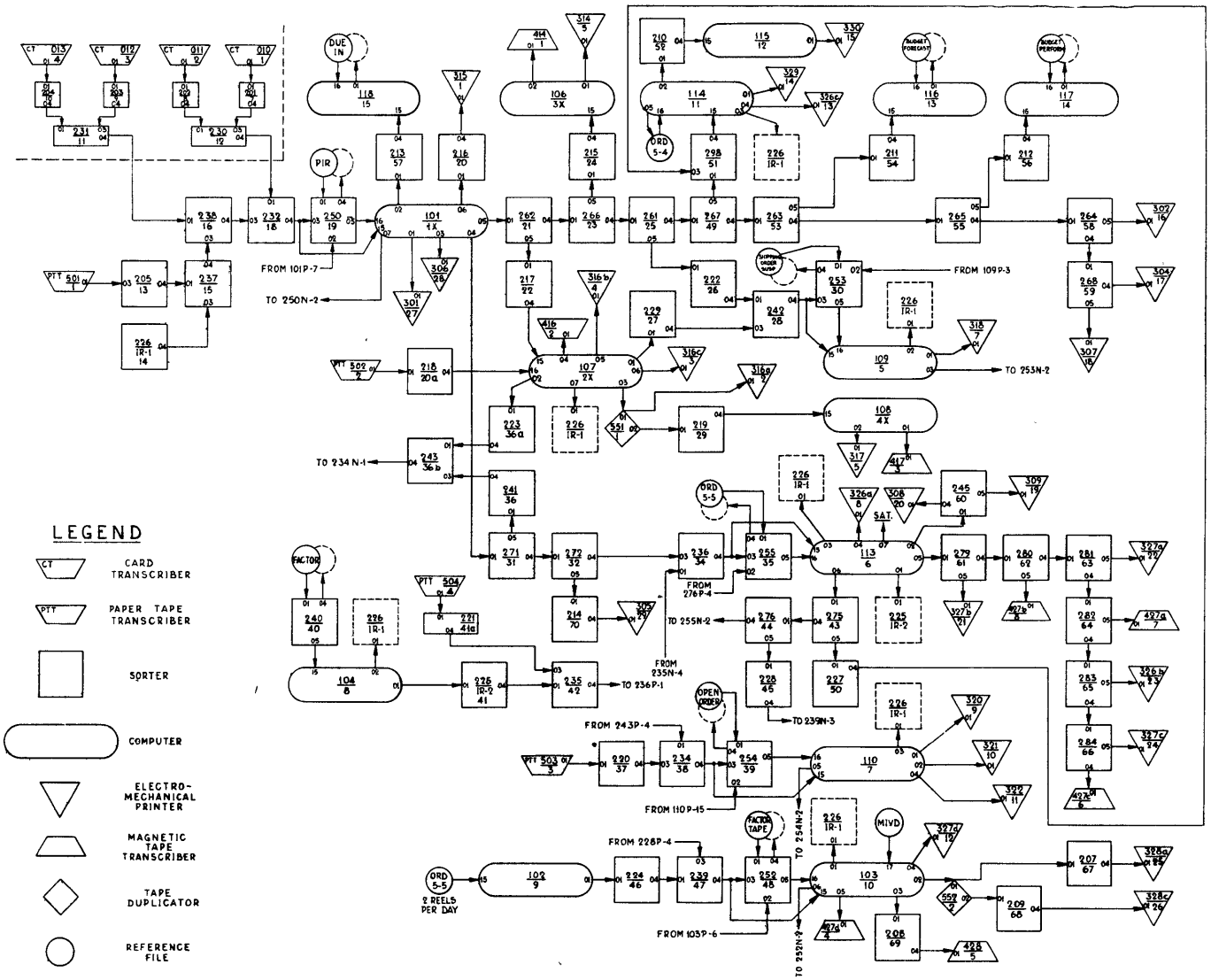


Fig. 3. Consolidated flow chart—one cycle

sing of data and the operators who do the work should be reviewed. This area of system central design leaned heavily upon know-how in the field of human engineering, because it was very important that the system be easy to operate and at the same time protected from mistakes which the operators might make. It is easy to see that a wrong connection, or a wrong machine setup, or an operation done out of order, might cause destruction of valuable data and great loss in time. To avoid these things, tight control was provided for all operator activity. Fig. 11 shows the flow of operating instructions among the system operators, and the checks used to guard the system against operator mistakes. This is a closed-loop type of operation, and is based upon a scheduling technique which was developed specifically for this job. It permits the use of both data-processing machines and tape stations on essentially an availability basis, yet provides tight control over the flow of data through the system. It also permits the processing of several batches of data through the system at the same time.

Schedulers

The schedulers receive, from the planning group, sets of detailed instructions for processing batches of data, and, from the operating group, both status indications on the progress of work already in process on the machines, and indications of tape stations which were available for handling new data. With this information, they complete the preparation of operation cards for all the operators. To avoid the consequences of human error in doing this work, two schedulers are used. They work independently at filling out identical decks of operation cards and the correctness of their work is checked by the interim scheduler, who inspects completed cards for identity and rejects cards containing errors. This procedure, called verification through independent duplication, is a powerful tool for checking the accuracy of operator performance. It enables the operators to work quickly at simple tasks and is quite effective for spotting errors made by either of the two people who do identical work.

Interim Scheduler

The interim scheduler controls the distribution of operation cards to all operators. He has to make sure that the system is ready for each operation before he releases the cards. To do this he receives operation cards for completed operations from the operators and checks them off against a list of prerequisite operations. Each operation card has a list of prerequisite operations which must be completed before it can be released. This list is prepared by the planning group and

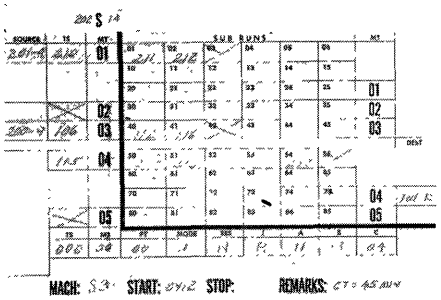


Fig. 4. Sorter operation card

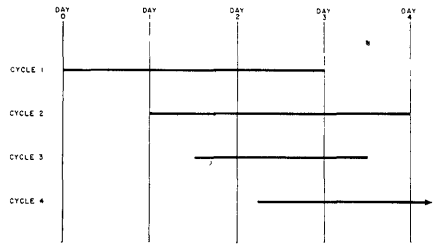


Fig. 5. Cycle overlap chart

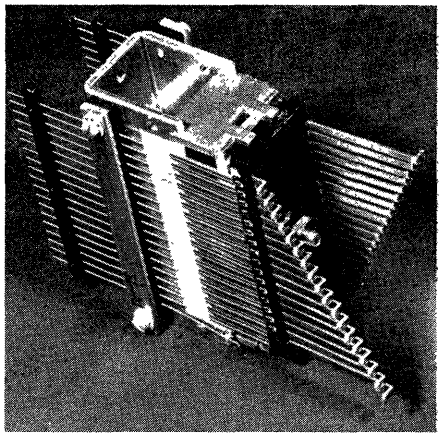


Fig. 7. Wire spring relay

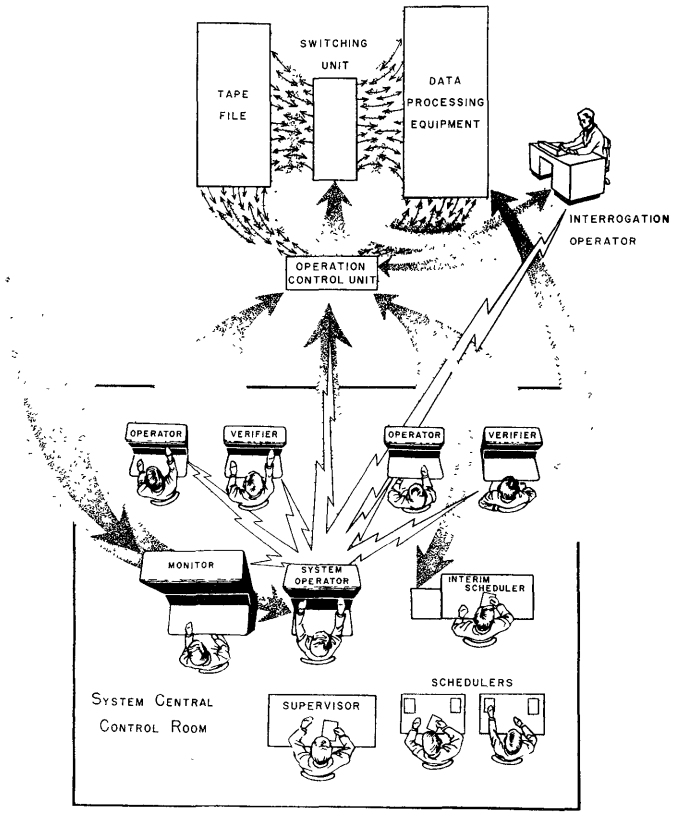


Fig. 6. System central diagram

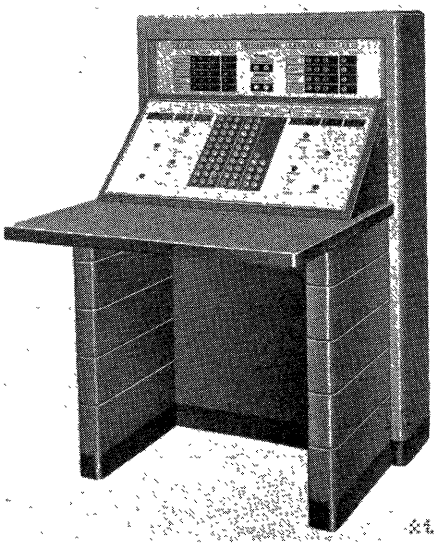


Fig. 8. System operator console

forwarded to the control room as part of the operating instructions. The purpose of the list is to provide a routine procedure for controlling the order in which operation cards are released to the operators, and thereby control the order of processing data through the system. When the cards are delivered to the system operators, the system is ready to handle the data as soon as machines are free. The work described thus far requires judgment and decision-making by the schedulers. In this system also, there is a considerable time lag between the release of operation cards and the actual setup and operation of the machines. This time lag is a very important factor in allowing enough time for judgment in filling out the cards, checking them, and determining when to release each card.

