AFIPS

CONFERENCE PROCEEDINGS

VOLUME 24

1963 FALL JOINT COMPUTER CONFERENCE

1963
SPARTAN BOOKS, INC.
Baltimore, Md.
CLEAVER-HUME PRESS
London

Library of Congress Catalog Card Number: 55–44701

Sole Distributors in Great Britain, the British Commonwealth and the Continent of Europe:

CLEAVER-HUME PRESS
10-15 St. Martins Street
London W. C. 2

LIST OF JOINT COMPUTER CONFERENCES

1. 1951 Joint AIEE-IRE Computer Conference, Philadelphia, December 1951
2. 1952 Joint AIEE-IRE-ACM Computer Conference, New York, December 1952
3. 1953 Western Computer Conference, Los Angeles, February 1953
4. 1953 Eastern Joint Computer Conference, Washington, December 1953
5. 1954 Western Computer Conference, Los Angeles, February 1954
6. 1954 Eastern Joint Computer Conference, Philadelphia, December 1954
7. 1955 Western Joint Computer Conference, Los Angeles, March 1955
8. 1955 Eastern Joint Computer Conference, Boston, November 1955
9. 1956 Western Joint Computer Conference, San Francisco, February 1956
10. 1956 Eastern Joint Computer Conference, New York, December 1956
11. 1957 Western Joint Computer Conference, Los Angeles, February 1957
12. 1957 Eastern Joint Computer Conference, Washington, December 1957
13. 1958 Western Joint Computer Conference, Los Angeles, May 1958
14. 1958 Eastern Joint Computer Conference, Philadelphia, December 1958
15. 1959 Western Joint Computer Conference, San Francisco, March 1959
16. 1959 Eastern Joint Computer Conference, Boston, December 1959
17. 1960 Western Joint Computer Conference, San Francisco, May 1960
18. 1960 Eastern Joint Computer Conference, New York, December 1960
19. 1961 Western Joint Computer Conference, Los Angeles, May 1961
20. 1961 Eastern Joint Computer Conference, Washington, December 1961
21. 1962 Spring Joint Computer Conference, San Francisco, May 1962
22. 1962 Fall Joint Computer Conference, Philadelphia, December 1962
23. 1963 Spring Joint Computer Conference, Detroit, May 1963
24. 1963 Fall Joint Computer Conference, Las Vegas, November 1963

Conferences 1 to 19 were sponsored by the National Joint Computer Committee, predecessor of AFIPS. Back copies of the proceedings of these conferences may be obtained, if available, from:

- Association for Computing Machinery, 14 E. 69th St., New York 21, N. Y.
- American Institute of Electrical Engineers, 345 E. 47th St., New York 17, N. Y.
- Institute of Radio Engineers, 1 E. 79th St., New York 21, N. Y.

Conferences 20 and up are sponsored by AFIPS. Copies of AFIPS Conference Proceedings may be ordered from the publishers as available at the prices indicated below. Members of societies affiliated with AFIPS may obtain copies at the special "Member Price" shown.

Volume	*List Price*	*Member Price*	*Publisher*
20	$12.00	$7.00	Macmillan Co., 60 Fifth Ave., New York 11, N. Y.
21	6.00	6.00	National Press, 850 Hansen Way, Palo Alto, Calif.
22	8.00	4.00	Spartan Books, Inc., 301 N. Charles St., Baltimore 1, Md.
23	10.00	5.00	Spartan Books, Inc.
24	16.50	8.25	Spartan Books, Inc.

NOTICE TO LIBRARIANS

This volume (24) continues the Joint Computer Conference Proceedings (LC55–44701) as indicated in the above table. It is suggested that the series be filed under AFIPS and cross referenced as necessary to the Eastern, Western, Spring, and Fall Joint Computer Conferences.

CONTENTS

Page *Page*

PREFACE

This volume provides the permanent record of the Technical Program of the 1963 Fall Joint Computer Conference.

In organizing the program, specific attention was directed toward providing a broad cross-section of material which covered all aspects of the field and reversing the trend toward concentration on software apparent in the past few Conferences. The accomplishment of this goal is exhibited by the large number of papers contained herein, encompassing hardware, software, systems, and applications. In addition, a group of papers presents specific proposals for the application of computers to important social problems.

The volume contains the full text of the 55 papers selected for presentation and discussion at the Conference. It represents the culmination of much hard work by many people—in particular, the authors, to whom we are indeed grateful for their contributions; the Technical Program Committee, session chairmen, panelists, and reviewers who put together the program; and the entire Conference Committee whose untiring efforts made the Conference possible.

JAMES D. TUPAC
Conference Chairman
1963 FJCC

AN EXPERIMENT IN NON-PROCEDURAL PROGRAMMING

Jesse H. Katz and William C. McGee
TRW Computer Division
Thompson Ramo Wooldridge Inc.
Canoga Park, California

1. INTRODUCTION

Computer processes have traditionally been specified by means of procedural languages. That is, a computer program is generally expressed as a sequence of steps.

Three reasons can be cited to explain why programs are written in a step-by-step fashion. First, digital computers themselves, for reasons of economy, operate sequentially and thus require step-by-step directions called instructions. Thus, in the early days of computers, programs written at the machine language level were—of necessity—one hundred percent procedural. The methods of machine language programming undoubtedly had an impact on the specification of the higher-order languages developed subsequently (e.g., ALGOL, COBOL, FORTRAN), with the result that these higher-order languages are also largely procedural.

A second reason for writing programs as sequences of steps is that many computer processes are inherently sequential; i.e., certain things must be done before other things are done. This is particularly true of numerical processing.

Third, we use step-by-step programming in many cases simply because we find it easier to think of a process as a sequence of steps, regardless of whether the process is truly sequential or not. In this way, we limit to a manageable number those things which must be kept in mind at any one time.

The extent to which procedural languages are now ingrained makes it certain that they will be with us for many years to come. But the perpetuation of these languages—to say nothing of their proliferation—is not without disadvantages. The major disadvantage is the difficulty of specifying computer processes in a manner which is independent of the computer on which they are to be carried out.

As a simple example, consider the evaluation of the expression

$$y = ax + b$$

On a one-address computer, the machine language procedure for this process might be expressed as follows:

```
CLA  X
MPY  A
ADD  B
STO  Y
```

On a three-address computer, the procedure would be quite different:

```
MPY  A/X/TEMP
ADD  TEMP/B/Y
```

Clearly, the procedure for this process is strongly machine dependent.

The more sophisticated languages like ALGOL and FORTRAN overcome problems like the one illustrated above by admitting the *non-procedural* specification of certain arithmetic processes. In FORTRAN, the process is expressed simply as

```
Y = A*X + B
```

Such statements imply an ordering of the operations to be carried out (i.e., that dictated by the rules of algebra), but are properly classed as non-procedural since the procedure for carrying out the required operations is not explicitly stated.

By means of such non-procedural specifications, large parts of a computer process can be stated in an entirely non-procedural way in a language like FORTRAN, if one is willing to construct sufficiently long and complex expressions. However, most computer processes also require operations which cannot be expressed non-procedurally in a language like FORTRAN. The main culprits are conditional operations, i.e., operations which are dependent on values of input data, and input/output operations.

With such problems in mind, workers in data processing theory have been giving increasing attention to non-procedural programming languages. A non-procedural programming language may be defined as one in which a computer process is expressed solely in terms of the *results* of the process, rather than in terms of a procedure by which the results are to be produced. In engineering terms, the non-procedural language allows the programmer to define the *final state* of the computer as a function of its various possible *initial states.*

Since machine language will remain largely procedural, at least in the foreseeable future, a non-procedural language implies the need for a mechanism for transforming a non-procedural specification into a machine language program. Such a mechanism will be similar in function to current compilers; but because its input will differ fundamentally from that of current compilers, this mechanism might be better termed a *procedure-writer.* The essential feature of the procedure-writer is that the production of a step-by-step procedure is taken completely out of the hands of the programmer. As a consequence, the procedure-writer is free to develop procedures which are most appropriate to the computer which will execute them.

With this introduction, we shall now discuss a completed phase of a corporate sponsored research project being carried out in the TRW Computer Division of Thompson Ramo Wooldridge Inc. The aim of this research project is to explore techniques for developing machine-independent software. As a manufacturer of computers, TRW is faced with a major problem common to the industry: the need to repeat the development of software for each newly developed computer. We feel that by bringing to bear the techniques of automatic programming on software programs themselves, the time and cost of developing new computers and their attendant software can be cut significantly.

As part of our investigation, we have selected a typical software item, a conventional symbolic assembly program, and have derived a non-procedural specification of this process. For a non-procedural language, we turned to the Information Algebra, which was developed by the Language Structure Group of the CODASYL Development Committee.[1] Although intended primarily as a theoretical development, the Information Algebra proved quite adequate as a non-procedural programming language. The specification of the assembly program will be discussed in Section 3 of this paper, but first we shall describe the fundamental concepts of the Information Algebra itself.

2. BASIC CONCEPTS OF THE INFORMATION ALGEBRA

An essential feature of the Information Algebra is the manner in which the data involved in a process are represented. For each *property* in the process, a *property value set* is defined which contains all the possible values which the property can assume. For example, if *employee number* is a property, its value set might consist of the integers between 1 and 100,000. A *coordinate set* is then defined as the set of all properties involved in a process. For example, a coordinate set might consist of the properties *employee number, employee age,* and *employee sex.*

A *property space* is defined as the set of all points which can be found by taking one value from the value set of the first property, one value from the value set of the second property, and so forth, up to and including one value from the value set of the last property in the coordinate set. For the coordinate set (*employee number, employee age, sex*), such a point might be

$p = (12345, 43, M)$, where 12345, 43, and M are values from the value sets of employee number, employee age, and sex respectively.

Obviously, in all but trivial processes there will be a very large number of points in the property space. In our example, if the property value sets for employee number, age, and sex contain 100,000, 100, and 2 values respectively, the property space will in fact contain (100,000) (100) (2) = 20 million points. Equally obviously, not all of these points will be involved in a given processing. One of the major contributions of the Information Algebra is to give us a way of referring to and manipulating *sets* of points in the property space.

The above method for representing data is quite abstract. It does not, for example, say anything about how data points would be stored in a computer memory or on magnetic tape; or how they would be organized to take advantage of data redundancy or irrelevancy. It is this abstraction, in fact, which makes the Information Algebra a promising candidate for a machine-independent data processing language.

The principal function of data processing is to create output files from input files. In the Information Algebra, a file is represented as a set of points in property space and is called an *area.* The creation of output files from input files is equivalent to transforming one or more given areas (i.e., the input files) into one or more new areas (the output files). Unlike actual data processing, however, this transformation does not in any way depend on the *sequence* of operations, and it is sometimes helpful, in fact, to regard the transformations as occurring simultaneously.

To express these transformations, a number of operators or *functions* are provided. One such function is the *function of areas* (FOA), defined to be an operator which associates one and only one value with an area. For example, if an area A of the property space of the previous example has been established, then a function of this area f(A) might be defined which sums the values of age for each point in the area and divides the sum by the total number of points in the area. In other words, the function computes the average age of the persons in the file represented by area A.

In addition to representing files, areas of property space can be used to represent groupings within a file. In the illustration, it might be desired to group the points in area A such that all points in one group have age values between 0 and 4, all points in another group have age values between 5 and 9, and so forth. This facility is provided by a *glumping function* which partitions A into subsets of points called *elements* in such a way that an element contains all points in A having identical values for the given glumping function. The entire set of elements is called simply a *glump* and is denoted G(g, A), where g is the glumping function and A is the area.

One of the purposes of glumping functions is to define areas (elements are areas) which can then be operated on with a function of areas to define new areas in the property space. The FOA used for this purpose is called a *function of a glump* (FOG). The FOG creates a new area by defining a point in property space (not necessarily a different one) for each element in a glump. This area is denoted

$$A' = H(g, A)$$

where A is the original area, g is the glumping function for that area, and H is the function of a glump which creates the new area. The points of the new area are defined by stating the values which each property in the coordinate set is to take on for each element in the glump. Since glump elements are areas, these values are stated as functions of areas. For a coordinate set with k properties $q_1, q_2, \ldots, q_k$, the function of a glump is written generally as

$$H = \begin{cases} q_1' = f_1 \\ q_2' = f_2 \\ \cdot \\ \cdot \\ \cdot \\ q_k' = f_k \end{cases}$$

where the f_i, $i = 1, 2, \ldots, k$, are functions of an area, and the notation

$$q_i' = f_i \qquad i = 1, 2, \ldots, k$$

means that property q_i is to take the value assigned by f_i.

The function of a glump affords a way of defining a new file from a single file. In many applications, however, a new file is to be created by simultaneously considering the data in two or more files. For such situations, the Information Algebra introduces the *area set,* an ordered set of areas $(A_1, A_2, \ldots, A_n)$. The areas in an area set need not be disjoint (i.e., non-overlapping), but usually are. With such a set of areas, one can envision a process which selects a point from area A_1, then selects a point from A_2 and so forth until a point has been selected from each area. The set of points selected by this process is referred to as a *line,* and the points in the line are *ordered* since the areas in the area set were ordered.

One can now envision this process being repeated, e.g., selecting the same points as before from A_1 through A_{n-1}, but a new point from A_n. This new set of points defines another line. Similarly, other lines may be defined, until all possibilities have been exhausted. If the areas contain $m_1, m_2, \ldots, m_n$ points respectively, a total of $\prod_{i=1}^{n} m_i$ lines can be formed.

Out of the totality of lines intersecting an area set, one can now envision a selection process in which only those lines meeting a certain condition are selected. The set of lines so selected is called a *bundle,* and the condition for selection is expressed in a *bundling function.* The bundling function is a special case of a function of lines (FOL) which in general associates a single value with a line (in the same way that an FOA associates a single value with an area). The bundling function is called a *selection FOL,* because it can associate only two values with any line: *true* or *false.* The lines in a bundle are those lines for which the associated bundling function is true. All lines for which the bundling function is false are disregarded. As will be seen later, the purpose of the bundle is to associate points in different areas.

In many processes it is desired to associate a single new record with each occurrence of matching records from two or more input files. In the Information Algebra, this is accomplished with a *function of a bundle* (FOB). An FOB creates a new area by defining a point in property space for each line in a bundle. The area is denoted

$$A' = F(b, A_1, A_2, \ldots, A_n)$$

where b is the bundling function, the ordered set $(A_1, A_2, \ldots, A_n)$ is the area set over which b is defined, and F is the function of a bundle which creates a new area. The points in the new area are defined in a manner similar to the way points are defined in a function of a glump; the difference is that the functions which assign values to the various properties of the coordinate set are now functions of lines instead of functions of areas:

$$F = \begin{cases} q_1' = f_1 \\ q_2' = f_2 \\ \cdot \\ \cdot \\ \cdot \\ q_k' = f_k \end{cases}$$

In the next section the various functions of the Information Algebra will become clearer, as we see how they are put to use in specifying an Assembly Program.

3. APPLICATION OF THE INFORMATION ALGEBRA TO AN ASSEMBLY PROGRAM

To illustrate the application of the Information Algebra to software processes, we have selected a symbolic assembly program as a test device. Our assembler is a "subset" of the familiar SAP assembler used with the IBM 704/709/7090 family of computers; and for illustration purposes we shall assume that assembly is to be performed for machines in this family. In specifying this assembler, we have held its features close to the minimum so that we can concentrate here on the essentials. Nonetheless, all the necessary facilities for symbolic assembly are provided, including the automatic incorporation of subroutines from a subroutine file.

The symbolic instructions acceptable to this assembler are of two kinds: machine instructions and pseudo instructions. The former consists of instructions carried out by the computer (e.g., CLA, ADD, STO); while the latter consists of the following: BCD, BSS, DEC, LIB, OCT, ORG, REM, and SYN. Figure 1 sum-

Pseudo Instruction	Function	Type of Data in Location Field	Type of Data in Address Field	Effect on "Location Counter"
BCD	Provides for assembly of one to nine words of BCD information. High-order digit of address field specifies number of words set aside.	blank, symbol	BCD count: decimal integer BCD data: alphanumeric characters	+n, where n = 1,2,...,9
BSS	Reserves number of words specified in address field.	blank, symbol	decimal integer	+n
DEC	Provides for assembly of one decimal word.	blank, symbol	decimal number	+1
LIB	Provides for incorporating a subroutine from subroutine file. Symbol in location field identifies subroutine.	symbol	blank	+n
OCT	Provides for assembly of one octal word.	blank, symbol	octal number	+1
ORG	Sets "location counter" to value specified in address field.	blank	decimal number	set to r
REM	Provides for comment in program.	blank	(extended address field) alphanumeric characters	none
SYN	Sets symbol in location field "equal" to symbolic address in address field.	symbol	symbol ± decimal integer	none

Figure 1. Summary of Pseudo Instructions.

marizes the functions of the pseudo instructions, including their effect on the "location counter"; and shows the types of data acceptable in their respective location fields and address fields.

3.1 *Properties and Areas*

The coordinate set of properties required for this problem is illustrated in Figure 2. The ith property in the coordinate set is designated q_i. The general role of the several properties is as follows:

(1) q_1 is a sequence number associated with each symbolic instruction in the input program and the subroutine file. This property supplies the information which is normally supplied by the *order* in which symbolic instructions are fed to the assembler.

(2) q_2 through q_{11} correspond to the fields in a symbolic instruction, namely: location, operation, indirect address indicator, address base, address modifier, tag, decrement, remarks, BCD count and BCD data. The latter two fields are relevant for the BCD instruction only.

(3) q_{12} is a subroutine identifier associated with each symbolic instruction in the subroutine file.

(4) q_{13} is the binary location associated with each word produced by the assembler, and q_{14} is the binary word itself.

(5) q_{15} is an error indicator, for printing errors on the side-by-side listing.

(6) q_{16} through q_{18} are "intermediate" properties, required in establishing the intermediate areas.

An interesting sidelight to the selection of the coordinate set of properties for this problem is the generalized approach which one is virtually forced to take in defining the "entities" of the process. For example, the binary output of an assembler usually takes the form of a series of records, each containing a "load address" and a number of binary words to be loaded into consecutive locations, starting at the load address. Such an approach is obviously machine-dependent, since the format of the output record is strongly influenced by the peripheral equipment which is available to the assembler (and to the loader which loads the object program for execution). A preferable approach, therefore, is to define the binary output "records" simply as pairs of values, consisting of a load address and a binary word to be loaded at that address. The process of combining such records into a suitable physical record can then be left to the mechanism which translates the non-procedural specification into a running assembly program.

Also illustrated in Figure 2 are the input and output areas used in the specification of the assembler, together with an indication of the relevance or non-relevance of each property to these areas. Relevance is indicated by an X, and non-relevance by the symbol Ω.

The input areas are the symbolic input file A_1; and the symbolic subroutine file A_2. A third area A_3, which defines the mapping of symbolic operation codes into binary operation codes, may also be viewed as an input file. The rules for this mapping could have been buried in the specification, but have been isolated in this manner to make the specification as independent as possible of the computer on which the object programs are to be run.

The purpose of the assembler is to transform the input areas A_1 and A_2, with the help of the

PROPERTIES	PROPERTY VALUE SETS	AREAS				
		A_1 SYMBOLIC INPUT FILE	A_2 SYMBOLIC SUBROUTINE FILE	A_3 SYMBOLIC OP CODES AND BINARY EQUIVALENTS	A_4 OBJECT PROGRAM	A_5 SIDE-BY-SIDE LISTING
q_1 =SEQUENCE NUMBER	1,2,...	X	X	Ω	Ω	X
q_2 =SYMBOLIC LOCATION	blank,symbol	X	X	Ω	Ω	X
q_3 =SYMBOLIC OPERATION	3 alphabetic characters	X	X	X	Ω	X
q_4 =SYMBOLIC INDIRECT ADDRESS INDICATOR	blank, *	X	X	Ω	Ω	X
q_5 =SYMBOLIC ADDRESS BASE	blank, symbol, decimal number, octal number, *	X	X	Ω	Ω	X
q_6 =SYMBOLIC ADDRESS MODIFIER	blank, ± decimal integer	X	X	Ω	Ω	X
q_7 =SYMBOLIC TAG	blank, 0,1,...,7	X	X	Ω	Ω	X
q_8 =SYMBOLIC DECREMENT	blank, decimal integer	X	X	Ω	Ω	X
q_9 =REMARKS	47 alphanumeric characters	X	X	Ω	Ω	X
q_{10}=SYMBOLIC BCD COUNT	1,2,...,9	X	X	Ω	Ω	X
q_{11}=SYMBOLIC BCD DATA	54 alphanumeric characters	X	X	Ω	Ω	X
q_{12}=SUBROUTINE ID	symbol	Ω	X	Ω	Ω	Ω
q_{13}=BINARY LOCATION	15 bits	Ω	Ω	Ω	X	X
q_{14}=BINARY WORD	36 bits	Ω	Ω	X	X	X
q_{15}=ERROR CODE	E	Ω	Ω	Ω	Ω	X
q_{16}=COUNT 1	1,2,...	Ω	Ω	Ω	Ω	Ω
q_{17}=COUNT 2	1,2,...	Ω	Ω	Ω	Ω	Ω
q_{18}= AUXILIARY SEQUENCE NUMBER	1,2,...	Ω	Ω	Ω	Ω	Ω

Figure 2. Properties and Areas for Assembler Specification.

information in area A_3, into two output areas: A_4, which is equivalent to object program output of an assembler; and A_5, which corresponds to the side-by-side listing usually produced by an assembler. Our goal, then, is to derive two expressions which define each of the output areas in terms of the input areas; that is, we seek expressions of the general form

$$A_4 = f_1(A_1, A_2, A_3)$$
$$A_5 = f_2(A_1, A_2, A_3)$$

As might be expected, the functions f_1 and f_2 are quite complicated. To simplify the notation, we have introduced a number of intermediate areas. In general, each intermediate area is a function of one or more previous intermediate areas.

3.2 *Summary of Assembly Process*

The process by which the intermediate areas, and eventually the final output areas, are created is outlined below. The referenced equations are those in Appendix I.

(1) An area is constructed which consists of those LIB instructions in the original symbolic program whose location fields contain mutually distinct symbols. This area is then used, in conjunction with the area representing the symbolic subroutine file, to mark each valid LIB instruction with the memory requirement of its corresponding subroutine. Invalid LIB instructions, i.e., those with multiply-defined symbols and those for which no corresponding subroutine exists, are marked as errors. See eqs. (1), (2), (3).

(2) The symbolic instructions in the main symbolic program are re-sequenced to allow for the insertion of library subroutines. (We shall return to this particular operation in Section 3.3 and show in detail how it is accomplished.) The symbolic instructions belonging to the selected subroutines are also re-sequenced, so that they can be merged with the main program. The union of these two sets of instructions thus represents the "full" symbolic program. See eqs. (4), (5), (6).

(3) The BCD instruction gives rise to a vari-

able number of words in the object program, according to the number supplied in the address field of the instruction. The next step, then, is to re-sequence the symbolic program to leave room for words generated by the BCD instructions. See eq. (7).

(4) We next strip the symbolic program of REM, LIB and SYN instructions, since these instructions produce no words of code in the object program. A new property, "auxiliary sequence number," is introduced to record the re-sequencing. The auxiliary sequence number is then further adjusted to reflect the "expansion" caused by BSS instructions. Finally, with the aid of the ORG instructions, the auxiliary sequence number is mapped into binary location; the ORG instructions are then stripped from the program. See eqs. (8), (9), (10).

(5) The program, as it now stands, and the set of (stripped) SYN instructions are operated upon to produce a symbol table. All instructions whose location fields contain multiply-defined symbols are marked as errors. Also, any SYN instruction whose location field contains an undefined symbol is marked as an error. See eqs. (11), (12), (13).

(6) All object program words generated by BCD, DEC and OCT instructions are developed. See eqs. (14), (15).

(7) The set of machine instructions in the program is then partitioned into two subsets: those with symbolic addresses and those with non-symbolic addresses. The two classes of machine instructions are translated separately. Any instruction whose location field contains an undefined symbol or whose symbolic operation is invalid is marked as an error. The two output files, i.e., the side-by-side listing and the object program, can now be formed merely by taking the union of appropriate areas developed along the way. See eqs. (16), (17), (18), (19).

3.3 *One Equation in Detail*

The specification of the entire assembly process can be expressed by a set of nineteen equations. These are given in Appendix I.

At this point we shall discuss one of these equations in detail, so that greater insight might be gained into the methods of the Information Algebra. In step (2) of Section 3.2 it was explained that the main symbolic program must be re-sequenced to allow for the insertion of library subroutines. The expression for this operation is eq. (4) of Appendix I, namely,

$$A_9 = H_3(q_1, F_2(B_3)) \cup I_2'(B_3)$$

where

$$B_3 = (q_{1:1} < q_{1:2}, A_7, A_8) \;;$$

$$F_2 = (q_{17}' = q_{16:1}) \;;$$

and

$$H_3 = (q_1' = q_1 + \Sigma\, q_{17},\ q_i' = q_i,\quad i = 2, 3, \ldots, 11, 15)$$

The components of this expression are as follows:

(1) The area A_8 represents the symbolic program. Each valid LIB instruction contained in the program has been marked with the memory requirement of its corresponding subroutine (property q_{16}); and each invalid LIB instruction has been marked as an error.

(2) The area A_7 represents the subset of A_8 which consists of valid LIB instructions.

(3) The expression B_3 stands for a bundle which is formed by joining every point in A_7 to every point in A_8, and selecting only those lines for which the bundling function $(q_{1:1} < q_{1:2})$ is "true." In this case the bundling function means that a line is selected only if property q_1 (sequence number) in the first area of the bundle (A_7) is less than property q_1 (again sequence number) in the second area (A_8). By this means we "associate" each instruction in A_8 with *all* LIB instructions with lower sequence numbers. This is illustrated in Figure 3, in which the letters A, B, C, D, E, F, G stand for instructions with monotonically increasing sequence numbers and C and E are LIB instructions. Note that instructions D and E are associated only with C; F and G are associated with both C and E;

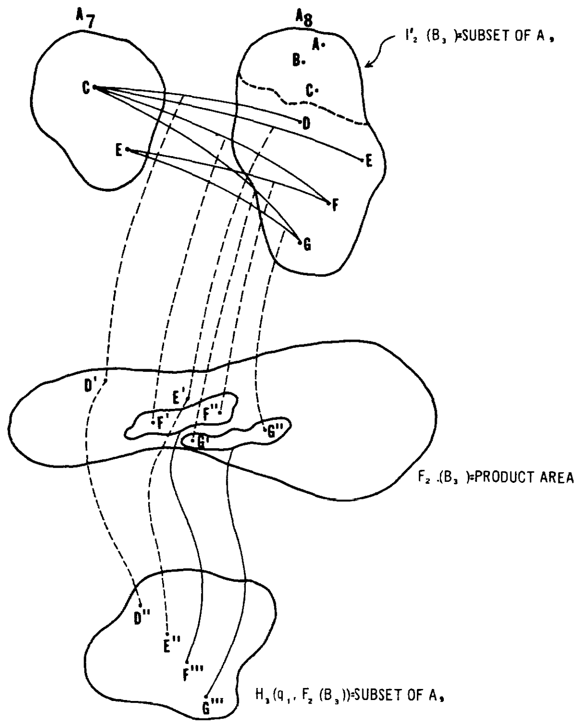


Figure 3. Formation of Typical Intermediate Area.

while A, B and C are not associated with any LIB instruction.

(4) The function F_2 maps each line in the bundle into a point in a "product area" in accordance with the definition

$$F_2 = (q_{17}' = q_{16:1})$$

This simply means that the property q_{17} (count 2) for every point in the product area is given the value of the property q_{16} (count 1) for the point in the first area (A_7) on the corresponding line. The remaining property values for the points in the product area are, by convention, simply copied from the points in the second area (A_8) on the corresponding lines. The result is one or more copies of each instruction having "associated" LIB instructions, with each copy holding "count 1" of an associated LIB instruction (i.e., the memory requirement of the corresponding subroutine).

(5) The product area is now glumped by the glumping function q_1, meaning that all points having the same sequence number will be in the same element. Thus, in Figure 3, F′ and F″ will be in the same element, G′ and G″ will lie in another element, and D′ and E′ will occupy separate elements by themselves. The elements of this glump are then mapped into a new area, one point per element, by the function of a glump H_3, as follows:

$$H_3 = (q_1' = q_1 + \Sigma\, q_{17},\ q_i' = q_i,$$
$$i = 2, 3, \ldots, 11, 15)$$

By means of this expression, each point in the new area is assigned values for properties $q_1, q_2, \ldots, q_{11}$ and q_{15}. By convention, the values of all other properties become null. Essentially, the new area contains a single point for each original instruction having at least one preceding LIB instruction, but now having a sequence number equal to the old sequence number plus the number of words required for all subroutines called by preceding LIB instructions.

(6) Finally, the area A_9 must also include $I_2'(B_3)$. This expression simply represents the complement of the intersection of the bundle B_3 with A_8, i.e., all instructions having no preceding LIB instructions (A, B and C in Figure 3), and hence for which no sequence number adjustment is required. The desired area A_9 is then the simple union of $I_2'(B_3)$ and the function of a glump H_3.

4. CONCLUSIONS

The original Information Algebra report[1] made it clear that the Information Algebra is a powerful language which provides for a concise formulation of those processes which we term "business data processing." The work we have just described suggests that the Information Algebra is applicable to a much broader class of computer processes. In particular, its successful application to the specification of an assembly program suggests that it might be suitable, with some minor extensions, as a software programming language.

Perhaps the most encouraging aspect of this exercise is that we were able to specify precisely a computer process without any assumptions as to the computer on which it would be executed. Nowhere, for example, were we required to postulate input or output devices, media, or formats. Similarly, we were able to sidestep completely the problem of the general organization of an assembly program (e.g., whether it is a "one-pass" or "two-pass" process), which is generally established by such computer characteristics as core size and the types and number of peripheral devices available. Our specification, in short, is strictly machine-independent, and could be rendered with equal facility into a suitable procedure for any computer.

It might be noted that our example does have machine-dependent aspects; however, these are a result of the process being specified rather than the specification technique. An assembly program by definition produces object programs for a specific computer, and the characteristics of this computer will unavoidably show up in the specification. In particular, the machine instruction format of the computer, and to a corresponding degree, the format of input symbolic instructions, are implicit in the specification of the assembler. This is quite a different type of machine dependency than the one we are concerned about, and in no way detracts from the machine-independency of the assembly process per se. With suitable generalizations, machine-dependent aspects of the former type can presumably be eliminated.

Of some concern to us was the fact that in developing the non-procedural specification, we were not able to avoid thinking in a step-by-step fashion. A little reflection, however, assured us that we were simply unable to adequately comprehend all parts of a complex process at once, and were using the same kind of "thought partitioning" which a mathematician uses in using two shorter expressions when a single longer one would do. To convince ourselves that this was indeed the case, we attempted to construct a single expression for each of the files produced by the assembly process by simple substitution of the constituent expressions. This was abandoned when the expression became longer than a standard desk, but not before we assured ourselves that it could in principle be done.

Only one major difficulty was encountered in using the Information Algebra, that being the difficulty of mapping a single point into a set of points, where the number of such points and the values associated with each are derived from the values associated with the given point. For such operations, it appears that an "inverse function of an area" might be a useful extension to the algebra. Except for minor notational difficulties, the Information Algebra otherwise proved to be quite adequate.

A problem of major concern which lies ahead is how a non-procedural specification is to be used to produce running programs. Two basic approaches might be used. In the first, the non-procedural specification would be introduced directly into the computer (with appropriate code transliteration) and executed interpretively. Such a process appears to be grossly inefficient with present computers, but might be feasible in computers with extensive facility for parallel operations.

The second approach, and probably the more practicable, is to translate the non-procedural specification into a machine-language program with a "procedure-writer" program. From the abstract nature of a non-procedural specification, it is clear that such a procedure-writer would have to know a great many details about the computer for which it is writing procedures (programs). To keep the procedure-writer itself independent of the object computer, such details should be introduced as input parameters rather than being buried in the procedure-writer. It is equally clear that in addition to the object program, the procedure-writer will also have to generate "human-procedures," i.e., procedures for using the program. The approach is illustrated in Figure 4.

Since non-procedural specifications are machine-independent, it is clear that the same specification can be introduced repeatedly to the procedure-writer, each time with details about different object computers, to produce different programs and usage procedures for each of the object computers.

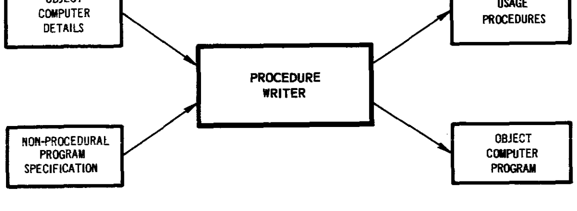


Figure 4. Operation of Procedure Writer.

Finally, since the procedure-writer itself is a computer process, it can be expressed in non-procedural form and introduced to an existing procedure-writer to produce a procedure-writer for any other computer.

The price paid for machine-independent computer process specification, of course, is in the complexity of the procedure-writer. With such rapid advances in the art of compiler construction, however, the procedure-writer appears technically feasible in the near future. When it is, a long step toward the goal of machine-independent programming will have been taken.

APPENDIX I: Assembly Program Specification

Let the symbolic input file A_1 be partitioned into two subsets $A_1{}^{(1)}$ and $A_1{}^{(2)}$, consisting of LIB instructions and non-LIB instructions respectively. Thus

$$A_1^{(1)} = G(q_3, A_1) \mid_{q_3 = \text{“LIB”}}$$

and

$$A_1^{(2)} = A_1 \cap (A_1^{(1)})$$

Let A_6 denote the set of LIB instructions whose location fields contain mutually distinct symbols. Then

$$A_6 = F_I(B_1) \qquad (1)$$

where F_I is the *identity* FOB;

$$B_1 = (q_{2:1} = q_{2:2} \wedge q_{16:1} = 1, H_1(q_2, A_1^{(1)}), A_1^{(1)}) ;$$

and

$$H_1 = (q_2' = q_2, q_{16}' = \Sigma\, 1)$$

Note that the complement of the intersection of B_1 with $A_1^{(1)}$ consists of all LIB instructions with multiply-defined location fields.

By considering the symbolic subroutine file A_2 jointly with A_6, one can construct the area A_7 which represents the LIB instructions marked with the memory requirement of the subroutines which they call. Thus

$$A_7 = H_2(q_1, F_I(B_2)) \qquad (2)$$

where

$$B_2 = (q_{12:1} = q_{2:2}, A_2, A_6)$$

and

$$H_2 = (q_i' = q_i, i = 1, 2, \ldots, 11, q_{16}' = \Sigma\, 1)$$

Note that each point in the complement of the intersection of B_2 with A_6 represents a LIB instruction for which there is no subroutine in the symbolic subroutine file.

We now designate by A_8 an area containing the following: the original symbolic program with valid LIB instructions marked with the memory requirement of corresponding subroutines, and with invalid LIB instructions marked as errors.

$$A_8 = A_1^{(2)} \cup A_7 \cup F_1(1, I_2'(B_2)) \cup F_1(1, I_2'(B_1)) \qquad (3)$$

where

$$F_1 = (q_{15}' = \text{“E”})$$

and "E" stands for "error."

Joint consideration of A_8 with A_7 enables us to construct A_9, which represents the original symbolic program re-sequenced to allow insertion of library subroutines.

$$A_9 = H_3(q_1, F_2(B_3)) \cup I'_2(B_3) \qquad (4)$$

where

$$B_3 = (q_{1:1} < q_{1:2}, A_7, A_8) ;$$

$$F_2 = (q_{17}' = q_{16:1}) ;$$

and

$$H_3 = (q_1' = q_1 + \Sigma\, q_{17}, q_i' = q_i, i = 2, 3, \ldots, 11, 15)$$

Let $A_9^{(1)}$ designate the subset of A_9 consisting of valid, re-sequenced LIB instructions. Then

$$A_9^{(1)} = G(q_3 \cent q_{15}, A_9) \mid_{q_3 \cent q_{15} = \text{“LIB”} \cent \Omega}$$

Operating jointly on $A_9^{(1)}$ and A_2, we construct an area A_{10} which represents subroutine instructions selected by LIB instructions and which are re-sequenced for insertion in the main program.

$$A_{10} = F_3(q_{2:1} = q_{12:2}, A_9^{(1)}, A_2) \qquad (5)$$

where

$$F_3 = (q_1' = q_{1:1} + q_{1:2}, q_{12}' = \Omega)$$

The area

$$A_{11} = A_9 \cup A_{10} \qquad (6)$$

now represents the "full" symbolic program.

Next, we further re-sequence this program to accommodate "expansion" of BCD instructions.