System Operators

The actual setup and operation of machines is done by the system operators. These people consist of a system operator who selects machines for operation and directs all operation, a monitor who keeps track of the progress of operations in the system as well as the operating status of machines, two teams of operator-verifiers who set up machines and connect tape stations to machines for operation, and a group of input, output, and auxiliary machine operators who set up and operate the machines which are not directly operated from the control room. All the operators work from operation cards.

The monitor receives operation cards from the interim scheduler and checks to make sure that the prerequisites have been met for each operation before he releases the card to the system operator for action. This checking procedure, called double checking, is useful for finding errors in very simple tasks. It is not so effective as verification through independent duplication, but it is faster.

The system operator receives cards from the monitor, selects machines for operation, and releases the setup to an available operator-verifier team. The selected operator-verifier team refers to its operation cards, and then each one proceeds independently to make identical setups on different consoles. The system central operation-control unit checks the identity of the setups, rejects errors, and transfers correct setups to the machines in the system. This is another form of verification through independent duplication, only, this time, a machine is used to check the identity of the work of the operators. This not only checks the operator and verifier but also pro-

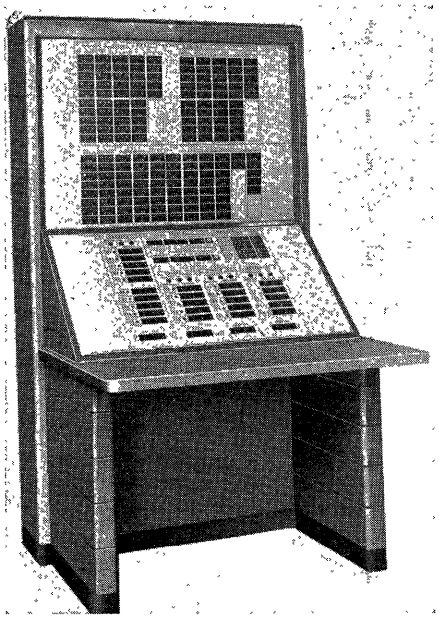


Fig. 9. Monitor console

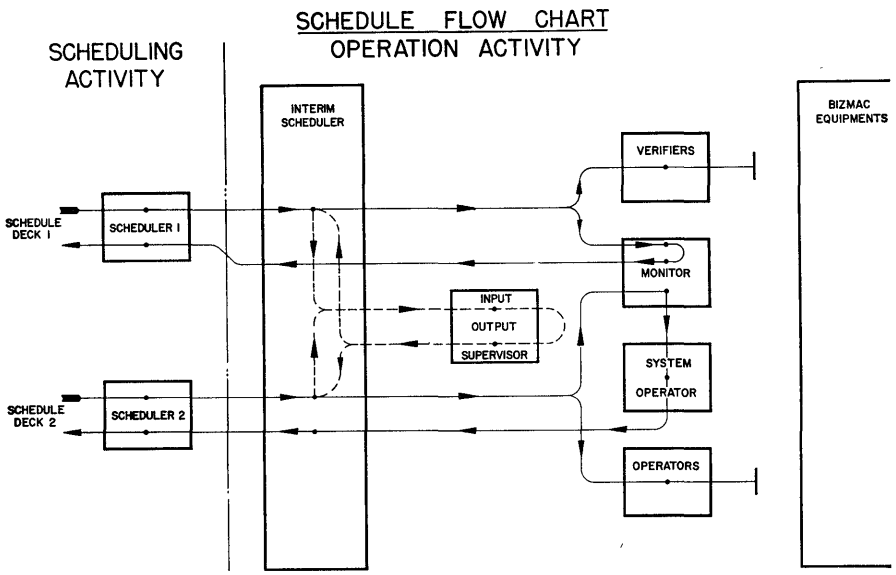


Fig. 11. Schedule flow chart

Fig. 10. Operator-verifier console

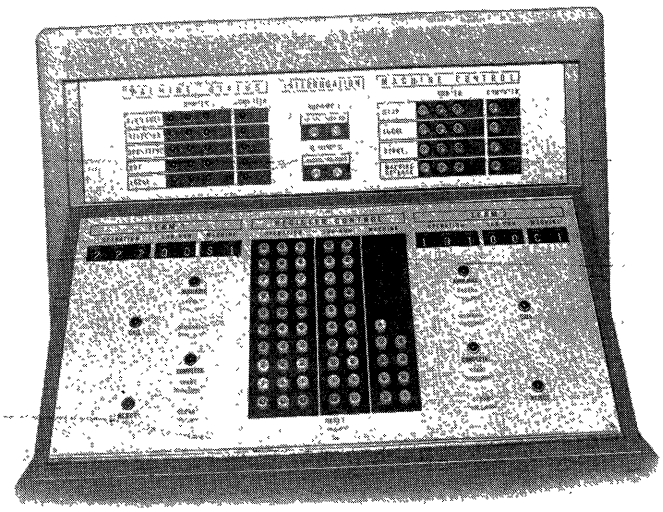


Fig. 12. System operator control panels

vides a second check on the identity of the work of the schedulers. When the setups are complete, the monitor checks his displays to make sure that the equipment functions properly, then signals the system operator that the operation is ready to proceed. The system operator starts the machine and then goes on to other work. At times of peak activity in the system, the system operator has both operator-verifier teams busy making setups for different operations at the same time.

The activities of the operators of the input, output, and auxiliary, machines are co-ordinated with the control room through the handling of the operation cards. The operation cards contain detailed instructions for machine setups, and each operator has to supply a written record of certain critical information with regard to how he did his work. This is checked by a floor supervisor before other dependent operations are permitted to go ahead.

All these operating procedures described were reduced to simple routine actions, and the procedures were made similar for all operations. This made them easy to learn, and easy to do. The operators at the consoles had very simple decisions to make, and could work fast to keep machines busy. The procedures used and equipment checks provided the necessary safeguards to protect the system against the consequences of mistakes by both the schedulers and the operators. Thus, human engineering provided the know-how for operating the equipments and controlling the processing of data through the system in a manner which was easy for operators to learn and do.

Consoles

The design of the consoles for the operators who work in the control room was an important factor in making the work easy for the operators and enabling them to work quickly and accurately. Fig. 12 is a close-up view of the system operator's console. On the left side of the vertical section are status indications for the most heavily used machines in the system. On the right side of the same panel are control buttons for the more important machine functions. In the center of the sloping panel is a register control for indicating the operation the system operator wants to initiate, and for selecting the machines he wishes to do the work. On the right and left sides of the

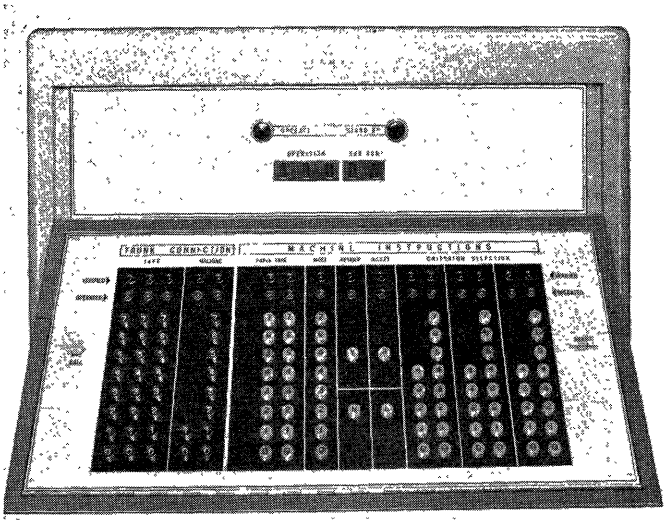


Fig. 13. Operator-verifier control panels

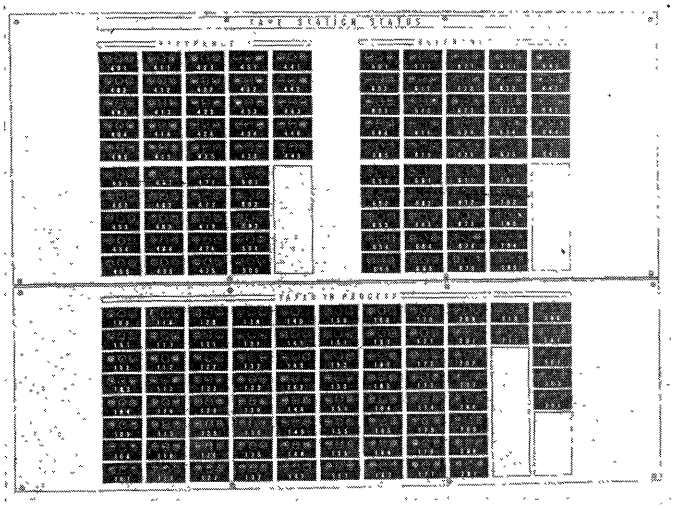


Fig. 14. Vertical section of monitor console

same panel are the team-control sections. The controls and indicators on these panels are arranged in the sequence for normal use, and follow the same design principles used in connection with the design of the control panels for the input and output machines.

Fig. 13 shows the control panels for the operator-verifier consoles. The vertical panel contains directions for action and registers for displaying the number of the operation card which is to be used for the machine setup. The sloping panel contains the controls used for connecting tape stations to machines and for setting up the computers and sorters. The controls are arranged for sequential operation from left to right. Color is used to separate the function of tape connection from machine instruction, and indications are provided to tell the operator whether or not his work is in agreement with his partners'. A call button is provided for signaling the system operator if there are operating difficulties. To make the work at these consoles even easier, the operation cards were designed so that the information was displayed on the cards in the same arrangement as the controls on the console. (See Fig. 4.)

Fig. 14 shows the vertical section of the monitor's console. This panel displays the operating status of each tape station in the system. Fig. 15 shows the sloping panel of the same console. This panel displays both the operating status of the sorters and computer, and the status of the data-processing operation which is in progress on each of these machines.

The human engineering of the system was carried beyond the system central control room to include all those areas where operators work with machines and

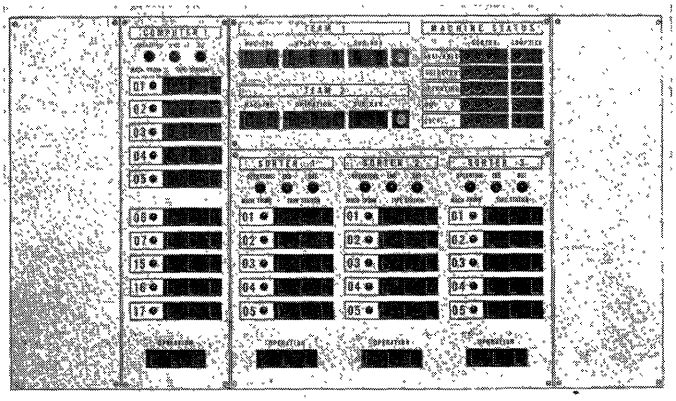


Fig. 15. Inclined section of monitor console

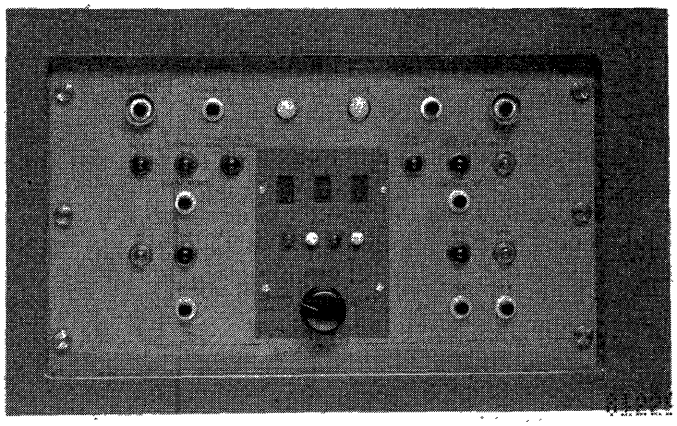


Fig. 16. Paper tape transcriber before human engineering

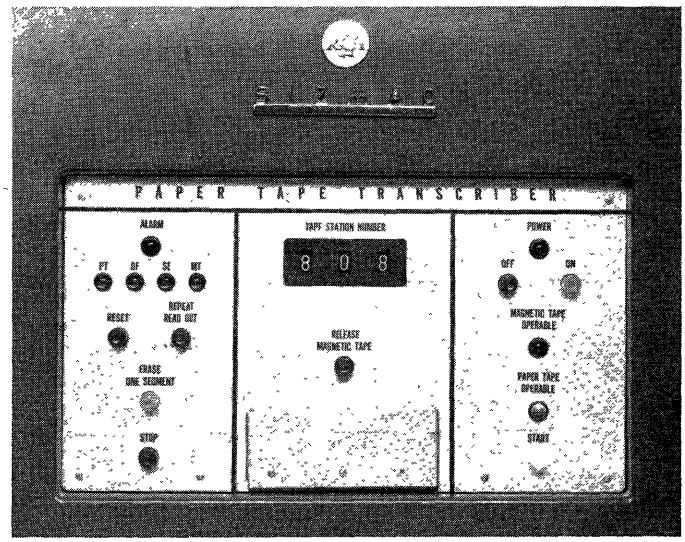


Fig. 17. Paper tape transcriber after human engineering

must be directed and co-ordinated from the system central. This included the design of control panels for operating the input, output, and auxiliary machines in the system. A brief review of the results of this effort will show how human-engineering principles lead to good equipment design.

The controls and indicators needed for the operation and maintenance of each machine were first divided into two classes, i.e., operation and maintenance. Where possible, those needed for operation were grouped on an operating panel, and those needed for maintenance on a maintenance panel.

The operating panel was made easily accessible to the operator and the maintenance panel accessible to the maintenance man. On the operating panels, controls were limited to a single design of push-button light which contains a light inside the button. This push-button light was specially designed for the RCA BIZMAC system. Indicators were limited to three colors: green, amber, and red. Throughout the system, green generally means: go, the situation is normal, or, proceed with the operation. Amber indicates normal operation—no operator action required. Red indicates trouble, an operating status which is not normal. The names of similar controls and indicators were made the same on all control panels. Directness of meaning, common usage, and avoiding the use of abbreviations were important considerations in choosing names. Names were chosen for controls to tell the operator what would happen when the control was activated. The indicator names tell the operator what has happened or what was happening. All control panels were designed with large readable letters and good contrast between letters and the background. Careful attention was paid in the selection of material to avoid glare.

The control panels for the input, output, and auxiliary machines were all similar in design and laid out so that an operator could easily learn to operate any machine. The panels were divided into three sections. The section on the operator's right contains all the controls and indicators which are used for normal operation and which are similar among all machines. The section on the operator's left contains all the controls and the indicators associated with abnormal or faulty operation. Wherever possible, the indicators for similar functions on the right and left side are opposite each other to make it easy to notice the cause of trouble. The center section of each panel contains the controls and indicators which are peculiar to the specific machine, or were placed in the center section to focus attention.

The basic pattern for arranging the controls and indicators follows the simple sequence: (1) an indicator to indicate the requirement for operator activity, (2) a control to effect the required action (push button), and (3) a feedback (light inside the push button) to indicate the successful accomplishment of the act. Where possible, the controls and indicators were arranged in sequences from top to bottom or left to right, in the order of normal use, to guide the operator and provide simple habit patterns. This procedure makes machine operation simple, straightforward, and easy to learn. When followed on all machines in the system, each operator can easily learn to operate any machine. Operators are interchangeable. New operators can be quickly trained.

An example of the control panel for the paper tape transcriber will illustrate some of the human-engineering design features which were mentioned. Two panels are shown for this machine to emphasize the value of styling, proper panel layout, and lettering. Fig. 16 shows the panel before human-engineering design. Fig. 17 shows the panel after the studies had been completed. The basic design shown in Fig. 17 was used on all input, output, and auxiliary machines in the system.

Summary

To summarize what has been said, return to Fig. 6. The system central is the focus of all operating activity for an RCA BIZMAC system. Its design includes providing the means for selecting machines for data-processing tasks, and tape stations for working with these machines, for connecting tape stations to machines, for setting up machines for operation, for operating the system, and for controlling all the things which contribute to coordinated and integrated system performance. Human engineering played a large part in making the system easy to operate and providing the know-how to protect the system from the effects of operator mistakes.

Through the use of the system central, the task of operating the system has been reduced to the point where a few operators located in a centralized control room can control the operation of about 200 machines, and rapidly process large quantities of data through complicated operating routines in a controlled and orderly fashion. This facility permits high utilization of data-processing equipment, provides effective control over the accuracy and efficiency of all aspects of data processing, makes it easy to accommodate changes in requirements, and does this all for a low investment in provision for operation.

Characteristics of the RCA BIZMAC Computer

A. D. BEARD L. S. BENSKY
D. L. NETTLETON G. E. POORTE

THE RCA BIZMAC computer has been developed as a major element of the RCA BIZMAC system which is primarily intended to handle cyclical accounting, such as inventory control, as employed at the Ordnance Tank and Automotive Command in Detroit, Mich. The computer resulting from this development is a large-scale data-handling device of speed and versatility which economically fulfills the requirements of the system. The design also permits the computer to be used in areas other than cyclical accounting; e.g., digital system simulation and statistical analysis.

The RCA BIZMAC system philosophy dictates that the computer:

1. Be capable of handling large amounts of alphanumeric data at high speeds.
2. Be proficient in making complicated logical decisions.
3. Possess facility in editing and organizing data.
4. Be moderately agile in arithmetic processes.
5. Be able to process variable item and message lengths in order to work efficiently with the remainder of the system.
6. Be competent in preparing data for specialized functions; e.g., sorting or document printing, although the computer need not be highly competent in these functions per se.