Thus, we get

$$A_{12} = H_4(q_1, F_4(B_4)) \cup I_2'(B_4) \qquad (7)$$

where

$$B_4 = (q_{1:1} < q_{1:2}, G(q_3, A_{11}) \mid_{q_3 = \text{"BCD"}}, A_{11});$$

$$F_4 = (q_{16}' = q_{10:1});$$

and

$$H_4 = (q_1' = q_1 + \Sigma\,(q_{16} - 1), q_i' = q_i,$$
$$i = 2, 3, \ldots, 11, 15)$$

The symbolic program is next "stripped" of REM, LIB and SYN instructions, which produce no object program code; an "auxiliary sequence number" is introduced to record the new sequencing. It will be seen, during the next several mappings, that this property will be transformed into the final binary location. Let $A_{12}^{(1)}$, $A_{12}^{(2)}$ and $A_{12}^{(3)}$ designate the subsets of A_{12} consisting of SYN, REM and LIB instructions respectively. Then

$$A_{12}^{(1)} = G(q_3, A_{12}) \mid_{q_3 = \text{"SYN"}};$$

$$A_{12}^{(2)} = G(q_3, A_{12}) \mid_{q_3 = \text{"REM"}};$$

$$A_{12}^{(3)} = G(q_3, A_{12}) \mid_{q_3 = \text{"LIB"}};$$

and

$$A_{12}^{(4)} = A_{12} \cap (\bigcup_{i=1}^{3} A_{12}^{(i)})'$$

represents all other instructions in A_{12}. The "stripped" symbolic program A_{13} may now be written as

$$A_{13} = H_5(q_1, F_1(B_5)) \cup F_5\,(1, I_2'(B_5)) \qquad (8)$$

where

$$B_5 = (q_{1:1} < q_{1:2}, \bigcup_{i=1}^{3} A_{12}^{(1)}, A_{12}^{(4)});$$

$$H_5 = (q_i' = q_i, i = 1, 2, \ldots, 11, q_{18}' = q_1 - \Sigma\,1);$$

$$F_5 = (q_{18}' = q_{1:1})$$

Note that $A_{12}^{(2)}$ and $A_{12}^{(3)}$ above are components of the side-by-side listing.

Next, the auxiliary sequence number is adjusted to take into account the BSS instructions. Thus

$$A_{14} = H_6(q_1, F_6(B_6)) \cup I_2'(B_6) \qquad (9)$$

where

$$B_6 = (q_{1:1} < q_{1:2}, G(q_3, A_{13}) \mid_{q_3 = \text{"BSS"}}, A_{13});$$

$$F_6 = (q_{16}' = q_{5:1});$$

and

$$H_6 = (q_i' = q_i, i = 1, 2, \ldots, 11,$$
$$q_{18}' = q_{18} + \Sigma\,(q_{16} - 1))$$

The symbolic program is next stripped of ORG instructions, and binary locations are supplied for all others. Designating by $A_{14}^{(1)}$ and $A_{14}^{(2)}$ the subsets of A_{14} consisting of ORG instructions and non-ORG instructions respectively, we have

$$A_{14}^{(1)} = G(q_3, A_{14}) \mid_{q_3 = \text{"ORG"}};$$

$$A_{14}^{(2)} = A_{14} \cap (A_{14}^{(1)})';$$

and can write

$$A_{15} = H_7(q_1, F_7(B_7)) \cup F_8(1, I_2'(B_7)) \qquad (10)$$

where

$$B_7 = (q_{1:1} < q_{1:2}, A_{14}^{(1)}, A_{14}^{(2)});$$

$$F_7 = (q_{16}' = q_{5:1}, q_{17}' = q_{18:1});$$

$$H_7 = (q_i' = q_i, i = 1, 2, \ldots, 11, q_{13}'$$
$$= q_{16}(\max\{q_{17}\}) + q_{18} - \max\{q_{17}\} - 1);$$

and

$$F_8 = (q_{13}' = k - 1 + q_{18:1}, q_{18}' = \Omega)$$

where k is an arbitrary origin. Note that $A_{14}^{(1)}$ is a component of the side-by-side listing.

We now take the first step towards the creation of a symbol table and construct A_{16}, which represents all instructions in $A_{15} \cup A_{12}^{(1)}$ whose location fields contain mutually distinct symbols.

$$A_{16} = F_I(B_8) \qquad (11)$$

where

$$B_8 = (q_{2:1} = q_{2:2} \wedge q_{16:1}$$
$$= 1 \wedge q_{2:1} \neq \text{"b"}, H_1(q_2, A_{15} \cup A_{12}^{(1)}), A_{15} \cup A_{12}^{(1)})$$

and "b" indicates "blank." Note that all instructions in $A_{15} \cup A_{12}^{(1)}$ with multiply-defined symbols in their location fields will belong to the complement of the intersection of B_8 with $A_{15} \cup A_{12}^{(1)}$.

Next, the symbol table A_{17} is constructed with q_2 the symbol and q_{13} the corresponding binary location.

$$A_{17} = A_{16}^{(2)} \cup F_9(B_9) \qquad (12)$$

where

$$B_9 = (q_{5:1} = q_{2:2}, A_{16}^{(1)}, A_{16}^{(2)});$$

$$F_9 = (q_2' = q_{2:1}, q_{13}' = q_{13:2} + q_{6:1});$$

$$A_{16}^{(1)} = G(q_3, A_{16}) \mid_{q_3 = \text{"SYN"}};$$

and

$$A_{16}^{(2)} = A_{16} \cap (A_{16}^{(1)})'$$

Each point in the complement of the intersection of B_9 with $A_{16}^{(1)}$ represents a SYN instruction whose address field symbol cannot be matched with the location field contents of any point in $A_{16}^{(2)}$.

The area

$$A_{18} = A_{16} \cup F_1(1, I_1'(B_9)) \cup F_{10}(1, I_2'(B_8)) \quad (13)$$

where

$$F_{10} = (q_{15}' = \Omega \leftarrow (q_{2:1} = \text{“b”}) \rightarrow \text{“E”})$$

represents the program of $A_{15} \cup A_{12}^{(1)}$, with error codes supplied. Note that $G\ (q_3, A_{18})|_{q3=\text{“SYN”}}$ is a component of the side-by-side listing and that the binary location for these instructions is Ω, as required.

It is convenient at this point to consider A_{18} partitioned into six subsets as follows:

$$A_{18}^{(1)} = G(q_3, A_{18})|_{q_3=\text{“BCD”}};$$
$$A_{18}^{(2)} = G(q_3, A_{18})|_{q_3=\text{“BSS”}};$$
$$A_{18}^{(3)} = G(q_3, A_{18})|_{q_3=\text{“DEC”}};$$
$$A_{18}^{(4)} = G(q_3, A_{18})|_{q_3=\text{“OCT”}};$$
$$A_{18}^{(5)} = G(q_3, A_{18})|_{q_3=\text{“SYN”}};$$

and

$$A_{18}^{(6)} = A_{18} \cap \left(\bigcup_{i=1}^{5} A_{18}^{(i)} \right)'$$

which represents machine instructions.

The expanded and translated BCD instructions can be written as

$$A_{19} = F_{11}(1, A_{18}^{(1)}) \bigcup_{j=1}^{8} F_{12}^{(j)}\ (q_{10:1} > j, A_{18}^{(1)}) \quad (14)$$

where

$$F_{11} = (q_{14}' = q_{11:1}^{(1)})$$

and the $F_{12}^{(j)}$, $j = 1, 2, \ldots, 8$ constitute a class of FOB's as follows:

$$F_{12}^{(j)} = (q_1' = q_{1:1} + j, q_i' = \Omega, i = 2, 3, \ldots, 11,$$
$$q_{13}' = q_{13:1} + j, q_{14}' = q_{11:1}^{(j+1)}) \quad j = 1, 2, \ldots, 8$$

The translated DEC and OCT instructions can be written as

$$A_{20} = F_{13}(1, A_{18}^{(3)} \cup A_{18}^{(4)}) \quad (15)$$

where

$$F_{13} = (q_{14}' = q_{5:1})$$

It is now convenient to regard the machine instructions $A_{18}^{(6)}$ as partitioned into two subsets $A_{18}^{(7)}$ and $A_{18}^{(8)}$, the former having symbolic addresses and the latter non-symbolic addresses.

$$A_{18}^{(7)} = G(g_1, A_{18}^{(6)})|_{g_1=(q_5 \epsilon V_2 - \{\text{“b”}\})}$$

where V_2 is the property value set of the second property, viz., symbolic location; thus V_2—{“b”} represents the set of all possible symbols. And

$$A_{18}^{(8)} = A_{18}^{(6)} \cap (A_{18}^{(7)})'$$

Making use of the file of symbolic operation codes with binary equivalents A_3, we can now represent the translated machine instructions with non-symbolic addresses by

$$A_{21} = F_{14}(B_{10}) \cup F_{15}(1, I_2'(B_{10})) \quad (16)$$

where

$$B_{10} = (q_{3:1} = q_{3:2}, A_3, A_{18}^{(8)});$$
$$F_{14} = (q_{14}' = q_{14:1} \oplus q_{4:2} \oplus q_{7:2} \oplus q_{8:2}$$
$$\oplus (q_{13:2} + q_{6:2} \leftarrow (q_{5:2} = \text{“*”}) \rightarrow q_{5:2});$$

and

$$F_{15} = (q_{14}' = 0, q_{15}' = \text{“E”})$$

Above it is assumed that $q_{14:1}$ is the “skeleton” machine instruction, into which bits are merged according to other components of the symbolic instruction. The character “*” in F_{14} is the symbolic representation for “current value of location counter.”

Making use of the symbol table, we can represent translated machine instructions with symbolic addresses by

$$A_{22} = F_{16}(B_{11}) \cup F_{15}(1, I_3'(B_{11})) \quad (17)$$

where

$$B_{11} = (q_{3:1} = q_{3:3} \wedge q_{2:2} = q_{5:3}, A_3, A_{17}, A_{18}^{(7)})$$

and

$$F_{16} = (q_{14}' = q_{14:1} \oplus q_{4:3} \oplus q_{7:3} \oplus q_{8:3}$$
$$\oplus (q_{13:2} + q_{6:3}))$$

Finally, we can write expressions for the side-by-side listing A_5 and the object program A_4.

$$A_5 = A_{12}^{(2)} \cup A_{12}^{(3)} \cup A_{14}^{(1)} \cup A_{18}^{(2)} \cup A_{18}^{(5)}$$
$$\cup A_{19} \cup A_{20} \cup A_{21} \cup A_{22} \quad (18)$$
$$A_4 = F_{17}(q_{14:1} \neq \Omega, A_5) \quad (19)$$

where

$$F_{17} = (q_i' = \Omega, i = 1, 2, \ldots, 11, 15)$$

APPENDIX II: Summary of Notation

Symbol	*Meaning*
A_i	ith area
A'_i	complement of A_i with respect to property space
$A_n^{(i)}$	ith subset formed in a partition of A_n
B_i	ith bundle
$I'_i(B_j)$	complement of intersection of B_j with ith area in area set
F_i	ith function of a bundle
F_I	identity function of a bundle
$F_n^{(i)}$	ith function of a bundle in a class of functions of a bundle
$G(g_i, A_n)\|_{g_i = c}$	glump element which is subset of A_n consisting of points for which glumping function g_i equals c
H_i	ith function of a glump
q_i	ith property
$q_{i:j}$	ith property in jth area of designated area set. (This notation differs from that in the Information Algebra report insofar as the subscript meanings are interchanged. We consider the revised notation more convenient.)
$q_{i:j}^{(k)}$	kth component of $q_{i:j}$, where the value of $q_{i:j}$ is a vector
q'_i	ith property, with a newly assigned value
$\max\{q_i\}$	maximum value of q_i in designated area
$q_j(\max\{q_i\})$	q_j for the point with maximum q_i in designated area
V_i	property value set of ith property
$\cup$	point set union
$\bigcup_{i=1}^{m} A_n^{(i)}$	point set union of areas $A_n^{(1)}, A_n^{(2)}, \ldots, A_n^{(m)}$
$\cap$	point set intersection
$\wedge$	logical AND
ϵ	belongs to
¢	concatenation operator
$\oplus$	OR operator
$f_1 \leftarrow f_2 \rightarrow f_3$	if f_2 true, take value of f_1; otherwise take value of f_3
"b"	"blank"
"E"	"error"
Ω	irrelevant property value

REFERENCE

1. An Information Algebra. Phase I Report—Language Structure Group of the CODASYL Development Committee. *Communications of the ACM*, April, 1962.

SIMULATION OF AN ASSEMBLY OF SIMPLIFIED NERVE CELLS ON A DIGITAL COMPUTER

R. E. Sears and S. M. Khanna
IBM Federal Systems Division
Bethesda, Maryland

INTRODUCTION

A digital computer program simulating an assembly of simplified nerve cell models has been developed for an IBM 709 Data Processing System. The design of the program and the experiments performed with it were undertaken in order to develop techniques for simulating large assemblies of cells, and to study the network response when selected cell parameters are varied. This paper describes the computer program and some of the experimental results obtained from the program.

The Unit of the Network—Simplified Cell Model

The cell model used in the experiment is illustrated in Fig. 1. Impulses arrive at a given cell from other cells or input wires via the synapses. The effect of each impulse is summed in the cell, and this summed synaptic effect decays with time at a rate determined by the decay constant of the cell, Fig. 2. The accumulated synaptic influence in a cell is compared to the current threshold value for that cell. If the threshold value is exceeded, the cell fires and sends impulses to all its connecting wires. After firing, the cell goes into a refractory period, during which the cell is cleared of all accumulated synaptic effect, and its associated threshold is raised to a high value. The threshold returns to its normal value over a period of time determined by the cell refractory period constant, Fig. 3.

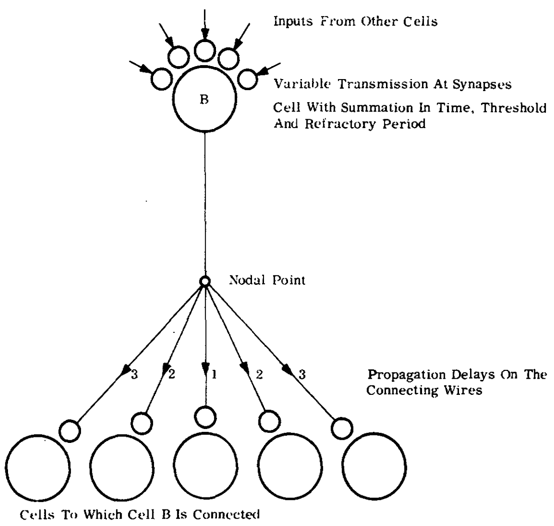

Inputs From Other Cells
Variable Transmission At Synapses
B
Cell With Summation In Time, Threshold And Refractory Period
Nodal Point
3
2
1
2
3
Propagation Delays On The Connecting Wires
Cells To Which Cell B Is Connected

Figure 1. Cell Model Used in the Simulation, Showing Inputs to Cell (B) and Outputs from Cell (B). (For Simplicity Only Five of the Eleven Interconnections Are Shown.)

Communications between the cells in successive layers is via the synapses. The synapses used in the program have an initially minimum transmission value. Each successive impulse arriving at a synapse increases its transmission until a maximum value is reached.

Network Organization and Properties

The network organization used in the simulation is illustrated, in simplified form, in Fig.

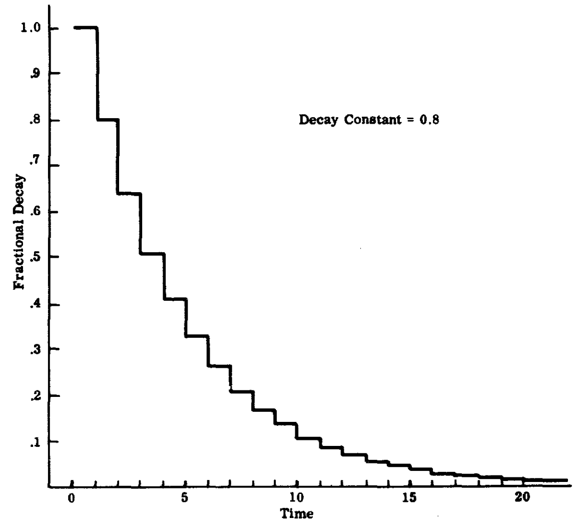


Figure 2. Simulated Cell Decay—Fractional Change of Cell Contents with Time.

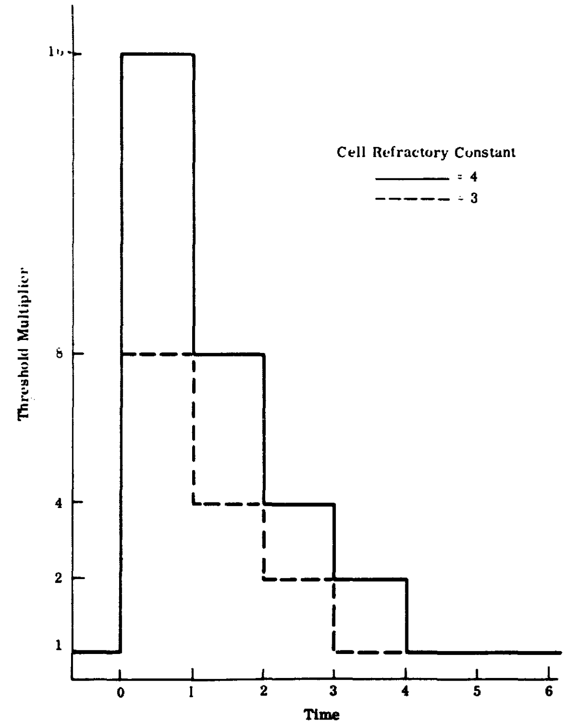


Figure 3. Simulated Refractory Period—Change in Cell Threshold with Time After Cell Has Fired.

4. Only symmetrically connected networks have been used with the symmetry and the layer arrangement selected primarily to simplify the experiments. Each cell in any layer connects to and interacts with several adjacent cells in the next lower layer. The cells in the last layer connect to the output wires.

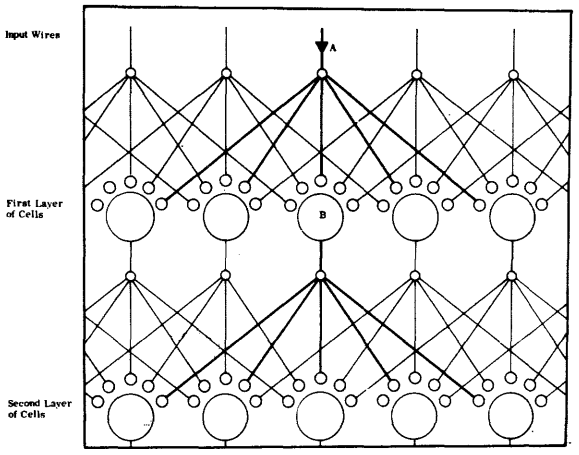


Figure 4. A Part of the Network Showing Interconnections of an Input Wire (A) and of a Cell (B). (For Simplicity Only Five of the Eleven Interconnections Are Shown.)