The RCA BIZMAC computer may be described as a large-scale serial 3-address stored-program digital machine. It has a magnetic-core memory and employs magnetic tape as its basic medium for input and output. Certain specialized features which make it especially adept in cyclical accounting applications are:

1. Completely variable word length in all internal operations.
2. Highly flexible instruction complement directed toward editing and organizing facility.
3. A control philosophy which offers extreme operational flexibility and simplifies trouble shooting and maintenance.

Description of the RCA BIZMAC Computer

Within this paper it is impossible to transmit all of the design features embodied in the computer. Hence this paper will describe briefly the organization of the computer and emphasize some of the features which distinguish it from other machines.

Basic Outline

The RCA BIZMAC computer processes items, defined as a group of one or more characters having some particular significance; e.g., a numerical quantity, an alphabetic name, a stock number composed of mixed letters and numbers, and messages, consisting of one or more related items, of completely variable length. Each alphanumeric item is handled on a character-by-character basis. Each character consists of six binary digits plus a parity digit. Instruction execution time is a function of the item length; for example, an addition of two 10-digit numbers consumes approximately 500 microseconds, while the addition of two 5-digit numbers consumes only 280 microseconds.

The machine employs a 3-address instruction code and possesses 22 distinct operations each of which has several variations. The principal input and output is through magnetic tapes operating at 10,000 characters per second. As many as 15 individual tapes can be connected to the computer at one time. The magnetic-core memory of the machine is comprised of two banks of 2,048 characters each with 20 microseconds access time, backed up by a 32,768-character auxiliary memory of 5 milliseconds average access time. The logic of the computer is implemented by use of approximately 5,000 vacuum tubes and 18,000 diodes.

The block diagram is shown in Fig. 1. The A, B, and C counters are used to address the memory, A being associated with the left bank, B with the right bank, and C with either bank. Each memory location, capable of storing one character, is individually addressable by these counters.

A. D. Beard, L. S. Bensky, D. L. Nettleton, and G. E. Poorte are with the Radio Corporation of America, Camden, N. J.

The program counter keeps track of the next program step to be executed and is used to locate groups of instructions to be transferred from the auxiliary memory into the high-speed memory. The program subcounter functions as a conventional program counter only for those instructions which have been stored in the magnetic-core memory, which will be referred to as the high-speed memory throughout the remainder of the paper. The memory registers, left and right, receive characters read to or from the memory and serve to supply the adder, symbol recognition, and the magnetic tapes with information. Symbol recognition is used to determine the end of operands, sign of operands, and similar functions. The adder and converter work in excess-3 code and produce a checked result which is temporarily held in the adder output register prior to its transfer to the high-speed memory. The tape logic, in conjunction with the adder output register, accepts characters arriving from tape at an asynchronous rate and transfers them to the high-speed memory. An explanation of storage and flow of instructions and data follows. This will serve to illustrate more fully the functions of the blocks just described.

Instruction Storage and Flow

Instructions

The 3-address instructions of the computer are fabricated from the information contained in eight RCA BIZMAC characters. The operation code is composed of two characters, the first specifying the basic instruction, the second specifying variations of the instruction. Twenty-two basic instructions are provided in the computer. However, the programmer by use of the variation code may exercise options on these basic instructions, thus providing valuable programming flexibility with consequent reductions in computing time. Each of the three addresses consists of two characters whose 12 usable bits (the parity bits are used for checking only) permit the selection of one of the 4,096 individual memory locations. Each address in general specifies one of the limits of an item and may be selected by the programmer for optimum memory usage.

Surge of Instructions

The extensive program for a given computer run is economically stored in the auxiliary memory. However, in order to achieve a reasonable average instruction access time, a relatively small chain of instructions comprising that portion of the program in process is transferred from

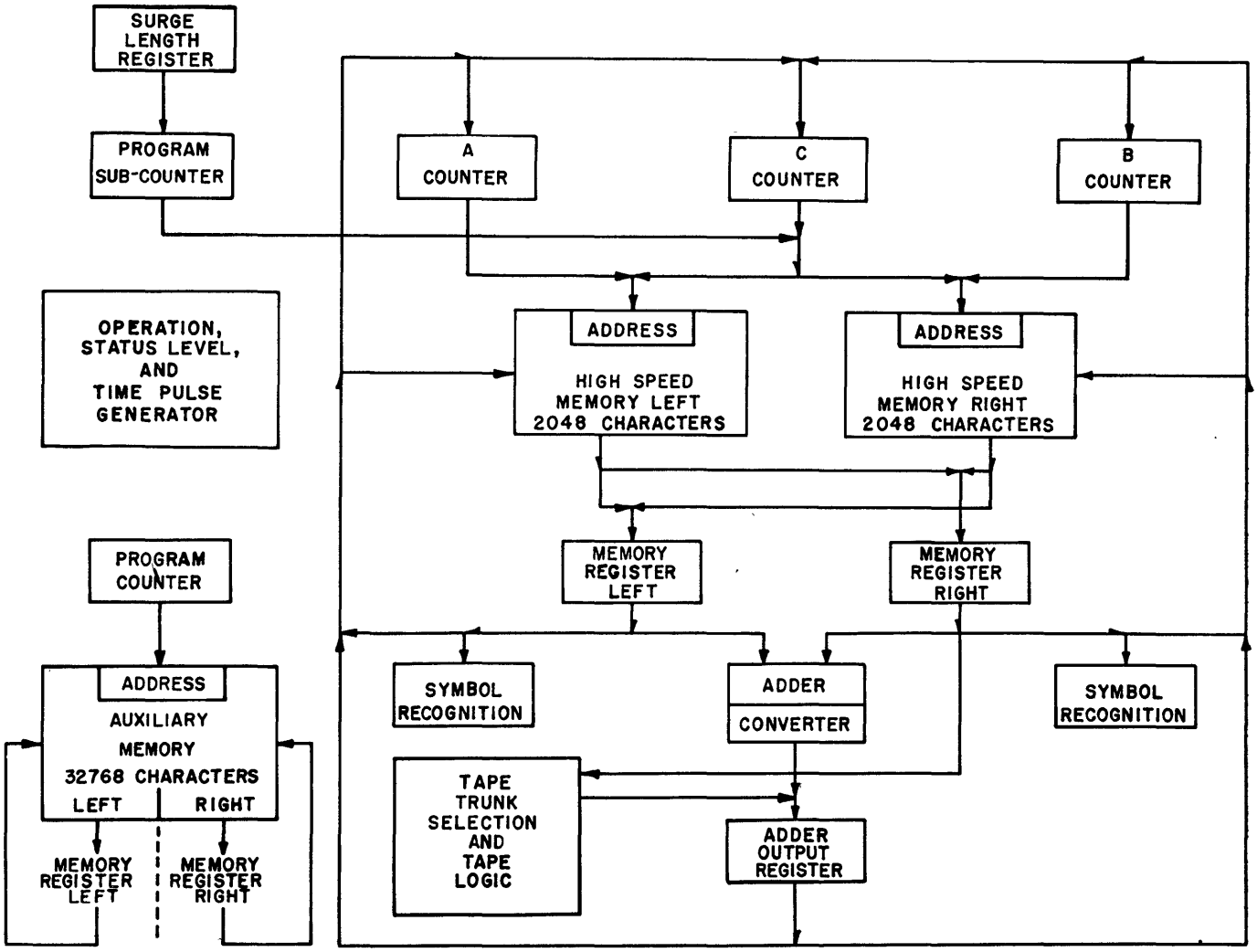


Fig. 1. Computer block diagram

the auxiliary memory and stored in the high-speed memory. This is referred to as a surge of instructions. When these instructions have been executed, an automatic surge of the next group from the auxiliary memory into the high-speed memory occurs. The number of instructions surged is programmed in multiples of four up to a maximum of 64.

Addressing of Instructions

Instructions for transfer from the auxiliary memory are selected by the program counter which maintains the number of the instruction next to be executed. Transfer of control may be effected by setting this counter to an appropriate instruction number. Once a group of instructions has been transferred from the auxiliary to the high-speed memory, the program subcounter functions to address the high-speed memory during the rapid read-out of instructions to the operation registers and the addressing counters. By programming, the contents of this subcounter may be changed to effect a transfer of control to other instructions already contained in the high-speed memory storage. Another important function of the subcounter is to determine when the last instruction in the surge has been executed, and thus initiate the next surge of instructions from the auxiliary memory.

Staticizing of Instructions

The initial step prior to every instruction execution is to read out of the high-speed memory the proper instruction and store it in the operation and variation registers and the *A*, *B*, and *C* counters. This will be referred to as "staticizing an instruction." During the read-out from the high-speed memory, both banks are addressed in parallel by the program subcounter. Four 20-microsecond cycles are required to staticize the instruction since two characters are read out simultaneously. When the staticizing has been completed, the program control signals for the execution of the instruction.

DATA STORAGE AND FLOW

Data enter the computer from magnetic tapes via trunks specified by the program. Characters from tape enter a 1-character buffer and are then transferred to memory registers, left or right, and then to the high-speed memory. Under the control of the program, items are assigned individually to various high-speed memory locations thus enabling the programmer to arrange the information as it enters the computer.

Internal data may be transferred within a memory bank, from one bank to the other, or between the high-speed and the auxiliary memories. All characters transferred internally or externally flow through the memory registers where they are checked for even parity, and special symbols are recognized. Data to be transferred are specified by indicating the address of one limit of an item and are terminated by either recognition of a control symbol or a specified limit. In arithmetic processes, data flow is from the

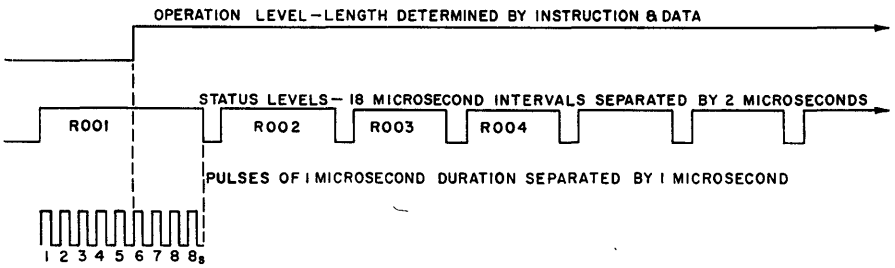


Fig. 2. Three levels of intelligence: operation level, status level, and time pulses

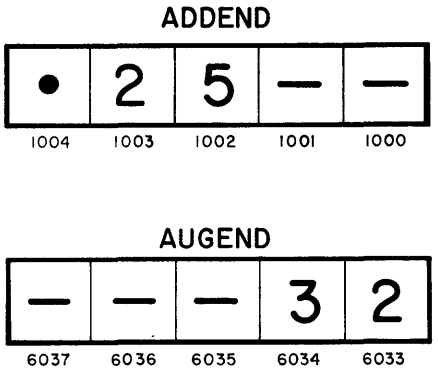


Fig. 4. Arithmetic operands addressed respectively at locations 1000 and 6033

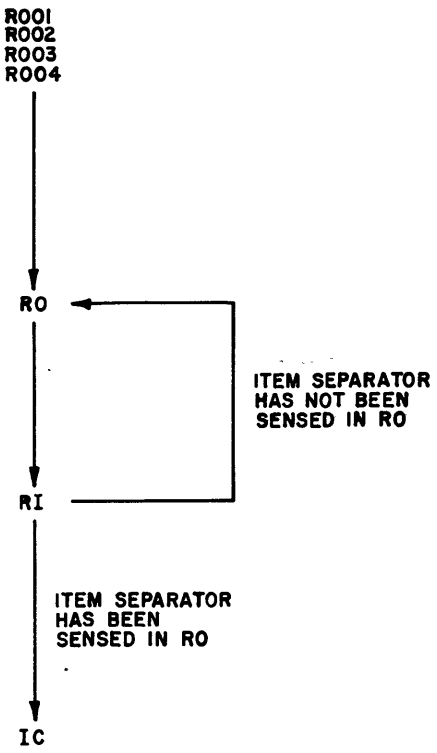


Fig. 3. Status flow diagram of the transfer of data instruction

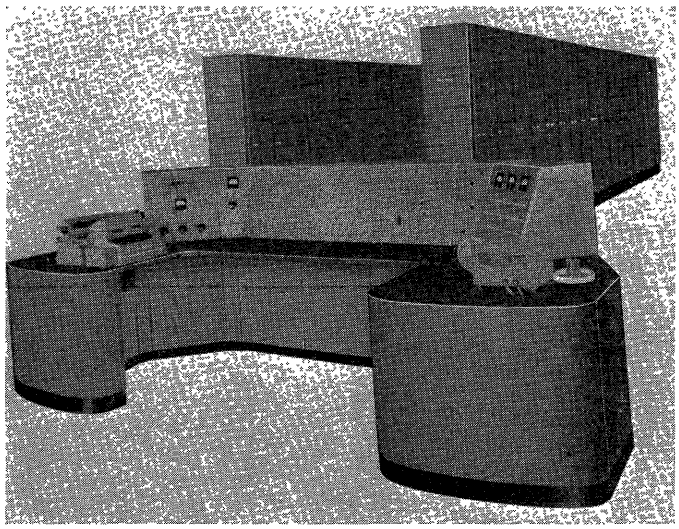


Fig. 5. RCA BIZMAC computer

memory through the memory registers to the adder, then to the adder-output register. From the adder-output register data are returned directly to the memory. Data flow in the arithmetic processes is controlled by recognition of special symbols.

Information is read out to magnetic tape from the memory through the memory register right and then to the trunk specified by the programmer. A return signal derived from the current passing through the tape recording heads is returned to the memory register left and a comparison is made between the two memory registers to insure the correct receipt of information at the recording head.

Computer Concepts

The RCA BIZMAC system requires a computer that can accept and generate magnetic tapes with variable item and message lengths. In order to conserve high-speed memory, and to minimize operating time, variable item lengths are used internally also. This leads to a machine of advanced design, embodying several original concepts.

Three Levels of Intelligence

Three levels of intelligence exist within the computer. First are the 22 individual operations, some of which have several variations. Next are the twelve status levels which cause the machine to perform a simple function. These status levels, with a brief description of each, are found in Table I.

Table I

Status level	Function
ROO1	Read out first part of an instruction
ROO2	Read out second part of an instruction
ROO3	Read out third part of an instruction
ROO4	Read out fourth part of an instruction
RO	Read from the memory
ROM	Read out a multiplier digit
RS	Read out and search for a significant character
RI	Read into the memory
RIC	Read a special character into the memory
RE	Obtain new instructions or data from auxiliary memory
RD	Dummy cycle
IC	Instruction complete—prepare for next instruction

Finally, there is a repetitive chain of eight time pulses which time the elementary operations during a status level. In general, control gates have at least these three levels of intelligence applied: an operation, a status level, and a time pulse. Fig. 2 shows the time relationship between these three control levels.

Most digital computers have the ability to select for subsequent execution one or two or more sequences of instructions. Not only does the RCA BIZMAC computer possess this ability, but it also possesses the ability to select during the execution of a given instruction the most productive sequence of status levels. This selection is made on the basis of the data upon which the computer is operating, and provides the means for eliminating nonproductive steps in data processing. Use of the status level concept also makes possible economic mechanization of instructions and variations thereof.

A simple transfer of data within the high-speed memory will be used to exemplify the use of status levels. See Fig. 3. First, status levels *ROO1*, *ROO2*, *ROO3*, and *ROO4* are executed, thus staticizing the instruction. Then a sequence of *RO* and *RI* levels is executed to transfer successive characters of an item from one group of memory locations to another. After each *RI* a choice, based on the absence or presence of an item separator symbol, is made either to continue the sequence of *RO* through *RI*, or next to use status level *IC* denoting the end of the operation. In this way only the time necessary to transfer the item in question, regardless of length, is used in instruction execution.

Variable Item Length

The RCA BIZMAC system employs variable message and item lengths on all tapes. The computer, which must accept and generate such tapes, uses variable item lengths internally also. This permits optimum use of high-speed memory, and minimizes instruction execution times as well.

In general, the computer operates only on those digits which are present in an item, terminating operations when the lack of significant characters is recognized. If a single digit is present in a certain item location, the computer takes only the time to process that digit. If in that same location in the processing of a succeeding message, four digits are present, the computer will take the longer time necessary to process the four digits.

A simple example involving the addition of two positive numbers will suffice to show the way in which the computer processes items of variable length. Assume that the operands have been placed in the high-speed memory as shown in Fig. 4. In this example the addend has been assumed to have been addressed at high-speed memory location *1000*, and the augend at location *6033*. These addresses are supplied by the addition instruction. In processing these operands the computer first examines, simultaneously, the characters in locations *1000* and *6033*. Upon finding a space in *1000*, the decision is made to continue the search for a digit. The existence of a digit in location *6033* causes the digit, in this case a 2, to be stored in the memory register until the search for the least significant digit of the addend is completed. Once the least significant digits have been located, addition proceeds in a normal fashion until the end of the operands is sensed. In the case of the addend the terminating symbol is an item separator symbol while in the augend a space serves to notify that digits have been exhausted.

Thus, the concept of variable item length is instrumented not only in input devices and storage and work tapes but also within the, RCA BIZMAC computer. Here, it permits a flexibility in the use of the memory, allowing maximum use of available locations together with a minimum operating time not set by some fixed word length or even by the possible maximum length of given items, but by the number of significant digits actually present in an item.

Input-Output

The RCA BIZMAC computer must be able to accept data from input tapes and to generate output in the form which is standard for the RCA BIZMAC system. Therefore, an input tape for use with a given program may contain items of variable length, within a message of variable length, and the precise location in terms of the number of characters after the start of the message is not known. Since information is read from tape with the most significant digit first, the memory location which must necessarily be specified for each item is that of the item separator which precedes the most significant character of that item. These memory locations are contained in a block of addresses that are stored in the auxiliary memory together with the program. These addresses are transferred to the tape control logic, an address at a time, as each item separator coming in from tape is sensed. Should an item be of less than the assigned length, spaces will remain in the memory to the right of the item.

A compress instruction is therefore provided which removes all excess spaces to the right of an item in a specified assemblage of data prior to read-out. Hence, the RCA BIZMAC computer accepts messages in the most compact form, distributes these items in the memory in locations of nonstandard maximum length, and then compresses all items to be read out into the most compact form which is used on all tapes.