Impulses arriving on the input wires affect the cells in the first layer, first eliciting no response from the cells. Then, as the synaptic transmission increases in strength, some cells in the first layer fire. As the excitation is repeated, the stimulus affects layers deeper and deeper in the network and, in time, the output wires. Thus, stimulus repetition creates a path of activity in the network which starts in the first layer, goes towards the lower layers, and ends at the output wires.

The network used by the program consists of 80 input wires, 800 cells in 10 layers of 80 cells each, and 80 output wires. Each cell receives impulses from 11 synapses, making a total of 8,800 synapses in the network.

Basis of Simulation—"Simulated Clock"

In actual nerve nets, excitation sequences are essentially parallel events; many cells may be excited at any one time. Since a digital computer is a sequential machine, parallel operations must be transformed into a series of sequential operations that occur during some artificial unit of time. Therefore, the simulation program is based around the concept of a unit clock-step. All time-dependent functions, cell decay, cell refraction, and the connecting wire delays, are represented in terms of clock-steps. The computer is then able to perform parallel operations by stopping the clock at each time step. The actual duration of any clock-step is variable and is dependent on the number of cells fired, and the number of impulses in the network. After all the necessary operations for a unit clock-step are completed, the clock is stepped to a new value (Fig. 5). All time-dependent functions are then adjusted so that they correspond to the new time value. Because the clock-step is a quantized unit of time, the time-dependent functions are represented as step functions rather than as continuous functions.

Principles of Simulating Cell Functions

The principles of simulating the cell functions are illustrated in Fig. 5. Impulses arrive at the synapses at different times, due to the propagation delays of the connecting wires. In any one clock-step, only one impulse arrives at any one synapse and this results in synaptic transmission to the cell to which the synapse is connected. The value transmitted is dependent on the history of excitation of that synapse. Initially, the transmission value in each synapse register is set to a specified minimum. Each impulse arriving at the synapse increases the transmission by adding a specified number to the value in the synapse register, but a limit is placed on the maximum transmission value beyond which no increase takes place.

In any one clock-step, a cell can be affected by up to 11 synapses. Synaptic effect is summed linearly to the current contents in the cell, and this sum is stored in the cell register. At each clock-step, the contents in the cell register are compared with the contents of that cell's threshold register, which contains the current firing threshold for the cell. If the contents in

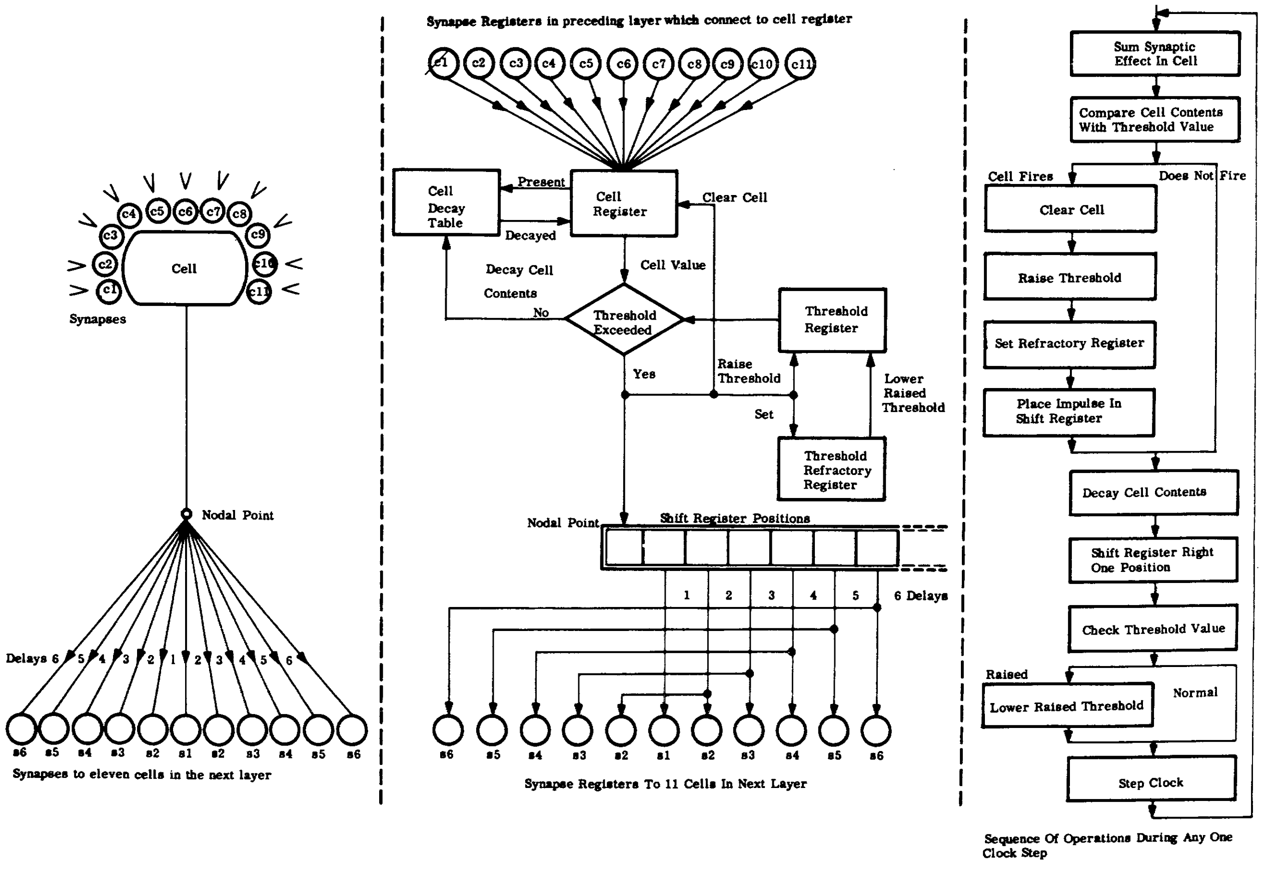

Synapses
Cell
Nodal Point
Delays 6 5 4 3 2 1 1 2 3 4 5 6
s6 s5 s4 s3 s2 s1 s2 s3 s4 s5 s6
Synapses to eleven cells in the next layer
Synapse Registers in preceding layer which connect to cell register
Cell Decay Table
Present
Decayed
Cell Register
Clear Cell
Decay Cell Contents
Cell Value
No
Threshold Exceeded
Yes
Threshold Register
Raise Threshold
Set
Lower Raised Threshold
Threshold Refractory Register
Nodal Point
Shift Register Positions
6 Delays
Synapse Registers To 11 Cells In Next Layer
Sum Synaptic Effect In Cell
Compare Cell Contents With Threshold Value
Cell Fires
Does Not Fire
Clear Cell
Raise Threshold
Set Refractory Register
Place Impulse In Shift Register
Decay Cell Contents
Shift Register Right One Position
Check Threshold Value
Raised
Normal
Lower Raised Threshold
Step Clock
Sequence Of Operations During Any One Clock Step

Figure 5. Block Diagram Showing the Principles Used In Simulating the Cell Functions.

the cell register are less than the cell firing threshold, the cell does not fire and its contents are decayed by replacing them with a smaller number obtained from the cell decay table by a table lookup operation. If the contents in the cell register exceed the cell firing threshold, the cell fires and several operations are performed:

- The cell register is cleared.
- The cell firing threshold is raised by placing a large predetermined value in the threshold register. This value is decreased on each succeeding clock-step until it reaches its normal value, thereby simulating the cell refractory period. During this period of time, the value of the cell threshold is determined by the contents of the threshold refractory register.
- A "one" impulse is placed in the first position of the shift register. The shift register is shifted right one position at each clock-step. Thus, the "one" impulse appears at different positions of the shift register during succeeding clock-steps, thereby simulating the delay of the connecting wires. Connected to each position of the shift register, except the first position, are the synapse registers which contain a synaptic transmission value and the address of the cell that receives the synaptic value. Impulses are sent to the synapse register if the "one" impulse appears in the position to which they are connected.

The program consists of six main routines illustrated in simplified flow chart form in Figs. 6 through 11.

Input to the Program

The input to the program consists of the experimental parameters and the stimulus. (The experimental parameters are discussed in the next section.) The stimulus, an input pattern in punched card format simulating the output of sensors, is specified by defining the input wires to be excited and the sequence of excitation. Stimuli characteristics, such as amplitude distribution, time duration, time sequences, etc., are represented by proper placement of impulses (punches) in the input pattern. The

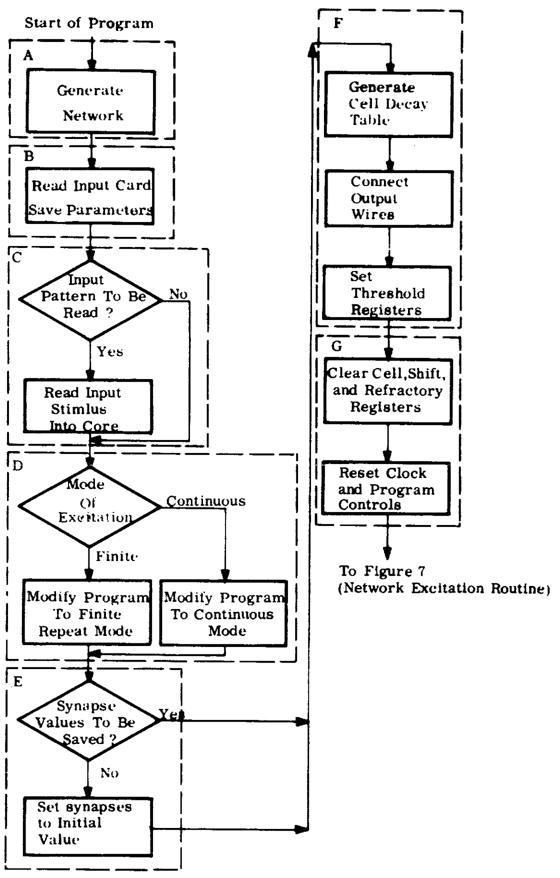

Start of Program
A
Generate Network
B
Read Input Card Save Parameters
C
Input Pattern To Be Read ?
No
Yes
Read Input Stimulus Into Core
D
Mode Of Excitation
Continuous
Finite
Modify Program To Finite Repeat Mode
Modify Program To Continuous Mode
E
Synapse Values To Be Saved ?
Yes
No
Set synapses to Initial Value
F
Generate Cell Decay Table
Connect Output Wires
Set Threshold Registers
G
Clear Cell, Shift, and Refractory Registers
Reset Clock and Program Controls
To Figure 7 (Network Excitation Routine)

Figure 6. Simulation Program Initializing Routine.

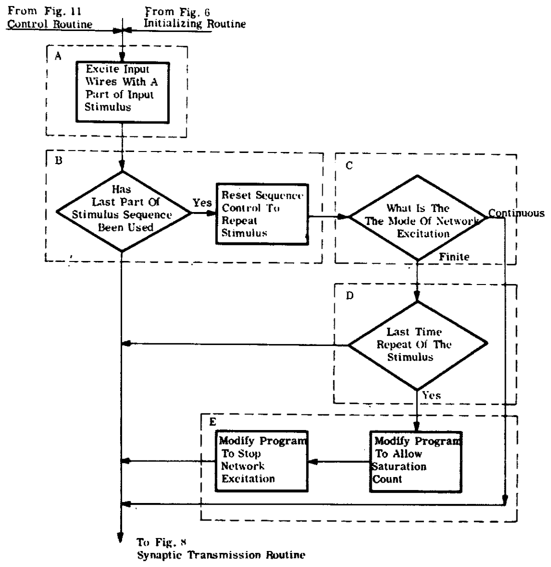

From Fig. 11 Control Routine
From Fig. 6 Initializing Routine
A
Excite Input Wires With A Part of Input Stimulus
B
Has Last Part Of Stimulus Sequence Been Used
Yes
Reset Sequence Control To Repeat Stimulus
C
What Is The The Mode Of Network Excitation
Continuous
Finite
D
Last Time Repeat Of The Stimulus
Yes
E
Modify Program To Stop Network Excitation
Modify Program To Allow Saturation Count
To Fig. 8 Synaptic Transmission Routine

Figure 7. Simulation Program Network Excitation Routine.

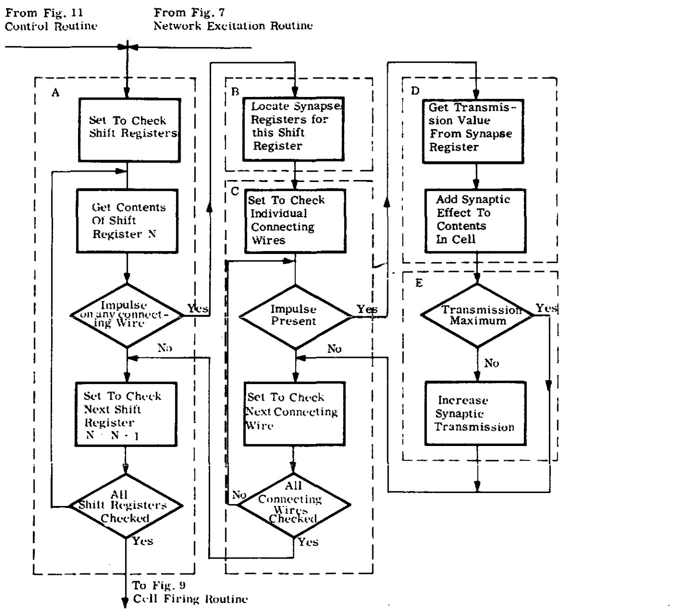

From Fig. 11
Control Routine
From Fig. 7
Network Excitation Routine
A
Set To Check Shift Registers
Get Contents Of Shift Register N
Impulse on any connecting Wire
Yes
No
Set To Check Next Shift Register N = N + 1
All Shift Registers Checked
Yes
B
Locate Synapse Registers for this Shift Register
C
Set To Check Individual Connecting Wires
Impulse Present
Yes
No
Set To Check Next Connecting Wire
All Connecting Wires Checked
No
Yes
D
Get Transmission Value From Synapse Register
Add Synaptic Effect To Contents In Cell
E
Transmission Maximum
Yes
No
Increase Synaptic Transmission
To Fig. 9
Cell Firing Routine

Figure 8. Simulation Program Synaptic Transmission Routine.

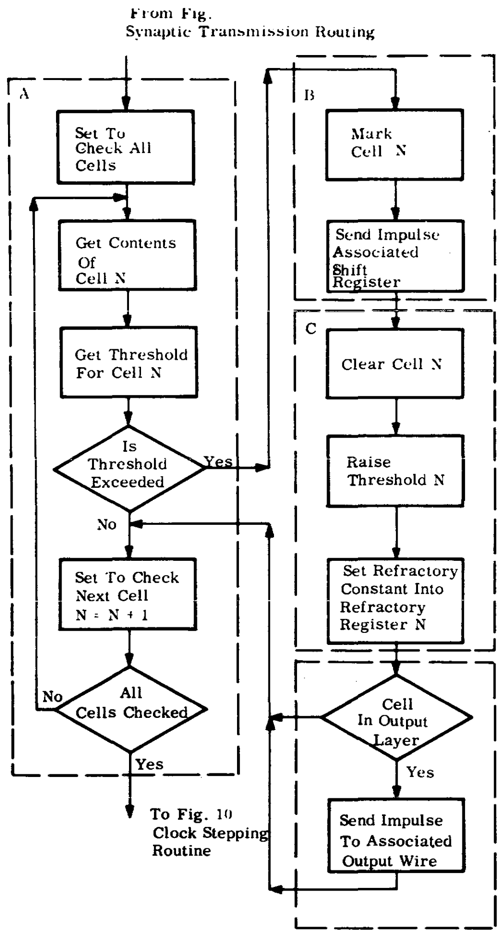


Figure 9. Simulation Program Cell Firing Routine.

stimulus is applied to the network in one of two selectable modes: finite and continuous. The finite mode repeats the stimulus a specified number of times. The continuous mode repeats the stimulus until the network saturates. The network is considered to be saturated when the total number of fired cells has not increased in 10 consecutive clock-steps. The stimulus used by the simulation program consisted of a maximum sequence of 36 parts, each part corresponding to a clock-step. There may be a maximum of 70 input wires excited in each part. Stimulus sequences longer than 36 are obtained by repeating the stimulus.

Output of the Program

The output of the simulation program is a printout consisting of four parts:

- The parameters used in the experiment are printed in column format with the parameter name at the top of each column, Fig. 12. The number in each column is the value used in the experiment. The experimental parameters consist of:

  A. RUN—indicates the experiment series number.

  B. CELLS

  (i) THRES—indicates the threshold setting at which a cell fires.

  (ii) DECAY—an integer value indicates the number of clock-steps that elapse before the contents of a cell decay to one-half their value. A decimal value indicates the fraction by which the contents of a cell decay at each clock-step. (During the course of the experiments the method of simulating cell decay was modified to provide a smoother cell decay action. The method of one-half decay per specified number of clock-steps was removed and the rate method was installed by using a cell decay table.)

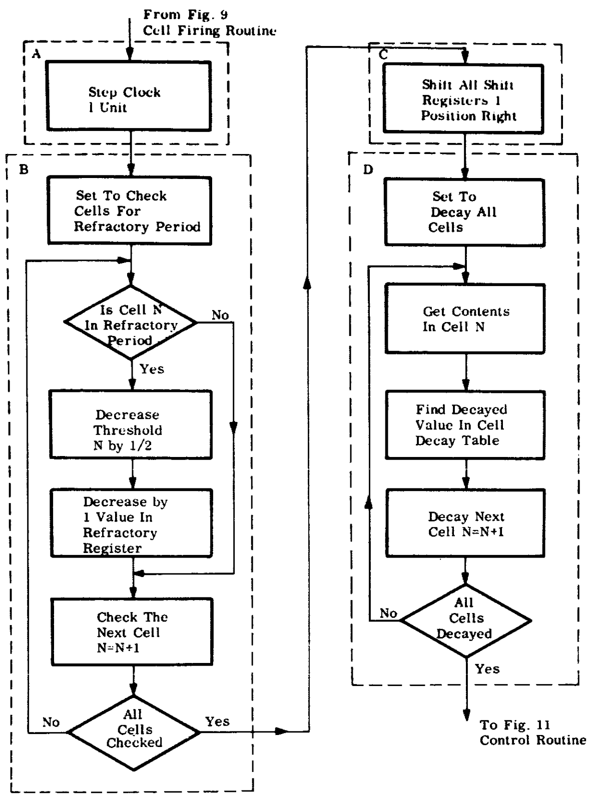

From Fig. 9
Cell Firing Routine
A
Step Clock 1 Unit
B
Set To Check Cells For Refractory Period
Is Cell N In Refractory Period
No
Yes
Decrease Threshold N by 1/2
Decrease by 1 Value In Refractory Register
Check The Next Cell N=N+1
All Cells Checked
No
Yes
C
Shift All Shift Registers 1 Position Right
D
Set To Decay All Cells
Get Contents In Cell N
Find Decayed Value In Cell Decay Table
Decay Next Cell N=N+1
All Cells Decayed
No
Yes
To Fig. 11
Control Routine

Figure 10. Simulation Program Clock-stepping Routine.