Editing

Automatic editing of messages and items is a prominent feature of the RCA BIZMAC system. It is instrumented primarily by the computer. Automatic editing results in minimization of the complexities of input transcription and output printings, and helps maintain magnetic tapes of minimum lengths. Some provisions for editing messages and items are as follows:

Read-In From Magnetic Tape

Of the several means at the programmer's disposal for rearranging items of a message the most versatile is "random composition" during "read-in" from magnetic tape. By random composition the programmer may reorder all or some of the incoming items, and also leave blank areas for those items to be generated by computation if desired. Thus he may compose an output document or abstract a message intended for another device in the system; e.g., an output printer or a sorter.

Computational Editing

To some degree, editing is also accomplished by the computer during computation proper. Since a 3-address instruction code is used, arithmetic instructions can place items to conform with desired output message format. This method of editing must of necessity be restrained by computational programming demands. Transfer item and internal block transfer operations can also aid the editing process.

Write-Out to Magnetic Tape

A third means for accomplishing editing is provided in the computer's flexible write-out-to-tape operation. Segments of a message may be gathered from several areas of the computer high-speed memory and composed into one message on the output tape by using a combination of several of the variations on the write-out operation.

Hierarchy of Memories

To satisfy the RCA BIZMAC system requirements, large amounts of storage and fast processing of data are necessary. Since no one storage medium is available which has the characteristics of fast access time and large storage capacity (10^8 to 10^{10} bits), the requirements must be met by the proper system integration of several types of storage media. These media include magnetic tape and magnetic cores.

The bulk store of the system is the magnetic-tape file. The data from tapes are read into the high-speed random-access core memory of the computer. Once within the memory, the data can be processed and manipulated at high speeds. The fast random-access feature of this memory is an underlying factor in the flexibility that the computer achieves. It is used to match the data rates of the tapes to the computer, it makes possible the random composition of items as they arrive from tape, it provides rapid access

to instructions and data, and it relieves the arithmetic unit from providing storage and shifting registers. The auxiliary storage is a backup store for the core memory. It provides an economical means to store data and instructions thereby effectively increasing the internal memory capacity and achieving a compromise between cost and access time.

External Control

The over-all system operating philosophy is that of centralized control. Appropriate remote indicators and controls from most of the equipments, including the computer, are provided at the system central consoles. Control of the system is under the direction of a centralized team. To control the system efficiently, and to minimize potential human operating errors, simplified controls and proper operating techniques have been designed. The computer itself has a console designed primarily for maintenance and checking operations, and includes a wide variety of status indicators, as well as the necessary controls to test properly and service the computer.

Fig. 5 shows the RCA BIZMAC computer.

Accuracy Control

The system philosophy of centralized control with its concept of minimum human intervention makes automatic detection and correction highly desirable. The problems involved in providing such a feature are many and complex. Detection circuits are necessary to insure reliable output. Once an error is detected, a correction must be made. The correction is a direct function of where the error occurred in the program, what caused it, whether it was a transient or permanent failure, etc. To simplify this complex problem, errors are classified in two general groups. The first group includes errors resulting from a known permanent component failure or major malfunction of external equipment. The machine is stopped immediately and suitable maintenance is performed or replacement is made. The second group includes those errors that result from either transient or questionable permanent-component failure, or voltage transients. This group generally requires a rerun of a portion of the program to determine if a shutdown is required. All errors in this group are handled in the same manner. A complete rerun of the transaction (computation) in process takes place automatically. The necessary clearing, backing up of tape, and other appropriate operations required are performed prior to the rerun. Programmed counters are provided to limit the number of reruns that can be performed.

Examples of the type of error detection found in group one are as follows:

1. Magnetic tape moving forward when it should be moving in reverse, or vice versa.
2. Magnetic tape circuits not operable.
3. End of magnetic tape.
4. The counter which addresses the high-speed memory during instruction read-out is not cycling properly.

Examples of error detection found in group two are as follows:

1. Parity failures.
2. Adder comparator (arithmetic is performed twice. The second addition is performed using complemented operands and the results are then compared).
3. Verify operation where data are compared bit for bit.
4. Arithmetic overflow.

Conclusion

The RCA BIZMAC computer is a major element of the RCA BIZMAC data-handling system. The several novel features which it incorporates permit it to fulfill its missions in data conversion, data editing, and data generation. It is designed to operate upon variable item and message lengths in order that the system may maintain the economy in reduced tape lengths, and that the computer may achieve maximum useful data rates. Every attention has been given to make it a part of an integrated data-handling system.

Programming a Variable-Word-Length Computer

L. S. BENSKY T. M. HUREWITZ R. A. C. LANE A. S. KRANZLEY

INVESTIGATION of commercial applications for electronic data-processing systems has revealed certain basic characteristics. Very high volumes of input and output data are handled with a modest amount of data calculation. Variability in data length, data occurrence, and in procedures for handling of these data is another major characteristic.

L. S. Bensky, T. M. Hurewitz, R A. C. Lane, and A. S. Kranzley are with the Radio Corporation of America, Camden, N. J.

The manner in which the RCA BIZMAC computer handles variability in all of its aspects has provided a uniquely adaptable tool for commercial applications. The intermediate function of preparing the computer for these applications, programming, is therefore unique in many respects.

Working with clear and concise definitions of commercial applications, the programmer is concerned with applying computer flexibility in an optimum manner. The variable-word-length computer permits concentration of effort in applying flexibility to the handling of data. Programming results may be measured in terms of effectiveness in computer-time and storage utilization, and accuracy control tempered by the availability of well-defined problems and programming time.

Efficiency in Program Composition

One of the prime objectives in the writing of data-processing programs is minimization of the over-all computer time required to accomplish a specific task. In commercial applications, where the work load for the computer is essentially of a cyclic nature, significant cost reductions may be realized from effective equipment utilization. These gains take the form of a smaller complement of equipment, or the performance of more tasks with existing equipment.

It has previously been stated that commercial applications are characterized by a high volume of input messages. Each of these messages is handled individually by the computer until all have been processed. Programs for such applications, then, are used in a repetitive fashion, with each cycle representing action taken on a single message. A shortening of the message cycle by only a few milliseconds will have a significant cumulative effect if the cycle is repeated many times. For example, 36 milliseconds cut from a cycle which is repeated 100,000 times amounts to a total saving of 1 hour of computer time.

A complete and well-organized definition of the problem is the starting point from which an efficient program can be obtained. Characteristics of data handled by the computer (as shown on standard data sheets) directly affect the efficiency of programs. One outstanding example of this is the format of output messages to be printed. A good format will make maximum use of tabulating stops, thus reducing the programmed steps needed for line composition. The columnar alignment of items is another criterion of an output format which may be used to relate the problem definition to efficiency in the program.

Messages entering the computer frequently contain coded items which serve as the basis for decision-making. For example, a 1-digit decimal code may be used to distinguish ten distinct types of transactions. In a computer where instruction modification is easily performed, the choice of values for such codes permits reduction of decision-making sequences of instructions. With latitude in establishing such codes, it is possible to make each value correspond to an address in either the high-speed memory or the auxiliary memory. Decision-making would then consist of modifying a transfer of control (or other instruction) with the particular code value for each message. Other features which facilitate a reduction in the number of instructions executed, but not in the number provided, are consecutiveness, and ordering of codes by relative volume of appearance.

From descriptions of input and output data, the statement of intermediate operations may also be examined for areas affecting efficiency in the program. Inasmuch as the majority of commercial programs are executed in a cyclic fashion, the establishment of volume figures assumes major importance in selecting operation sequences. Emphasis is placed on optimizing those operations which represent major streams of data flow. Conversely, infrequently used sequences may be complex at small cost in over-all computer time.

Unique Characteristics of Variable-Word-Length Programming

Introduction

It has been stated previously that two basic requirements necessary for the production of efficient programs are: (1) a knowledge of the problem; and (2) a knowledge of the equipment to be used in the solution of the problem. Some of the flexibility available to the programmer in the RCA BIZMAC computer is briefly described and illustrated in the remainder of this section. Utilization of the instructions is explained in terms of the application of the RCA BIZMAC equipment to business problems.

A functional subdivision is made into the three broad categories: computer handling of programs, computational aspects, and editing of data.

Computer Handling of Programs

Programs for the RCA BIZMAC computer are stored permanently on magnetic tape. The initial preparation of the magnetic tape is accomplished in exactly the same manner as the initial input of data to the system. That is, a 7-channel paper tape is first prepared and verified, followed by transcription to magnetic tape. The indexed programs are then stored in what may be called a magnetic-tape program file. It is possible to store up to 2,500 different programs on a single reel of tape.

Part of the initial setup of a computer operation consists of transferring the proper sequence of instructions from the magnetic-tape program file to the auxiliary memory. This is accomplished by a short routine which will search the tape for the desired program and make the transfer. Excluding tape search time, it takes approximately 1 minute to load the auxiliary memory completely. Programs that require less than the full capacity of the auxiliary memory take proportionately less insertion time.

For most business applications, the entire program for a particular computer run will be loaded initially. In cases where many irregularities are to be handled automatically as part of one computer operation, those portions of the program used infrequently may be stored on magnetic tape. When necessary, these portions may be entered without any appreciable time delay.

It is quite probable that some applications will require that the program be modified at regular intervals, that running control totals be maintained, that dates be changed, etc. In these cases, a new program tape may be prepared at the conclusion of a computer run. Undoubtedly, changes will be made in existing programs as systems and procedures are altered. Changes of this nature can be made through the use of special service routines designed for this purpose.

Instructions are "surged" from the auxiliary memory into the high-speed memory in groups (multiples of four) of up to 64 instructions. The quantity of data transferred during a surge contributes to the time required to perform the surge function. There are cases where the large surges would be wasteful because only a few instructions are used before transferring control, and other cases where a part of the high-speed memory usually reserved for the storage of instructions is needed for data storage of temporary work area. It is clear that the ability to change the length of the surge automatically during the running of a program is an extremely useful tool for minimizing running time.

An excellent way to reduce program-access time is to refrain from surging. This can be accomplished through effective use of the instructions and data that are already stored in the high-speed memory, by transferring control to instructions that are stored in the memory, and by appropriate modification of instructions.

Address modification, or the ability of a computer to operate on its own instructions in the same manner as it operates on data, is an extremely important characteristic of efficient programming. It is useful where an identical sequence of instructions must be repeated in each case using operands stored in different memory locations. For example, it may be required to advance an address or series of addresses by a count of one preceding each cycle through a "loop." It may be required to advance or decrease an address by any constant amount. In any event, this can be accomplished through the use of the "binary add" or "binary subtract left-justified" instructions.

As has been mentioned, the "binary add" instruction adds RCA BIZMAC characters according to their binary equivalents. For example, excluding parity bits, $K + \$ = (101010)_2 + (000111)_2 = (110001)_2$. Octally, this would be $K + \$ = (52)_8 + (07)_8 = (61)_8$. A bit is carried into the next addition cycle if the sum of two characters is greater than $(77)_8$. Both auxiliary and high-speed

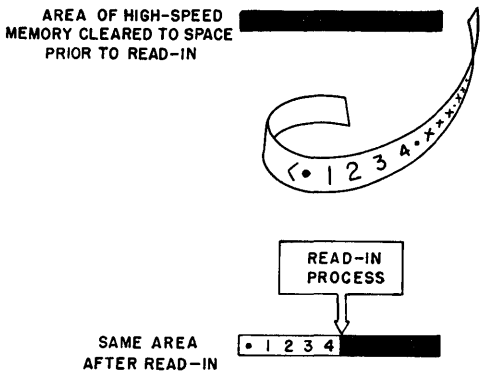


Fig. 1 (left). Status of high-speed memory locations allotted for an item with a maximum length of ten decimal digits

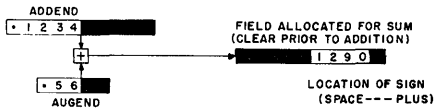


Fig. 2. Addition-operands justified left

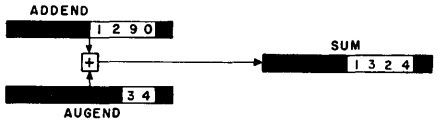


Fig. 3. Addition-operands justified right

memories are addressed octally in the program, and each address is expressed as four octal digits. Therefore, to advance an address by one, two, six, etc., it is necessary to store $(0001)_8$, $(0002)_8$, $(0006)_8$, etc., in the high-speed memory. Each constant uses only three memory locations (including one for the item separator). In cases where unused instruction addresses are available for the storage of the constant, no additional high-speed memory locations are required. For decreasing an address by a constant amount, "binary subtract left-justified" is employed in the same manner as the "binary add" instruction.

It is very convenient in many cases to generate the address instead of modifying an existing address. This can be accomplished quite simply by a "transfer of data" instruction to the surge area.

One application for this technique is in pigeonhole sorting where the numbers to be sorted are converted to addresses which are then placed in a "transfer of data" instruction prior to its execution. The same technique saves considerable program-running time, instruction storage and programming effort, when applied to problems involving the posting of amounts to one of a number of totals. For example, in life insurance accounting it may be necessary to accumulate the premium amounts received by state. The amounts need not be arranged in state order. A code indicating the state total to which each amount is to be posted may be associated with each amount. If the programmer is free to establish the state code for internal processing, a 2-character code may be used that corresponds to the storage location of the state total. The code is now transferred to a single "decimal add" instruction (A and C or B and C addresses). Thus, the posting is accomplished without the series of decisions normally required for the accomplishment of similar tasks. Only three instructions are used, including the completion of the addition. If the code is fixed by the procedural requirements it is often possible to establish a transformation procedure for conversion to memory addresses.

In computer handling of programs, a major concern in the use of subroutines is the return to the main routine after execution of the subroutine. Calling in a particular subroutine is no problem since it may be stored at a fixed location on the auxiliary memory. However, since a particular subroutine may be required in several different parts of a program, the return to the main routine necessarily must be variable. The return to the main routine is handled in the RCA BIZMAC computer by storing the address of the point of return in an unused portion of the high-speed memory prior to executing the subroutine. This is conveniently accomplished with the use of the "set up" instruction. The last instruction in the subroutine then must be a "refer" instruction which scans the high-speed memory address containing the point of return to the main routine.

The procedure outlined in the foregoing serves as a convenient tool in the initial development of a program. It represents a scheme whereby a rather complex and lengthy programming task can be subdivided into a number of simpler tasks. These simpler tasks may be programmed and coded individually. Two unused memory locations are assigned to each routine.

Each routine is terminated with a "refer" instruction which scans these locations. This assignment also serves to identify the particular routine. When each of these subroutines has been written, the program may begin with a series of "set up" instructions (one for each routine) in order to tie each of the parts together to form the whole program.

This method is particularly efficient in those types of problems that require these subroutines to be related in a variety of ways, depending on certain decisions and conditions which are established only during the running of the program. In these cases, the proper series of "set up" instructions must follow the decision-making portions of the program occurring within the main routine. Each "set up" instruction controls the "refer" action from one subroutine to the next.

Computational Aspects

An important and fundamental concept of a variable-item-length computer is the complete flexibility available in the definition of items for a given problem.

In an insurance company's billing operation, the policy holder's name and address may be defined as an item. One need only consider the names and addresses of a few friends to see that this particular item could range in length from less than 20 characters to more than 100. On the other hand, to facilitate arrangement of last names in alphabetical order, it may be desirable to define the last name as a separate item. Similarly, identifying information such as stock numbers or policy numbers, with their associated handling codes, may be handled as one item. It should be emphasized that item definition depends primarily on the meaning and intended use of the information.

Since the RCA BIZMAC computer is basically a serial machine and the high-speed memory is used to perform the functions of registers for the storage of all operands and results of arithmetic operations, the number of digits involved in arithmetic operations is unlimited. The use of multiple precision techniques in programming is wholly unnecessary. For example, to program the addition of two positive numbers of 100 decimal digits each requires one instruction that specifies the locations of the least significant digits for each of the operands and for the sum. This applies also in the event the numbers are each one decimal digit in length; or are of unequal length with no restriction on the degree of inequality.

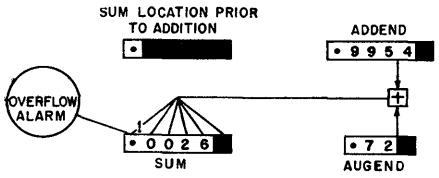


Fig. 4. Overflow in addition operation

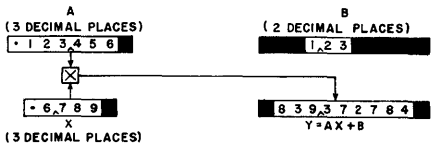


Fig. 5. The computation of y = ax + b involving decimal points

When considering the lengths of the various items to be processed in a particular computer run, it is necessary to determine the maximum number of digits that may occur in each item. Although this is not a requirement for magnetic-tape storage a field consisting of one or more memory locations must be established in the high-speed memory for the storage of each item. This field must be capable of storing the longest of the items for which it was allotted. Thus, if an item is less than maximum length it will be read in to the memory as indicated by Fig. 1. In this case the item is said to be left-justified with reference to its field.

If the item in Fig. 1 is to be added to a second item with a maximum length of five characters, it is necessary only to address the least significant locations of the fields containing the two operands in one "decimal add" instruction. No program routine is required to locate the least significant digits of the items, no instruction to justify the items right is necessary, and no extraction of digits is needed prior to the addition. According to the logic of the RCA BIZMAC computer the characters are read out from each of the fields serially. The actual addition takes place only when two non-space characters are recognized. This is illustrated in Fig. 2.

Gains in programming facility are accompanied by the minimization of the time required for the computer to execute the addition. Time is minimized because the addition is performed on significant digits only, and the time required to search each location for the least significant decimal digit is only half that of the addition of each pair of digits.

Let it be required to add the sum obtained in the above illustration to another 2-digit number where both operands are justified right. This operation is illustrated in Fig. 3.

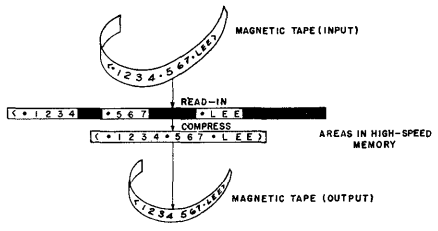


Fig. 6. Compression of output data for maximum tape utilization

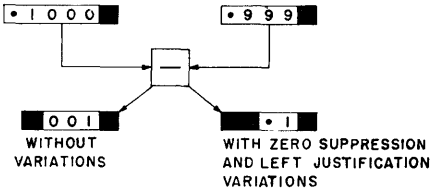


Fig. 7. Illustration of zero suppression with left justification

Spaces to the left of each of the operands add nothing to the sum. In the RCA BIZMAC computer, the addition operation ends on recognition of spaces or item separators to the left of both operands. Only those digits representing the sum are written into the field allocated, and the spaces to the left of the sum are those that were present prior to the addition. "Decimal subtract" and "multiply" are handled in a similar manner making optimum use of variable item lengths.