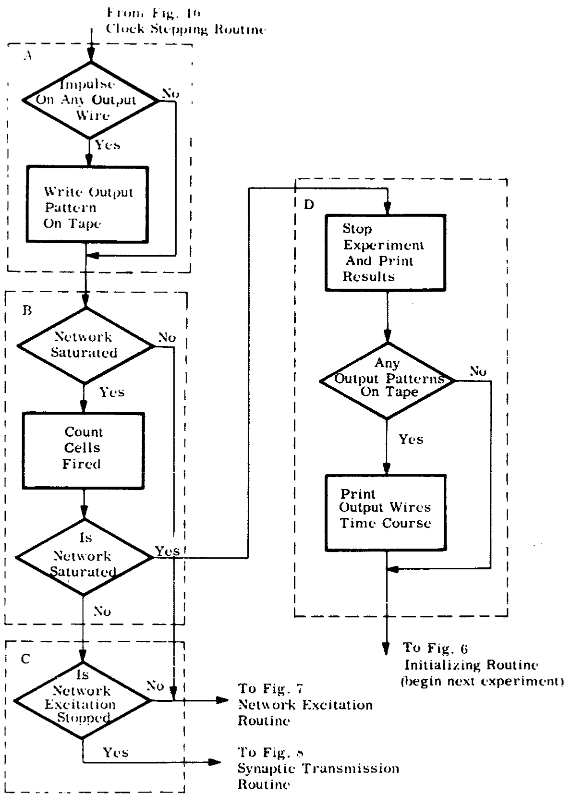

From Fig. 10
Clock Stepping Routine
A
Impulse On Any Output Wire
No
Yes
Write Output Pattern On Tape
B
Network Saturated
No
Yes
Count Cells Fired
Is Network Saturated
Yes
No
C
Is Network Excitation Stopped
No
Yes
To Fig. 7
Network Excitation Routine
To Fig. 8
Synaptic Transmission Routine
D
Stop Experiment And Print Results
Any Output Patterns On Tape
No
Yes
Print Output Wires Time Course
To Fig. 6
Initializing Routine
(begin next experiment)

Figure 11. Simulation Program Control Routine.

C. SYNAPSE VALUES—
(i) INITIAL—indicates the initial transmission value of the synapse.
(ii) MAX—indicates the maximum transmission value.
(iii) DELTA—indicates the increment in synaptic transmission per impulse.
(iv) SAVE—indicates by YES or NO whether a previous experiment's synaptic values are saved or not.

D. THRESHOLD RISE 2 EXP—indicates the number (2^n form) with which the cell threshold is multiplied at the beginning of the cell refractory period. This number also indicates the time duration of the cell refractory period in clock-steps.

E. INPUT REPEATS—indicates the number of times the stimulus is to be repeated.

F. CLOCK-STEPS—indicates the number of clock-steps during which the experimental record was made.

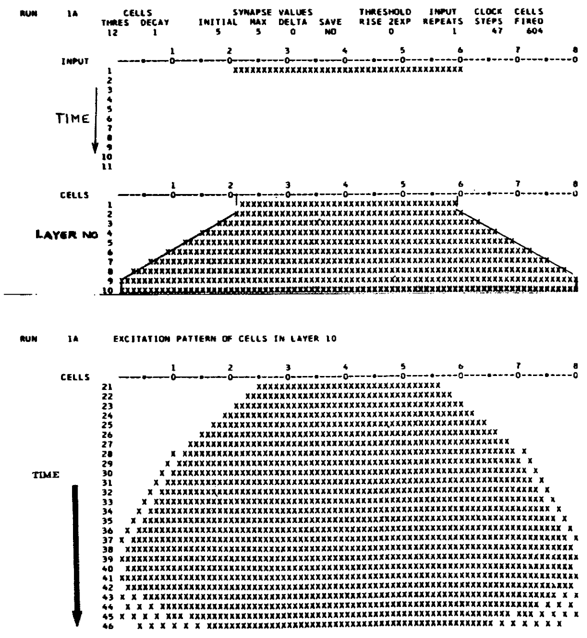


Figure 12. Computer Printout Showing Divergent Response with No Cell Refractory Period.

G. CELLS FIRED—indicates the number of cells that fired at least once during the experiment.

- The stimulus printout presents in pictorial form the stimulus used in the experiment. The horizontal heading corresponds to the 80 input wires. The vertical heading corresponds to succeeding clock-steps, 1 to 11. An X indicates the presence of an impulse. For example, in Fig. 12, the stimulus consists of 11 active parts which operate in the following manner: At clock-step 1, input wires 21 to 60 receive impulses. From clock-steps 2 to 11, the input wires do not receive impulses. At clock-step 12, input wires 21 and 60 will again receive impulses if the stimulus is to be repeated. Thus, this stimulus indicates that the input wires will be excited every 12th clock-step.
- The cells fired during the experiment printout presents in pictorial form the cell activity within the network. The horizontal heading, 1 to 80, corresponds to the cell numbers. The vertical heading, 1 to 10, corresponds to the layer numbers. An X indicates that a cell has fired at least once during the course of the experiment.
- The time course of excitation of output wires printout presents in pictorial form the output wire excitations as a function of time. The horizontal heading corresponds to the 80 output wires. The vertical heading corresponds to succeeding clock-steps, 21 to 46 in Fig. 12. An X indicates the presence of excitation on an output wire.

Experiments

The purpose of the experiments was to investigate the responses of the network to different kinds of inputs, and to observe the effect of changing selected experimental parameters on the response. The responses obtained may be divided into two very general classes defined as: the divergent response, in which the total number of fired cells is more at the output layer compared to the input layer; and the convergent response, in which the number of fired cells is less at the output layer. Figures 12, 13, and 14 show examples of divergent activity while Figs. 15 through 22 show examples of convergent activity.

In experiments with divergent response the effect of simultaneously exciting 40 input wires

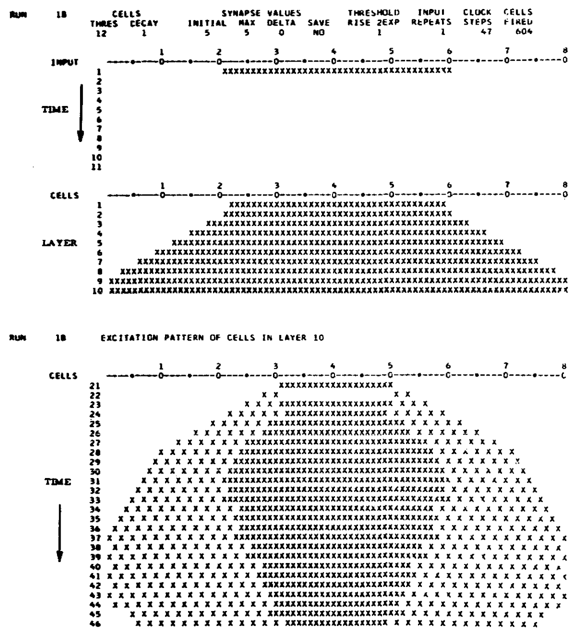


Figure 13. Computer Printout Showing Divergent Response with Cell Refractory Period of 1 Clock-step.

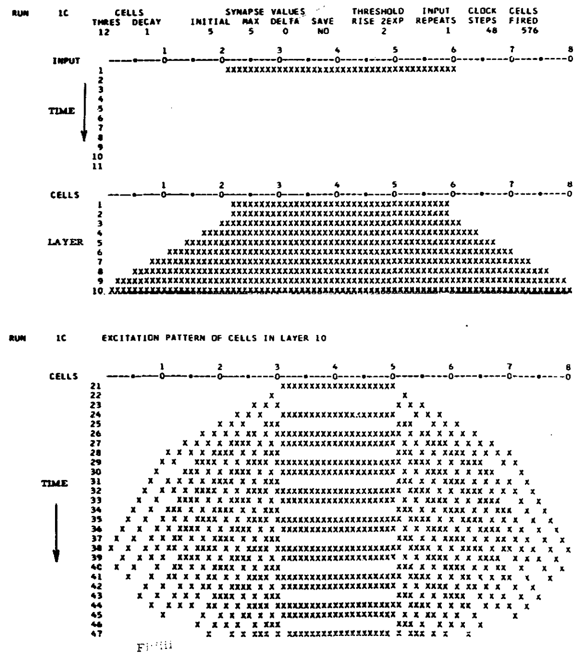


Figure 14. Computer Printout Showing Divergent Response with Cell Refractory Period of 2 Clock-steps.

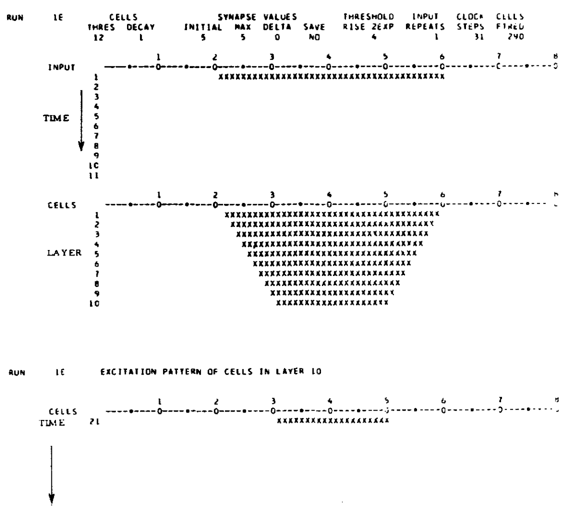


Figure 15. Computer Printout Showing Convergent Response with Cell Refractory Period of 4 Clock-steps.

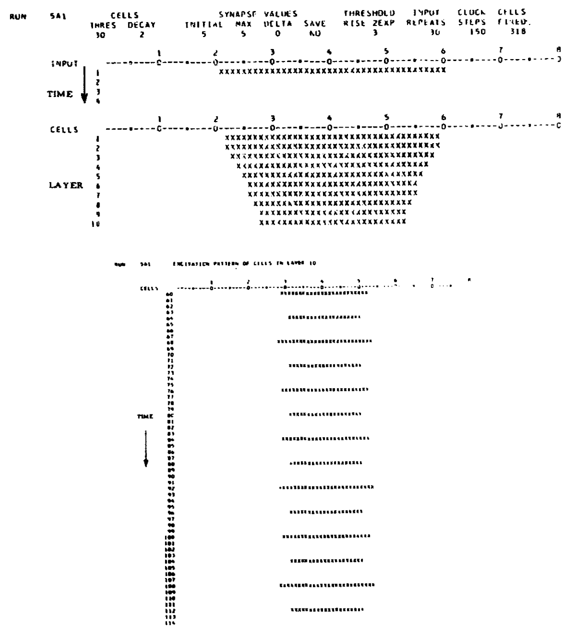


Figure 16. Computer Printout Showing Response to Repetitive Input with 40 Input Wires Excited.

on the output response is shown in Figs. 12 through 14. The variable used in this set of experiments is the cell refractory period. An input is applied only once at clock-step 1 and an output response first appears at clock-step 21. In Fig. 12, the output response lasts for a period over 20 clock-steps before ultimately dying out. As the duration of the refractory

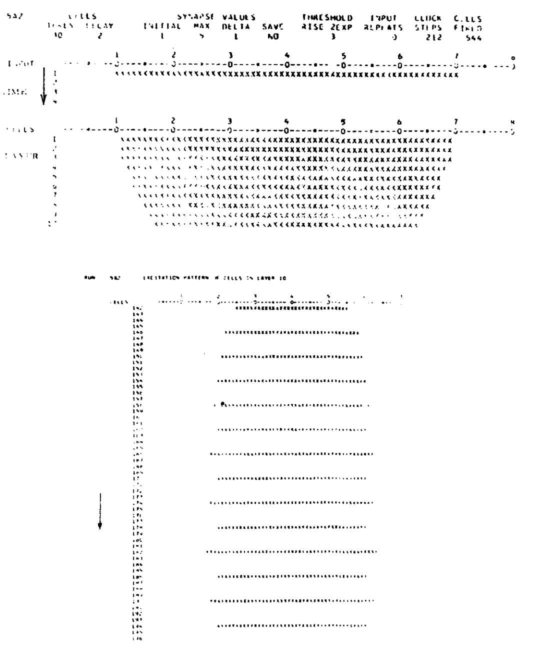


Figure 17. Computer Printout Showing Response to Repetitive Input with 61 Input Wires Excited and Cell Firing Threshold Set at 30.

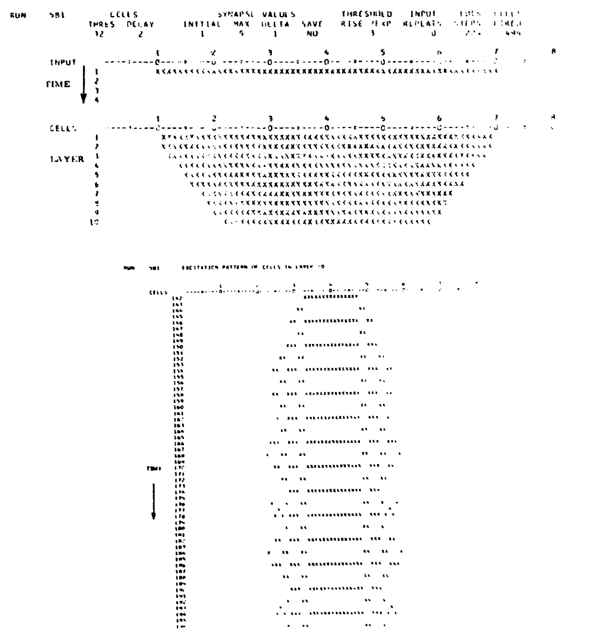


Figure 18. Computer Printout Showing Response to Repetitive Input with 61 Input Wires Excited and Cell Firing Threshold Set at 32.

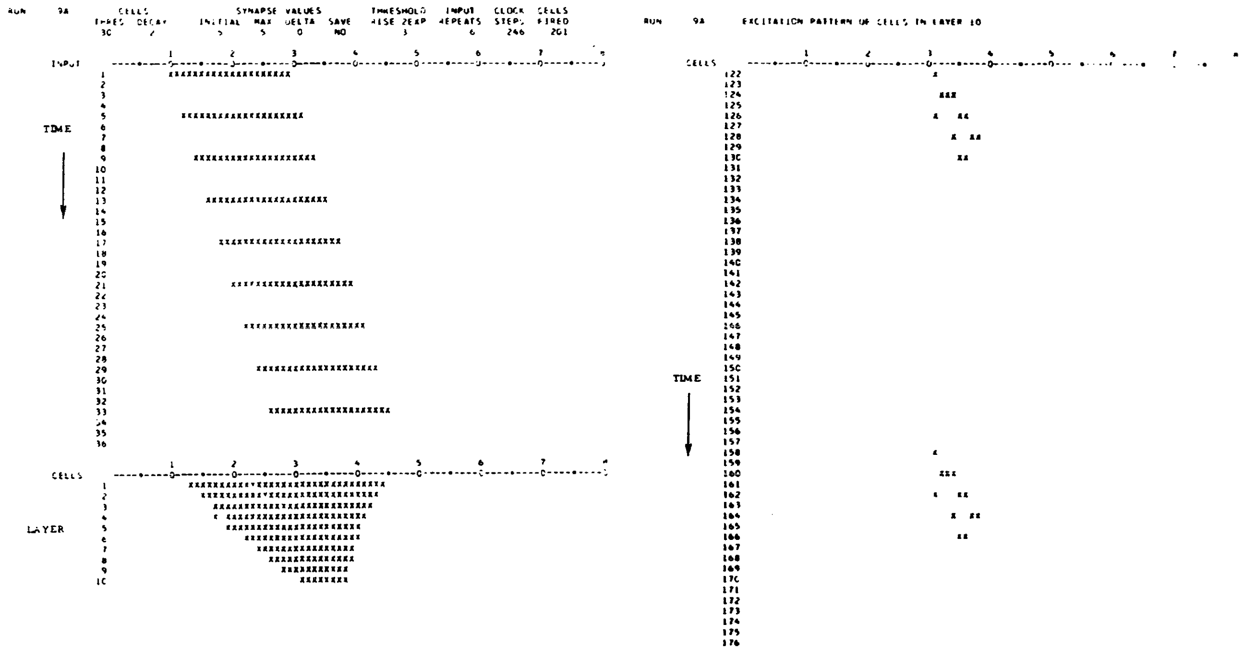


Figure 19. Computer Printout Showing Response to Moving Repetitive Input with 20 Input Wires Excited.

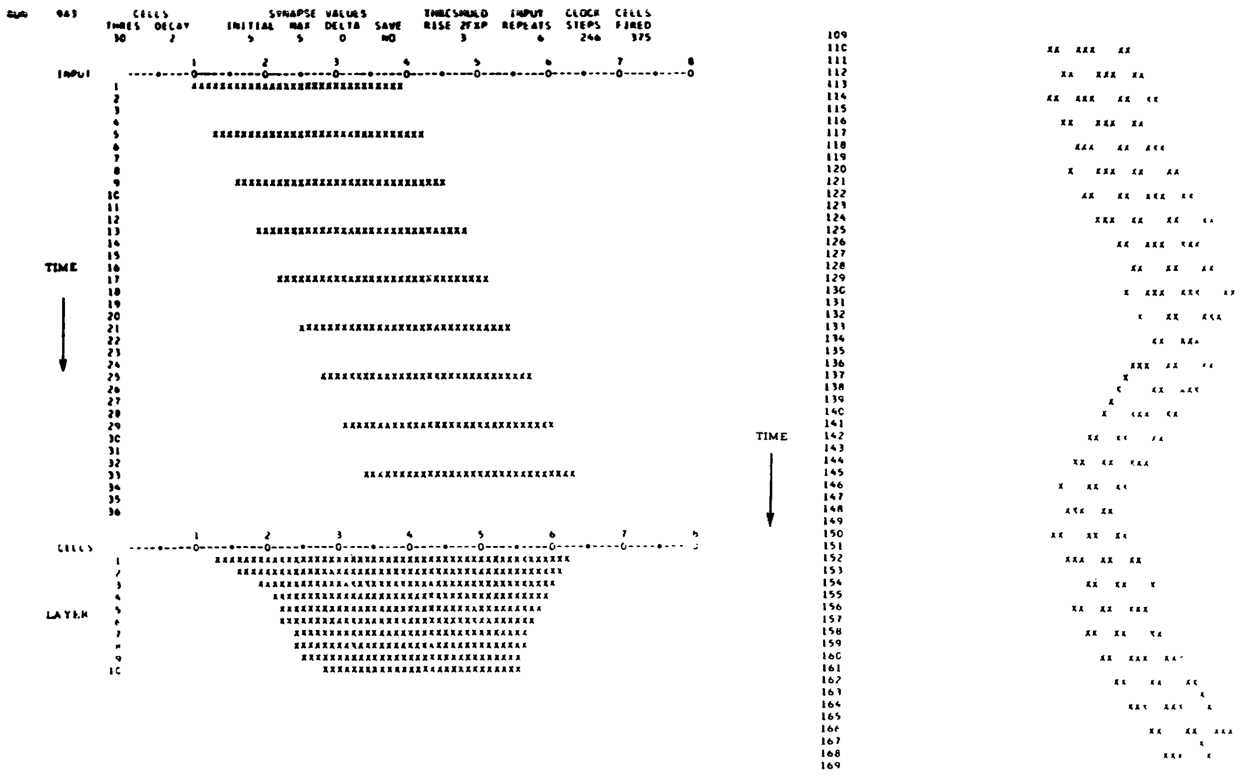


Figure 20. Computer Printout Showing Response to Moving Repetitive Input with 40 Input Wires Excited.

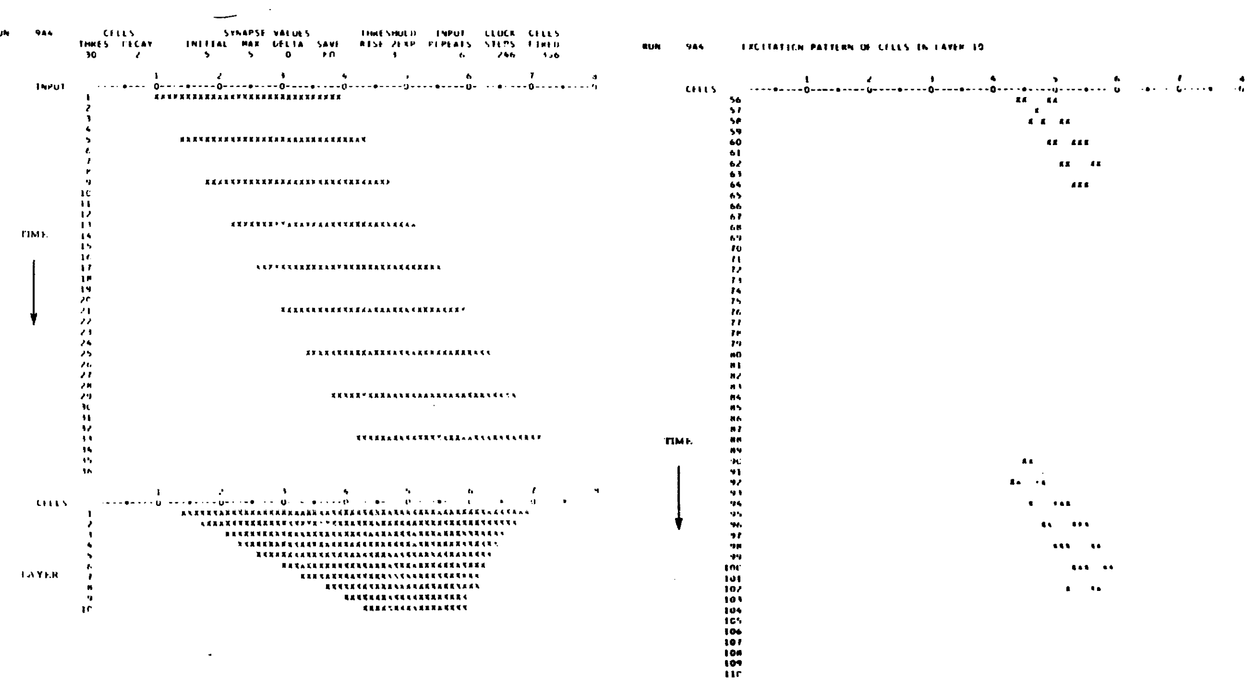


Figure 21. Computer Printout Showing Response to Moving Repetitive Input with 40 Input Wires Excited.

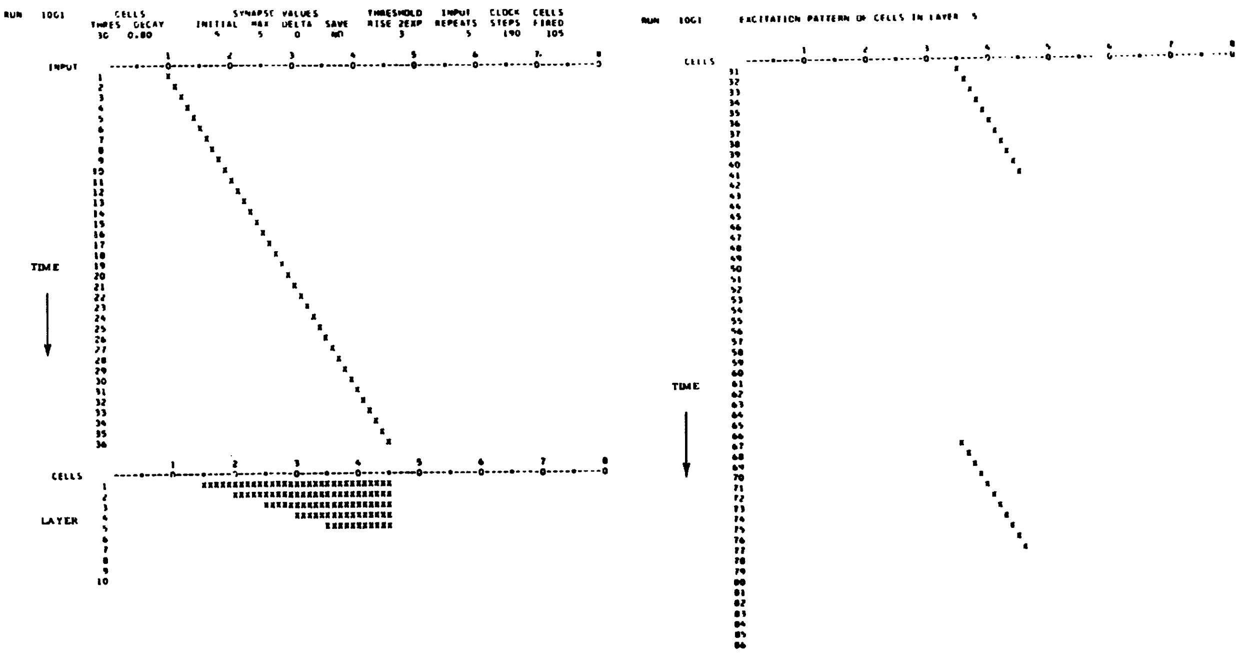


Figure 22. Computer Printout Showing Response to Moving Repetitive Input with a Single Input Wire Excited at Each Clock-step.

period is increased, the interval between the cell firing increases, Figs. 12 to 14, until in Fig. 15, no spreading in output response is observed. This output activity at first sight may appear to be "reverberatory" (in which also a single excitation leads to multiple responses). However, the multiple responses in these experiments are not caused by any positive feedback, but are due to the large variety of transmission paths and delays.

In experiments with convergent response the effect of the time-dependent properties of the input was investigated. The response of the network when exposed to a repetitive input is shown in Fig. 16. The output is a series of responses repeating at the input repetition rate.

The effect of varying the cell threshold on the response is shown in Figs. 17 and 18. The response pattern in Fig. 18 is spread over a period of time. This comparison illustrates that, to maintain simple stimulus response relations (of the type shown in Fig. 17), the parameter selection is fairly critical. Once these parameters are found, a simple relation between the input and output is observed for a variety of input patterns and their repetition rate. The problem of the critical parameters is largely due to the small number of interconnections between cells (eleven in this series), and is virtually eliminated in the later experiments by increasing the connections.

When the input sequence of excitation is sequential as shown in Figs. 19, 20, 21 and 22, a similar sequential output is obtained. A comparison of Figs. 20 and 21 shows that above a given speed of the input stimulus movement the output appears to be continuous.

SUMMARY AND CONCLUSIONS

This paper describes some preliminary techniques used in the simulation of an assembly of 800 cells on an IBM 709 computer. In the present experiments approximately $\frac{3}{4}$ of the core storage was used. These and later experiments with a new program indicate that the total number of simulated cells is limited to about a thousand cells, when the cells have all the properties described in this paper. The simulated running time (average) per input pattern was approximately two and a half minutes. To achieve this speed, all multiplication, division, or more complicated functions had to be replaced by the speedier table lookup procedures. The experimental parameters, input stimuli, and the network responses obtained in some of the experiments are shown. Emphasis is placed on responses under non-reverberatory conditions which are characteristic of the peripheral nervous system.

As a first step toward making a model of the peripheral processing in the nervous system, it was necessary to determine the problem involved in simulating assemblies of simplified models of nerve cells on a digital computer. The techniques reported are in their infancy and require a great deal of work to improve the accuracy of description and to include more properties of the biological cells. It is encouraging, however, to be able to simulate as many as a thousand cells reasonably accurately for studying models of peripheral processing.

ACKNOWLEDGEMENT

The authors wish to acknowledge the support of Dr. H. P. McCartney in preparing this paper.

REFERENCES

1. BRAZIER, M. A. B. (1960), *The Electrical Activity of the Nervous System,* MacMillan Co., N. Y.
2. FARLEY, B. G., and W. A. CLARK, JR. (1961), "Activity in Networks of Neuron-like Elements," *Proceedings of the Fourth London Conference on Information Theory,* Ed. C. Cherry, Butterworths, London.
3. KHANNA, S. M., and C. R. NOBACK (1963), *Neural Networks and Artificial Intelligence,* IEEE Publications S-142.
4. ROCHESTER, N., J. H. HOLLAND, L. H. HAIBT, and W. L. DUDA (1956), *Tests on a Cell Assembly Theory of the Action of the Brain Using a Large Digital Computer,* IRE Trans. Inf. Theory, IT2-3, 80-93.
5. TASAKI, I. (1953), *Nervous Transmission,* C. Thomas, Springfield, Illinois.
6. TASAKI, I. (1959), Conduction of the Nerve Impulse," *Handbook of Neurophysiology,* Vol. I, pp. 75-121.

CYCLOPS-1: A SECOND-GENERATION RECOGNITION SYSTEM*

T. Marrill,† *A. K. Hartley,*† *T. G. Evans,*‡ *B. H. Bloom,*§
D. M. R. Park,§ *T. P. Hart,*§ *and D. L. Darley*†

1. OPERATING THE SYSTEM

CYCLOPS–1 is a recognition system programmed for a general-purpose digital computer, the PDP-1. It uses no special-purpose hardware. The three principal modes of operation of the system are (a) pattern input, (b) input identification, and (c) scene analysis.

(a) *Pattern Input*

The visual pattern to be analyzed by the system is drawn on the face of the computer CRT display scope by means of a light pen. This mode of input is particularly well suited to a system, such as the present one, whose aim is to demonstrate, and experiment with, principles of pattern recognition. In a production system, the emphasis will turn to the scanning of hard copy or of photographic transparencies. CYCLOPS-1 has been designed with this aim in mind. This means, in particular, that the *temporal* information, i.e., the information about the order in which the elements of the pattern are drawn, is *not* used by the system in any way (See Section 2).

The light pen is shown in Fig. 1. Its use is as follows. As soon as the system has been put into pattern input mode (this is done by typing a command on an on-line typewriter), a raster of points is displayed on the scope. If the tip of the light pen is now put on or near the surface of the scope, the raster vanishes (the program has found the light pen), and a little cross is displayed opposite the tip of the pen. If the pen is moved with the tip touching the surface of the scope, the cross remains under the tip. If the pen is withdrawn from the surface of the scope, the raster will reappear and the pen may be repositioned.

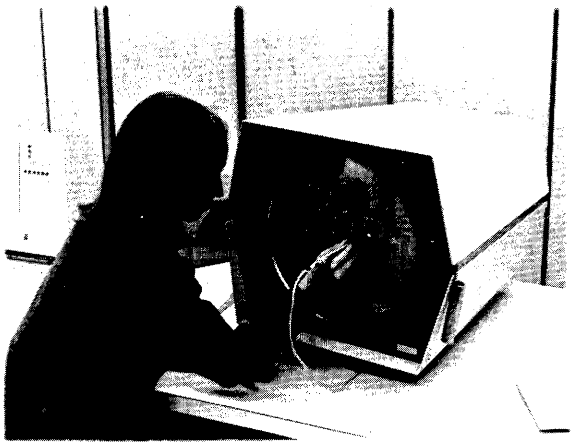


Figure 1. Pattern being drawn on scope.

As the operator holds the light pen, his index finger rests on a small switch. When the tip of the pen is on the scope, the operator may *write,* i.e., leave a trace behind on the scope, by push-

*This work was supported by Air Force Cambridge Research Laboratories under Contract AF 19(628)–227.

†Bolt Beranek and Newman Inc., Cambridge, Massachusetts.
‡Bolt Beranek and Newman Inc., and Massachusetts Institute of Technology. [Now at Air Force Cambridge Research Laboratories.]
§Bolt Beranek and Newman Inc., and Massachusetts Institute of Technology.

ing this penswitch (see Fig. 1). No trace is left when the penswitch is released. Anyone can learn to write or draw with the pen in a few minutes.

The pen-tracking program determines the position of the pen every 20 milliseconds. When the operator writes by pushing the penswitch, the successive pen locations are entered into an "input table" and remain displayed on the scope. (This table represents the input to the recognition part of the system.) Pen locations are given as (x, y) coordinates, each coordinate having an accuracy of 10 bits. The patterns are thus drawn on a matrix containing slightly over 1,000,000 cells.

(b) *Input Identification*

The program may be commanded from the typewriter to assign the pattern currently on the scope to one of 36 categories (26 letters and 10 digits). The letter or digit is typed out on the typewriter. If the program is unable to perform the identification, a question mark is typed out.

Examples of characters recognized by the system are given in Figs. 2 and Fig. 3. In Fig. 2 we see the same character in different positions, size, slant, and style. Four characters differing in identity are shown in Fig. 3.

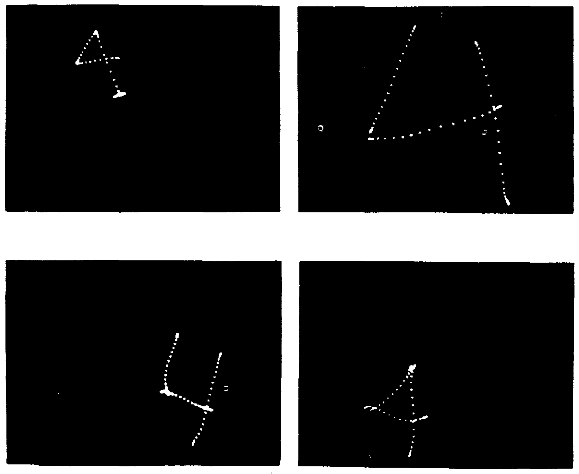


Figure 2. Four examples of the numeral 4 recognized by the system.

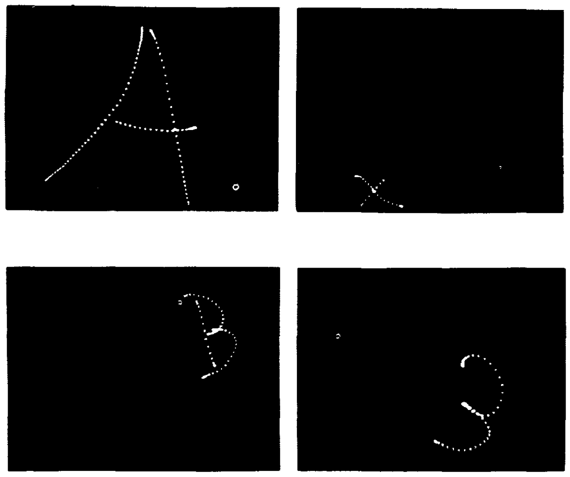


Figure 3. Four characters differing in identity recognized by the system.

Input identification proceeds in two stages: first, a line-forming stage, in which the individual points are formed into line segments; second, the identification itself. These two stages are discussed at greater length in Sections 2 and 3, below.

(c) *Scene Analysis*

The program may be commanded from the typewriter to analyze a complex scene which has been drawn on the scope. Such a scene may consist of *known items,* i.e., shapes which have been defined as being significant (the alphanumeric characters); and of other items, some of which may be significant to the viewer but not to the system, and some of which may be merely background "noise". An arbitrary number of known items may be present simultaneously; they may be of different sizes and orientations; they may overlap, or be inside of each other; they may be superimposed on an arbitrary backgound.

An example of a scene to be analyzed by CYCLOPS-1 is given in the topmost frame of Fig. 4. The analysis of this scene consists in giving the identity, location, and size of each known item. The output of the program is presented in two ways. On the typewriter, the identity, location, and size of each item are typed out. The location is given by the (x, y) coordinates of the centroid of the item; the size

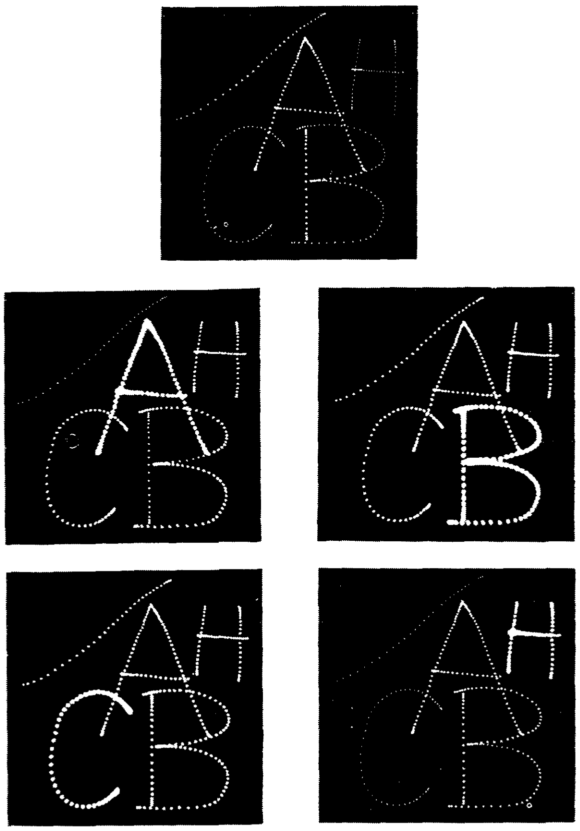


Figure 4. A "scene" analyzed by the system. The input is given in the top frame. The results of the analysis are shown in the bottom four frames (further information is also typed out by the typewriter).

is given as one of three categories: large, medium, and small. (Other descriptions of position and size could readily be implemented.) On the scope, each of the known items which has been found is brightened in turn (see four bottom frames of Fig. 4).