The ability to multiply and accumulate (i.e., $x = ab + c$) with one instruction is incorporated in the RCA BIZMAC computer. This is accomplished by addressing the product (ab) to a location in which another item is stored (c). This particular feature evolves from the use of high-speed memory locations as partial-product registers in the multiplication process.

Although the number of digits occurring in the results of arithmetic operations is usually predictable, there may be certain instances, particularly in the case of some accumulated totals, where the capacities of the fields allotted for the results may be exceeded (see Fig. 4). In order to avoid such a possibility, a variation of the decimal arithmetic operations is provided to detect overflow. It will actuate an overflow alarm, indicating that a result has exceeded the capacity of its field.

The handling of decimal points in the RCA BIZMAC computer can be considered neither floating-point nor fixed in the strict sense of these terms. It employs the advantage of both and might be termed "absolute variable." All arithmetic operations treat the operands as whole numbers, and decimal points per se are entirely excluded from the operations.

It is the function of the programmer to determine the magnitudes of each operand, and by proper addressing, to insure that the correct results of operations involving decimal places are obtained. In most business applications, this is easily accomplished. In scientific computation this could be accomplished using well known floating-point tecnniques. The computation of $y = ax + b$ involving decimal points is illustrated in Fig. 5 where " $\wedge$ " indicates the theoretical location of the point.

In business data processing there is a definite need for handling alphabetic information in a manner similar to numerical data. This is illustrated quite vividly in the sorting of names into alphabetic order, and determination of the relative order of items such as stock numbers, policy numbers, etc., where the items consist of both alphabetic and numeric information. In addition, considerations relative to address modification (previously explained) require the inclusion of binary operations.

The numbers and letters in the RCA BIZMAC code have been assigned binary codes such that the commonly accepted ordering (i.e., 0, 1, 2,..., 9; A, B, C, ..., Z) corresponds to increasing numerical values of their binary equivalents. Thus, the ability to handle the characters of an item according to their true binary values implies the ability to determine the normal alphanumeric ordering of a pair of digits (e.g. $3A$, CM, AB, etc.).

To determine which of two items containing combinations of alphanumeric characters is of larger magnitude, it is only necessary to subtract one from the other and examine the sign of the results. The "binary subtract" instruction is used. The test of the sign is accomplished with the "conditional transfer of control." Thus, a combination of only two instructions is sufficient to determine the relative magnitudes of two items regardless of lengths and alphanumeric composition.

Editing of Data

Consideration must be given to the form and content of data handled by the various pieces of equipment comprising the system. Messages entering these equipments must have proper control symbols, item ordering, and positioning to insure that output information is correctly produced. The process of adding, deleting, and rearranging data within a message for subsequent operations is defined as editing. Most of the editing work is handled by the computer, since it is the

most flexible unit for data processing.

Many of the problems in editing incoming data for the computer are handled by the "read-in" instruction. It was previously noted that items in an incoming message may be placed randomly in the high-speed memory under the direction of the program. Random composition on the read-in is used to store items in such a manner that subsequent data transfers are minimized. For example, arithmetic instructions require operands to be located in opposite banks of the high-speed memory. When two incoming items are to be added, they are initially placed in opposite banks by the read-in instruction. In the same way, items may be stored in correct order for writing out to tape. The best arrangement of the read-in is made by deciding how each incoming item is to be subsequently used.

Items in a reference file which are not required for a particular computer operation may be discarded by the "read-in" instruction by addressing to a special discard location. In some operations where long, consolidated reference file messages enter the computer for summarization or modification, several hundred high-speed memory locations and clearing operations for these locations may be saved. Item-separator symbols which are required as control symbols for subsequent operations may be generated by providing item addresses in excess of input message requirements. Otherwise, these additional control symbols must be placed in the high-speed memory as part of a special editing subroutine.

Although it is possible to arrange the items of an input message in a form suitable for output, it is frequently the case that more than one arrangement is desired. This happens where several types of output documents and an updated reference file are to be produced by the computer operation. Intelligent use of the rather potent transfer-of-data instruction is called for here.

The amount of processing that can be accomplished during one computer operation is dependent to a large extent on the amount of storage that is available. This refers to the number of tapes that can be utilized as well as internal storage. In the RCA BIZMAC computer, as many as 15 separate magnetic tapes can be connected at the same time. Five of these are input tapes and ten are output tapes. The output tapes may be used in a read or a write status. This multiplicity of input and output tapes makes it possible to prepare many different output documents in a single computer operation, thus eliminating other machine intervention between the computer and the high-speed printer.

Maximum tape utilization is obtained by using the "compress" instruction to eliminate nonessential space symbols from the data to be written out (see Fig. 6). Such space symbols represent the difference between the maximum number of locations allowed for items in the high-speed memory, and the actual number of characters in the items.

Composition of messages routed from computer to printer presents a somewhat different problem. Here, the emphasis is shifted to line and page composition. Left or right columnar alignment of items (justification), item ordering, and the insertion of control symbols are the elements to be considered in programming for a specific printed format. A requirement in business data processing, particularly with regard to output documents, is the suppression of excess zeros in numeric quantities.

The suppression of zeros in the RCA BIZMAC computer is automatic and is included as a variation of the "decimal add" and "decimal subtract" instructions (see Fig. 7). Each item to be written out must be examined for its justification. If an item is left-justified (having the most significant character next to the item-separator symbol) and is to be printed with right justification, a "justify right" instruction must be executed. When items must be left-justified, an arithmetic operation using the justify left variation will provide the necessary shift.

Evaluation of Variable-Word-Length Programming

In programming for commercial data-processing problems, minimization of over-all computer time is the major criterion for evaluation.

Instruction Utilization

The RCA BIZMAC computer provides a list of instructions specifically designed to handle variability in data and variability in processing requirements. An illustration of this versatility is furnished by considering the decimal arithmetic instructions. The foregoing text mentions in detail the following functions:

1. Operands are located and results are stored by one instruction ($A + B = C$), eliminating the need for special registers.
2. No shifting (justification) of operands is necessary since instructions operate on significant characters only.
3. There is no preparatory extraction of characters before operation.
4. The operations are algebraic and sign handling is automatic.
5. Decimal points are program-handled in an absolute variable fashion.
6. Automatic suppression of zeros may be included.
7. Results may be automatically justified.

In addition, the particular features of the computer which reduce instruction requirements for total data-processing functions are important to the programmer. For example, storage of the program in the high-speed memory assists in the use of address modification techniques which are an important requirement of commercial data-processing programs. Random composition of data entering the computer is another feature which makes possible more efficient combinations of instructions in subsequent data processing steps.

Another category of instruction functions is useful for purposes of data-processing economy outside of the computer operation proper. For example, use of the "compress" instruction is related specifically to tape-storage economies. It should be noted this also provides subsequent reduction of read-in time throughout the system.

Storage Utilization

As mentioned previously, the use of variable and adjustable-field item lengths permits appreciable savings. Magnetic-tape space is saved by putting on the tape only these characters that actually appear in the input, thus reducing the length of tape necessary to hold a given piece of information. This implies a reduction in the number of reels of tape and consequently fewer tape-handling devices required. It also results in saving of computer time since tape movement consumes a large portion of the total time. (Savings up to 70 per cent over maximum field data storage are achieved.) Finally, a saving is achieved in the internal storage required by having to provide only sufficient storage for each item to accommodate the maximum length of that item, rather than having to provide the same maximum amount of storage for each item. In the former case, the storage required is merely the total of the maximum item lengths for all of the items to be handled, while in the latter the storage required would be the maximum required for the maximum length item multiplied by the number of items to be handled.

The computer is designed to permit optimum use of internal storage facilities. The use of an auxiliary memory for the

storage of both data and instructions in any desired array, the variability of surge length, the unrestricted ability to transfer control within the high-speed memory, and finally the reduction in high-speed memory work area requirements brought about by the ability to compose output data randomly, exemplify the tools that are in the hands of the programmer to maintain an efficient balance between time and storage.

Adaptability to the Problem

Throughout this paper, pertinent characteristics of commercial data-processing problems have been mentioned when descriptions of functional and programming features called for them. A review of these characteristics in retrospect completes the evaluation of variable-word-length programming by the RCA BIZMAC.

Variability in data (size and occurrence) and in procedural requirements is handled by providing a list of instructions, and other computer characteristics, especially designedf or flexibility. This makes it possible to write very compact programs for handling business problems. Flexibility in assignment of internal storage at several levels is the necessary corollary to the efficient programming of commercial data-processing problems.

From detailed analyses of many types of commercial data-processing problems, it has been found that some categories of procedures are related to the end use of output data. Of greater importance, considerable similarity of procedures among apparently unrelated applications is evident. It turns out that these similar procedures are composed of basic operations which are identical in nature. This simplifies the programming task particularly where equipment functions are specifically designed to respond to problem data-characteristics.

Detailed programming for many commercial data-processing problems will furnish the final evaluation of variable-word-length programming. It is certain that this expansive phase in the art of programming will be marked by further advances in programming techniques; these new techniques in turn will influence future equipment design. Such advances will be based on substantially improved knowledge of data characteristics, the standardization of routine business processes, and the acceptance of mathematical techniques as tools in scientific management.