Thus, the scene in Fig. 4 generates the following printout:

```
al        34,140        med
hl       321,243        sml
cl    —228, —180        med
bl      157, —187       med
```

By means of scene analysis the system is capable of finding items which may be difficult to find by eye. Thus, the scene shown in the first frame of Fig. 5 is analyzed as shown in the remaining frames.

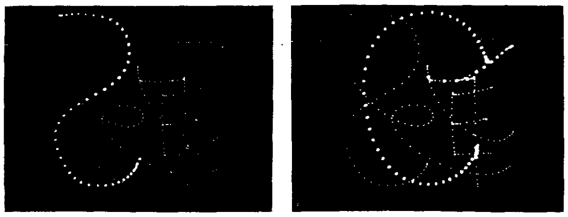


Figure 5. A more difficult "scene". The input is shown in the first frame, the analysis in the remaining frames.

2. LINE-FORMING

The first stage of processing deals with the formation of *line segments.* The input to this stage consists of a table of (x, y) coordinates of points, each point being specified by 20 bits. The system assumes nothing about the order of points in this table. It does not know which points go with which others to form line segments, or, for that matter, whether there are any line segments present, or even which points are near which other points. The table may be randomized prior to the line-forming stage without affecting the output.

The attempt has been made to design the line-forming program so that, in general, its output agrees with what the human eye sees. Thus, in Fig. 6 we see exactly two line segments: one fairly straight segment, and one looped segment. The line-forming program, given this input, will generate the same two line segments.

More formally, what is required of the line-forming program is the following: (a) to as-

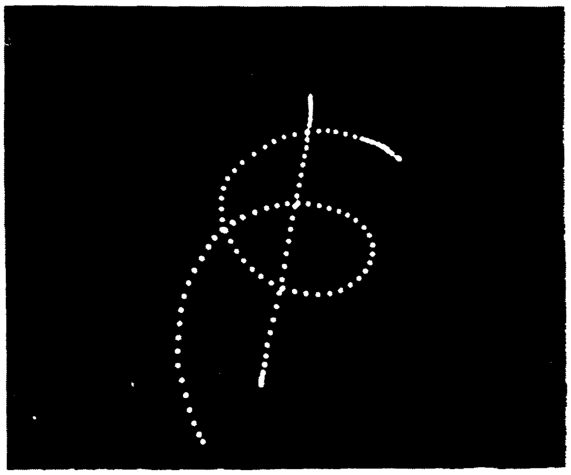


Figure 6. Illustration of line-forming by system (see text).

sign each point in the table to some line segment (where a line segment is a set of points with certain continuity properties) or to none at all; and (b) within each set of points assigned to a line segment, to order the points in the "natural" order, as seen by the eye. (Each line segment allows two natural orderings, depending on the sense in which arc length is taken to increase; the program picks one of the two arbitrarily.)

The formation of line segments proceeds in three stages: triplet formation, triplet chaining and segment connecting. To form a triplet, we select a point and try to find two nearby points which are on opposite sides of, and approximately colinear with, the original point. The chaining process begins after all possible triplets have been formed. Two triplets are joined provided they have two common points in the proper order. For example, the triplets abc and bcd will form the segment abcd. This segment may be extended by combining it with other triplets according to the same rule. When the current segment cannot be extended any further, the process is repeated with the remaining triplets. Chaining terminates when no segment can be extended further.

Quite often the line segments created in the above process will not continue across intersections. For example, the letter X might consist of four segments. We examine the behavior of the segments in the vicinity of the end points, looking for pairs of segments which are reasonably close together, have the same curvature and slope, and which would meet if extended. If two segments are found which satisfy this criterion, they are connected.

3. IDENTIFICATION: HYPOTHESIS GENERATION AND TESTING

The identification process operates on the line segments found in the line-forming stage. Identification may be understood in terms of two subprocesses, hypothesis-generation and hypothesis-testing, as illustrated in Fig. 7.

The process operates as follows: An hypothesis concerning the nature of the input (e.g., that it is the letter Q) is generated for input to the hypothesis-tester. If found acceptable, the hypothesis becomes the output of the system. If found unacceptable, a new hypothesis is formed and tested.

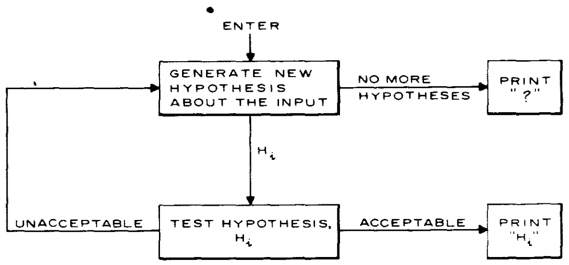


Figure 7. Flow-chart of the identification process (after line-forming has occurred).

In the present system, the hypothesis-generation scheme is entirely straightforward; the hypotheses are simply tested in a predetermined, fixed order.

The hypothesis-testing scheme, on the other hand, is considerably more complex. Consider the following simple example. Suppose the hypothesis-generator generates the hypothesis that the input pattern consists of the digit *one*, (specifically, let us say, that variety of the digit *one* which consists of a single stroke). How might we test this hypothesis? If the input consists of a *one* and of nothing else of significance, then there must be one very predominant line segment (minor ones representing noise may also be present). Let us see if that is the case. If so, we are on the right track. If not, the hypothesis must be rejected. Now, if there is one predominant line segment, it must be, roughly speaking at least, straight. Let us see whether it is. If so, we are still on the right track. If not, we must reject the hypothesis. Now, if our predominant line segment is more or less straight, it must also be approximately vertical. If not, we must reject the hypothesis. If so, we can, with the utmost confidence, accept it. This is precisely the technique used by the program to test the hypothesis in question.

An hypothesis-testing program consists of a series of questions about *characteristics* of particular segments in the pattern, of the relationships between the segments, or of the pattern as a whole. In the hypothesis described above, for example, we asked questions about the following segment characteristics:

1. The *predominance coefficient* of the line segments in the pattern. (This coefficient is defined as the ratio of the arc length of the segment to the total arc length of all segments.)
2. The *straightness coefficient* of a segment. (This coefficient is defined as a ratio of the straight-line distance between the end points of the segment to the arc length of the segment.)
3. The *orientation* of the segment. (This is defined as the angle made with the positive x-axis by the straight line connecting the ends of the segment.)

CYCLOPS-1 has forty-two such segment and pattern characteristics available to the hypothesis. Two more examples are:

4. *Intersections* between segments i and j. (Find the locations of all intersections between line segments i and j. The two segments need not be distinct. Thus, one may ask for the intersections of a line segment with itself.)
5. *Inflection Points.* Find the locations of inflection points on the designated line segment.

The hypothesis-testing programs, written in the vocabulary of characteristics of the pattern, constitute the *definitions* of the items known to CYCLOPS-1. To define a new item, that is, to enlarge the recognition repertoire of the system, it is necessary to do two things: (a) write and add to the system an hypothesis-testing program for the item in question and (b) add the name of this program to the list of hypotheses. These tasks are easily performed by someone familiar with the language. To add a new item, say *square,* to the system would require about two hours for programming and check-out.

The running time of the system is not proportional to the number of hypotheses. There are two reasons for this: first of all, a large portion of the running time is spent in the line-forming stage, and is therefore independent of the number of hypotheses; secondly, certain characteristics requested by the hypothesis-testing programs require a great deal more processing than others; the values of these time-consuming characteristics are remembered rather than recalculated. Thus, if in testing hypothesis H_1, we ask for intersections between segments i and j, we remember the results. Now, assume H_1 is rejected; if in the process of testing another hypothesis H_2 we again ask about intersections between i and j, these intersections are automatically looked up by the intersections program, and are not recalculated. Hence, the incremental time required for testing a new hypothesis may be quite short.

The total time taken to identify an item varies greatly, depending primarily on the number of points involved. The identification of the items shown in Figs. 2 and 3 took between three and twelve seconds each.

4. SCENE ANALYSIS

The process of scene analysis differs only slightly in outline from the process of input identification. The principal difference lies in the hypothesis-testing programs. In the case of input identification, the program hypothesizes that the input, *in toto,* belongs to a certain category, i.e., consists of a known item, with perhaps a small amount of noise added. In the case of scene analysis, the program hypothesizes only that, somewhere within a background of arbitrary nature, there is a given known item. This latter type of hypothesis we have called a *super-hypothesis.* Whereas a hypothesis-testing program merely answers "yes" or "no", a super-hypothesis-testing program answers "yes" or "no", and if "yes", gives the segment numbers of the line segments making up the known item found.

The process of scene analysis proceeds as follows. The first super-hypothesis is tested. If the answer is "yes", the segment numbers of the item are saved, and the line segments in question are withdrawn from further consideration; the same super-hypothesis is then tested again. Only when the answer is "no", do we turn to a new super-hypothesis. When all super-hypotheses or segments have been exhausted, the analysis process ends.

The super-hypotheses are written in the same language as the hypotheses, but are appreciably more complicated. Thus, whereas the hypothesis-testing program for the letter *A* takes half a page of coding, the super-hypothesis-testing program for *A* takes a page and a half. The scheme of this latter program, by way of illustration, is roughly as follows:

1. Look for all approximately straight segments slanting up to the left.
2. Look for all approximately straight segments slanting up to the right.
3. Find pairs of segments, with one member from 1 and one from 2, such that members of a pair are nearly the same length, and have their topmost points close together.
4. Look for more or less straight lines that are approximately horizontal.
5. Compare the segments found under 4 with the pairs found under 3. If a situation is found in which the end points of the third segment are reasonably close to mid-points of the other two, return with "yes" and the appropriate segment numbers.

The running time of a scene analysis is considerably longer than that of simple identification and varies greatly, depending on the number of points and the complexity of the scene. The analysis of the scene shown in Fig. 4 took about one and a half minutes.

5. DISCUSSION

A great deal of research in pattern recognition has been concerned with the classification problem[1][2]: given a number of classes (the alphabet, for example) and an input, one is required to assign the input to one of the classes.

The classification model of pattern recognition fails whenever there is more than a single item in the field of view (see, for example, Fig. 4). If there is more than one item, the problem is no longer to classify the input, but to distinguish the meaningful material from the background. In human perception the field of view contains a rich assortment of overlapping items, superimposed on complex backgrounds. CYCLOPS-1, with its hypothesis-generation and testing nature, is not bound by the limitations of the classification model. This has been demonstrated by the scene analysis capability of the system.

A rather different approach to the decomposition of overlapping line drawings using figure descriptions expressed in a list-processing language, is contained in Evans[3]. Earlier work in pattern recognition involving some decomposition of a figure into parts, not for the separation of overlapping figures but as part of a pattern classification procedure, is found in Grimsdale *et al.*[4] and Sherman[5].

There are several extensions to CYCLOPS-1 that may be mentioned as future research goals. The scene analysis capability of CYCLOPS-1 is two-dimensional, but could be extended to handle, three-dimensional scenes. Roberts'[6] work in the recognition and analysis of three-dimensional objects is applicable.

It would be desirable to replace the scope and light pen mode of input with an optical scanner. A compatible optical scanning technique, employing the PDP-1 computer has been demonstrated by Rudloe, Deutsch and Marill[7], and would be suitable to the present system.

It would be possible to incease the efficiency of the system by introducing more sophisticated hypothesis-generation schemes. Two techniques bear investigating: (a) If the system is to be used to read English text, the hypothesis-generation scheme should be made to reflect the statistical structure of English. (b) In any event, if, during the operation of the system, one or more hypotheses have been rejected, a great deal of analysis will already have been done on the input. This analysis should suggest the next hypothesis to test. In short, new hypotheses should be suggested by characteristics already discovered in the input.

Finally, one would wish to investigate ways in which a system of this type may be made to learn from examples. The work of W. Teitelman[8] may be particularly applicable in this connection.

BIBLIOGRAPHY

1. T. MARILL and D. M. GREEN, "Statistical Recognition Functions and the Design of Pattern Recognizers," *IRE Trans. on Electronic Computers,* EC-9, 472-477, 1960.
2. T. MARILL and D. M. GREEN, "On the Effectiveness of Receptors in Recognition Sys-

tems," *IEEE Trans. on Information Theory,* IT-9, 11-17, 1963.

3. T. G. EVANS, "A Heuristic Program to Solve Geometric-Analogy Problems," Ph.D. Thesis, Department of Mathematics, MIT, June 1963.
4. R. L. GRIMSDALE, F. H. SUMNER, C. J. TUNIS, and T. KILBURN, "A System for the Automatic Recognition of Patterns," *Proc. IEE,* Vol. 106, Pt. B, No. 26, P. 215, March 1959.
5. H. SHERMAN, "A Quasi-Topological Method for Machine Recognition of Line Patterns," pp. 232-237, ICIP Proceedings, Paris, France, June 1959.
6. L. G. ROBERTS, "Machine Perception of Three-Dimensional Solids," Ph.D. Thesis, Department of Electrical Engineering, MIT, June 1963.
7. H. RUDLOE, M. DEUTSCH, and T. MARILL, "PIP: A Photo-Interpretive Program for the Analysis of Spark-Chamber Data," *Communications ACM,* June 1963.
8. W. TEITELMAN, "New Methods for Real-Time Recognition of Hand-Drawn Characters," M.S. Thesis, MIT, 1963. Also, Bolt Beranek and Newman Inc. Report No. 1015.

SIMULATION OF A TURING MACHINE ON A DIGITAL COMPUTER

Robert W. Coffin, Harry E. Goheen,** and Walter R. Stahl****

I. INTRODUCTION

The theory of algorithms relies heavily upon the conceptual and theoretical usefulness of the Turing Machine.(1) Recent work by Trakhenbrot(2) has given further support to the hypothesis that "all algorithms can be given in the form of functional matrices and executed by the corresponding Turing Machines." Such a statement does not immediately suggest that all problems should be reduced to their equivalent Turing Machine, but the implication is clear that if certain problems, recognizable as algorithms, do not lend themselves to a solution in a formal logic structure, they may be reduced to a Turing scheme using a suitable strategy. Examples of such problems are revealed in the work by Lusted and Stahl(3) in the problem of medical diagnosis and by Stahl and Goheen(4) in simulation of the operation of biological cell systems.

For years the practical value of the Turing Machine has been discounted because there is widespread belief among mathematicians that a Turing Machine always requires a tape of infinite length and that there is no assurance that a particular algorithm will achieve a stable solution in a finite length of time. These misconceptions probably stem from the fact that the Turing Machine invariably enters the discussion of complex, self-organizing automata in which these conditions might exist. Furthermore, the work of A. M. Turing(5) preceded the growth of modern electronic technology by several years, consequently to implement a Turing Machine as a physical device, at that time, would have been an impractical and costly undertaking.

The authors and their colleagues have found that simulation of the Turing Machine on a Digital Computer is a useful and practical tool not only for problem solving and validation of algorithms but also for teaching students the fundamentals of programming. This paper will describe the digital program by which a generalized simulation of the Turing Machine has been accomplished.

II. PRELIMINARY CONSIDERATIONS

If there are M configurations corresponding to the Q-levels of the Turing Machine each requiring m bits, and if the Turing Machine requires an alphabet of size N, each symbol requiring n bits, we shall need a list of MN strings which we call machine quintuples. A machine quintuple will consist of:

(a) m bits identifying a Q-level,

(b) n bits identifying a symbol of the alphabet,

(c) m bits designating the next Q-level,

*Chief Programmer, Dept. of Automatic Data Processing and Biomathematics, Oregon Regional Primate Research Center, Beaverton, Oregon.

**Professor of Mathematics, Dept. of Mathematics, Oregon State University, Corvallis, Oregon.

***Associate Scientist, Dept. of Automatic Data Processing and Biomathematics, Oregon Regional Primate Research Center, Beaverton, Oregon, and Associate Professor, Dept. of Mathematics, Oregon State University, Corvallis, Oregon.

(d) 2 bits designating the next motion (either Left, Right, or Place),

(e) n bits designating the symbol to replace the symbol represented by (b) at the current location in the Turing tape.

In the descriptive language of the Turing Table, the last $(m + 2 + n)$ bits represent the (i, j) entry of the Turing Table, where i and j are given in (a) and (b) of the quintuple. The list of quintuples will require a maximum of $2MN\ (m + n + 1)$ bits. As a practical matter, Turing Tables are rarely "saturated," i.e., the entire matrix filled with non-zero entries, however the maximum bit requirement:

$$2MN\ (m + n + 1) = A \qquad (1)$$

is an important program design criterion.

To provide for rapid and intelligible interpretation of a processed Turing tape, it is desirable to limit the magnitude of the Turing alphabet to those alphabets of available input-output devices. It is important to note that this restriction is one of convenience since, in binary form, a symbol is limited only by the allowable n. If a Turing Machine requires a large alphabet it would be necessary to numerically precode the alphabet, or alternatively, use a two-for-one notation.

In addition to (1) a portion of the total available memory must be used to store the Turing tape. If this memory allocation is b bits, then the maximum possible length of the Turing tape will be the parameter:

$$[b/n] = B \qquad (2)$$

If the total number of bits in the memory of a specific computer is S, and the simulation program requires s bits, we have the following inequality which defines the maximum combination of the Turing Machine and tape that can be processed.

$$A + B < S - s \qquad (3)$$

In addition to (3), there is another restriction imposed upon the simulation program. We denote by r the time used in passing from one quintuple to the next. If an algorithm requires T quintuple cycles for solution, we have the following inequality in which C denotes the time available for the processing of the algorithm.

$$r\,T < C \qquad (4)$$

The inequalities (3) and (4) are viewed as general restrictions upon the magnitude of Turing Machine which can be simulated on a particular computer.

III. STRATEGY OF SIMULATION

The simulation of a machine in a digital computer requires two major routines that are logically distinct from each other.(6) The first of these routines serves to receive and interpret external notation and generate the "compiled logic" of the device to be simulated. Whether or not this routine is a true compiler, in the sense of current usage of the word, depends upon the logical complexity of the simulated device and the corresponding notation required to fully describe its function. The logic of the Turing Machine is quite simple and completely embodied in the construction of the quintuple which in turn is determined by the author of the Turing Machine. Therefore the builder is required only to make the notational translation of externally coded quintuple statements into machine language. To distinguish between a true compiler and the routine that generates the "compiled logic", we shall henceforth refer to the latter as the "builder" routine.

The essential steps of the builder routine are outlined below.

1. Read quintuple card.
2. HALT card? Yes, go to 14.
3. STOP quintuple? Yes, go to 12.
4. Extract (Q, S, Q, M, S).
5. Convert Q-s to binary.
6. Assign movement code.
7. Form function: (Q, M, S).
8. Form identifier (Q, S).
9. Store identifier and function.
10. Update storage.
11. Go to 1.
12. Insert stop code for function.
13. Go to 8.
14. Exit Builder.

Each quintuplet is punched on a separate card in the format

Q S Q M S for example
10 A 11 R B

where 10 A is the Turing Matrix identification pair and 11 R B is the function triplet.

The only exception to this format is the special stop card with the format:

Q S STOP

We have chosen to use a special stop code for three reasons. First, the special stop code makes it possible to distinglish between a normal halt and an error in the Turing Machine. Secondly, it is far more economical of computer time to make a simple, one-step test rather than go through three, multi-step tests to determine the (Qi, Sj, Qi, P, Sj) quintuple. Finally, the special stop code and the natural stop quintuple are theoretically equivalent.

The quintuple deck is terminated with the halt card:

HALT

The dimensions of the Turing Table are determined by the builder. The row dimension is equal to the largest defined value of Q, and the column dimension is equal to the number of different symbols defined in the quintuple list. It must be noted that the quintuple notation makes the determination of the dimensions unnecessary except for purposes of arraying the matrix on a line printer or other device.

Since the speed of the builder routine is limited by the speed of the card reader there is time for nonessential checking for mispunched cards. The inclusion of three such checking features has proved to be very helpful in ridding quintuple decks of gross errors. The first and most valuable check is dependent upon an alphabet definition card that is read immediately prior to the first quintuple card. A short routine generates a check list consisting of all of the defined symbols which can legally occur in the quintuple deck. Then as the quintuple cards are read, each symbol is checked against the defined alphabet. In the event an undefined symbol appears, the illegal symbol and the card containing the error are typed out with an error message. The error routine keeps a list of the addresses where the incorrect cards should be located, and the builder proceeds to the next card. After the HALT card is detected the program stops and allows the operator to insert the corrected cards in the card hopper. Upon restarting, the program will translate the corrected cards and store the corresponding entries in their original order. Similar checks are made to detect illegal movement and non-numeric symbols in the Q state, with identical procedures for error recovery.

After the building is complete, before any simulation can take place, a Turing tape must be loaded into memory. The Turing tape is punched on one or more cards, and each symbol is checked against the original alphabet list. If an illegal symbol occurs in the Turing tape, an error message is typed and the reader routine halts. Upon restarting, the corrected tape is read, checked and stored.

The second part of the program is called the "driver" routine which operates on an element of compiled logic, provides functional continuity between elements, and hence forces the simulation. In the ideal case, no part of the driving routine performs any function except to operate on an element or provide continuity between elements, however, the timing restriction will certainly have to be included in the driver routine and is considered to be an allowable artifact. Other artifacts may be justified for purposes of demonstration or instruction but are undesirable burdens on the simulation since time is at a premium and each artifact must consume time on every quintuple cycle. Clearly, any artifact that causes the driver routine to amend itself or alter an element of compiled logic is unallowable.

The following outline shows the essential steps of the driver routine.

1. Extract symbol from Turing tape.
2. Merge symbol with previous Q-level (from step 7).
3. Find function corresponding to (Q, S).
4. Is function a STOP? Yes, go to 14.
5. Substitute function symbol in Turing tape.
6. Move Turing tape "window".
7. Extract next Q-level.
8. Has time been exceeded? Yes, go to 10.
9. Go to 1.
10. Type time warning.
11. Halt.

12. Insert new time restriction.
13. Go to 1.
14. Output final Turing tape.
15. Halt.

The step which consumes the largest segment of time in the "Driver" is the table lookup to find the next function triplet, (step 3). The time required for this search can be greatly reduced if we take advantage of the natural tendency of Turing Machine authors to group important Q-levels together. Thus, the probability of finding the next quintuple in a close proximity to the current quintuple is much higher than finding it at a great distance. After a check is made against the current quintuple, a flutter search is used in which the search begins with the next entry forward of the current quintuple and then shifts to the last entry preceding then to the second entry forward and so forth until the required quintuple is located. It is important to note that the order in which the quintuples occur is important only to the efficiency of the algorithm but does not influence its logical operation.

IV. SAMPLE ALGORITHMS

The first problem is to detect the change of pattern in a strip of contrasting white and black segments as shown in Fig. 1 (a).

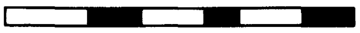


Figure 1(a).

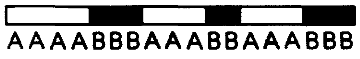


Figure 1(b).

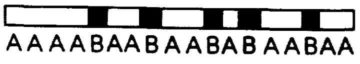


Figure 1(c).

To code the pattern as a string of symbols we simply assign the letter A to the white segments and the letter B to the black segments, and in both cases make the number of letters proportional to the length of each segment as shown in Fig. 1 (b). The problem is to write an algorithm in Turing notation that will operate on this coded string in such a way that only the points at which the pattern changes are marked. The desired solution is shown in Fig. 1 (c). This is logically equivalent to generating a unit pulse based upon a criterion of wave height or frequency.

The Turing Table for this algorithm is shown in Fig. 2, and is a simple 3 $\times$ 3 matrix with 2 null entries.

	*	A	B
Q1	2R*	—	—
Q2	2P*	2RA	3RB
Q3	3P*	2RB	3RA

Figure 2.

The strategy is to proceed from the left * to the right passing from Q1 to Q2 when either an A or B is encountered. If an A is seen, control stops in Q2 until a B is seen. The first B is retained and control goes to Q3 where successive B's are replaced by A's until an A is seen. The first A is replaced by a B and control passes back to Q2 which begins the algorithm again. If a B is seen as the first symbol, the same type of procedure is followed except control goes immediately to state Q3.

Fig. 3 shows the complete input deck as it appears before the start of a run.

```
*AB
1 *   2 R A
2 *   STOP
2 A   2 R A
2 B   3 R B
3 *   STOP
3 A   2 R B
3 B   3 R A
HALT
*AAAABBBAAABBAAABBB*
END TEST
```

Figure 3.

The first card is the alphabet definition card consisting of the three legal symbols: *, A, and B. Following the alphabet card is the seven-card Turing program which is terminated with a HALT card. The coded test string shown in Fig. 1(b) is next, followed by an END OF TEST card.

Fig. 4 shows the sequential operation of the Turing Machine. The current quintuple is

```
*AAAABBBAAABBAAABBB*    [ØØ1,*] [ØØ2,R,*]
 Δ
*AAAABBBAAABBAAABBB*    [ØØ2,A] [ØØ2,R,A]
  Δ
*AAAABBBAAABBAAABBB*    [ØØ2,A] [ØØ2,R,A]
   Δ
*AAAABBBAAABBAAABBB*    [ØØ2,A] [ØØ2,R,A]
    Δ
*AAAABBBAAABBAAABBB*    [ØØ2,A] [ØØ2,R,A]
     Δ
*AAAABBBAAABBAAABBB*    [ØØ2,B] [ØØ3,R,B]
      Δ
*AAAABABAAABBAAABBB*    [ØØ3,B] [ØØ3,R,A]
       Δ
*AAAABAAAAABBAAABBB*    [ØØ3,B] [ØØ3,R,A]
        Δ
*AAAABAABAABBAAABBB*    [ØØ3,A] [ØØ2,R,B]
         Δ
*AAAABAABAABBAAABBB*    [ØØ2,A] [ØØ2,R,A]
          Δ
*AAAABAABAABBAAABBB*    [ØØ2,A] [ØØ2,R,A]
           Δ
*AAAABAABAABBAAABBB*    [ØØ2,B] [ØØ3,R,B]
            Δ
*AAAABAABAABAAAABBB*    [ØØ3,B] [ØØ3,R,A]
             Δ
*AAAABAABAABABAABBB*    [ØØ3,A] [ØØ2,R,B]
              Δ
*AAAABAABAABABAABBB*    [ØØ2,A] [ØØ2,R,A]
               Δ
*AAAABAABAABABAABBB*    [ØØ2,A] [ØØ2,R,A]
                Δ
*AAAABAABAABABAABBB*    [ØØ2,B] [ØØ3,R,B]
                 Δ
*AAAABAABAABABAABAB*    [ØØ3,B] [ØØ3,R,A]
                  Δ
*AAAABAABAABABAABAA*    [ØØ3,B] [ØØ3,R,A]
                   Δ
*AAAABAABAABABAABAA*    [ØØ3,*]  STOP
                   Δ
RUN COMPLETE
```

Figure 4.

shown at the right side of the tape. The delta symbol under each line shows the next symbol to be viewed by the "window".

In this example the Turing tape is printed at the completion of every quintuple cycle consequently the "window" is shown to move only one symbol per line of output. Such an extensive output is desirable only for short, illustrative examples since an algorithm may involve many thousands of cycles for solution.

The second algorithm is a popular child's problem in which a number of boys and soldiers are to be transported across a river in a row boat. The boat is capable of holding one or two boys, or one soldier, but not one boy and one soldier. The basic solution strategy is a simple five step algorithm as follows:

1. two boys across river
2. one boy back
3. boy out, soldier in, soldier across.
4. boy back
5. go to 1.

When all of the soldiers are across the river, the algorithm reduces to a back and forth shuttle involving only boys with a net gain of one boy per two river crossings. The problem can be coded in an interesting manner with symbols representing the various objects involved.

G is a grassy spot on the river bank,
W is the water,
F is a seat in the row boat,
B is a boy,
S is a soldier,
% is the left bank of the river,
$\bowtie$ is the right bank of the river
$\neq$ is the tape end marker symbol
— is an intermediate symbol used to denote movement of the boat.

A sample configuration is shown in Fig. 5(a) with the solution configuration in Fig. 5(b).

At the beginning all of the boys and soldiers are on the right side of the river with the boat at the right bank. In the final string, all of the boys and soldiers have been transported to the left side and the boat is at the left bank.

Figure (6) shows the Turing program in quintuple form required to process any such input configuration.

```
#GGGGG%WWFF⌶SBBSSG#
```

Figure 5(a).

```
#BSSSB%FFWW⌶GGGGGG#
```

Figure 5(b).

```
 1 #  1 R #        1 G  2 R G        2 %  3 R %
 2 G  2 R G        3 ▬  4 R ▬        3 F  3 R F
 3 W  3 R W        3 #  4 R #        4 # STOP
 4 G  4 R G        4 S  4 R S        4 B  5 R G
 5 S  5 R S        5 B  6 L G        5 G  5 R G
 5 # 24 L #        6 F  7 L B        6 S  6 L S
 6 ▬  6 L ▬        6 G  6 L G        6 #  6 L #
 7 S  8 L S        7 B  8 P B        7 F  7 L B
 7 W  8 R W        7 G  8 P B        8 W  8 R W
 8 # 13 R #        8 % 13 R %        8 S 10 L -
 8 F 11 L -        8 B  9 L -        9 F  8 L F
 9 W 12 R B        9 -  9 L B        9 B  8 L B
 9 S  8 L S       10 S  8 L S       10 - 10 L S
10 W 12 R S       10 F  8 L F       10 B  8 L B
11 S  8 L S       11 - 11 L F       11 F  8 L F
11 B  8 L B       11 W 12 R F       12 S 10 L W
12 B  9 L W       12 F 11 L W       12 - 12 R -
13 S 15 L F       13 F 13 R F       13 B 14 L F
14 B 14 L B       14 S 14 L S       14 F 14 L F
14 % 14 L %       14 G 16 R B       14 # 31 L %
15 G 17 R S       15 F 15 L F       15 B 15 L B
15 S 15 L S       15 % 15 L %       16 F 16 R F
16 W 19 L W       16 % 16 R %       16 B 16 R B
16 S 16 R S       17 G 17 R G       17 B 18 R G
17 S 17 R S       18 % 18 R %       18 F 16 R B
18 G 18 R G       18 S 18 R S       18 B 18 R B
19 B 20 R -       19 - 19 R -       19 ▬ 23 L ▬
19 F 21 R -       19 # 23 L #       19 W 19 L W
20 F 19 R F       20 B 19 R B       20 - 20 R B
20 W 22 L B       21 F 19 R F       21 W 22 L F
21 B 19 R B       21 - 21 R F       22 B 20 R W
22 F 21 R W       22 - 22 L -       23 B 24 R F
23 F 23 L F       24 # 24 R #       24 G 25 L B
24 F 24 R F       24 ▬ 24 R ▬       24 B 24 R B
24 S 24 R S       25 ▬ 26 R #       25 G 25 L G
25 #  3 P ▬       25 S 25 L S       25 B 25 L B
26 S 27 L G       26 # 28 L #       26 G 26 R G
26 B 26 R B       26 F 26 L F       26 W  8 R W
27 F 26 L S       27 # 27 L #       27 B 27 L B
27 G 27 L G       28 G 28 L G       28 F 29 L B
28 ▬ 28 L ▬       28 B 29 L B       29 B 25 L B
29 F 29 L F       29 ▬ 30 R ▬       29 # 31 L ▬
29 G 29 L G       29 W  8 R W       30 ▬ 30 R ▬
30 B 28 L G       30 F 30 R F       30 G 30 R G
30 W 30 R W       31 S 31 L S       31 B 31 L B
31 F 31 L F       31 W 31 L W       31 G 32 L B
31 % 30 R #       32 # STOP         32 G 32 L G
32 B 32 L B       32 S 32 L S
```

Figure 6.

In addition to the standard solution algorithm, all contingencies are anticipated such as the presence of only one boy; or the absence of soldiers.

Because of the number of quintuple cycles involved it is not practical to show the test string after each cycle, however, figure 7 shows every second cycle.

Note that the "window" appears to take a two-symbol jump between lines.

V. DISCUSSION

The sample algorithms were run on a machine with a cycle time of 8 microseconds (SDS 920). The limiting factor, as mentioned above, is the time involved in finding the next quintuple in the total list. Since the efficiency is influenced by the order in which the quintuple deck is organized, the question arises whether or not it is possible to obtain an optimum ordering. Generally, this would be quite unlikely since an optimum order is dependent on the input string which may be composed of a random group of symbols. However, if quintuples, containing symbols which are used in conjunction with each other, are placed together in the input deck, the efficiency can probably be improved.

In the Boys and Soldiers algorithm we achieved a speed of 3,000 quintuple cycles/second by ordering the deck row-by-row as it appears in a Turing Functional Matrix. By rearranging some of the cards, particularly in Q-levels which deal with the left and right movement of the boat, we were able to increase the speed to about 3,300 quintuple cycles/seconds. In the opposite direction, we tried to arrange the deck in such a way that we produced a very inefficient operation and were able to reduce the speed to slightly less than 1,000 quintuple cycles/second. By shuffling the cards like a bridge deck, we obtained about 2,100 quintuple cycles/second.

It is not surprising that the row-by-row ordering is quite efficient since a Turing programmer normally adopts a row-by-row strategy in solving a problem, consequently the probability is high that significant quintuples will be grouped together. Experience with a

```
#GGG%WWFFⅡBSB#   [ØØ1,G] [ØØ2,R,G]
  Δ
#GGG%WWFFⅡBSB#   [ØØ2,G] [ØØ2,R,G]
    Δ
#GGG%WWFFⅡBSB#   [ØØ3,W] [ØØ3,R,W]
      Δ
#GGG%WWFFⅡBSB#   [ØØ3,F] [ØØ3,R,F]
        Δ
#GGG%WWFFⅡBSB#   [ØØ3,Ⅱ] [ØØ4,R,Ⅱ]
          Δ
#GGG%WWFFⅡGSB#   [ØØ5,S] [ØØ5,R,S]
            Δ
#GGG%WWFFⅡGSG#   [ØØ6,S] [ØØ6,L,S]
          Δ
#GGG%WWFFⅡGSG#   [ØØ6,Ⅱ] [ØØ6,L,Ⅱ]
        Δ
#GGG%WWBBⅡGSG#   [ØØ7,F] [ØØ7,L,B]
      Δ
#GGG%WW-BⅡGSG#   [ØØ8,B] [ØØ9,L,-]
      Δ
#GGG%WB-BⅡGSG#   [Ø12,-] [Ø12,R,-]
        Δ
#GGG%WBBWⅡGSG#   [ØØ9,-] [ØØ9,L,B]
      Δ
#GGG%WBBWⅡGSG#   [ØØ8,W] [ØØ8,R,W]
      Δ
#GGG%B-BWⅡGSG#   [ØØ9,W] [Ø12,R,B]
      Δ
#GGG%B-WWⅡGSG#   [Ø12,B] [ØØ9,L,W]
      Δ
#GGG%BBWWⅡGSG#   [ØØ9,B] [ØØ8,L,B]
    Δ
#GGG%FBWWⅡGSG#   [Ø13,B] [Ø14,L,F]
    Δ
#GGB%FBWWⅡGSG#   [Ø14,G] [Ø16,R,B]
    Δ
#GGB%FBWWⅡGSG#   [Ø16,F] [Ø16,R,F]
      Δ
#GGB%FBWWⅡGSG#   [Ø16,W] [Ø19,L,W]
      Δ
#GGB%F-BWⅡGSG#   [Ø2Ø,W] [Ø22,L,B]
      Δ
#GGB%W-BWⅡGSG#   [Ø22,F] [Ø21,R,W]
      Δ
#GGB%WFBWⅡGSG#   [Ø21,B] [Ø19,R,B]
        Δ
#GGB%WF-WⅡGSG#   [Ø19,B] [Ø2Ø,R,-]
        Δ
```

Figure 7.1.

```
#GSG%WB-W#BGG#   [Ø19,F] [Ø21,R,-]
        Δ
#GSG%WB-F#BGG#   [Ø22,-] [Ø22,L,-]
      Δ
#GSG%WWBF#BGG#   [Ø2Ø,-] [Ø2Ø,R,B]
        Δ
#GSG%WWBF#BGG#   [Ø19,#] [Ø23,L,#]
        Δ
#GSG%WWFF#BGG#   [Ø23,B] [Ø24,R,F]
        Δ
#GSG%WWFF#BGG#   [Ø24,#] [Ø24,R,#]
          Δ
#GSG%WWFF#BBG#   [Ø24,G] [Ø25,L,B]
          Δ
#GSG%WWFFⅡBBG#   [Ø25,#] [ØØ3,P,Ⅱ]
         Δ
#GSG%WWFFⅡGBG#   [ØØ4,B] [ØØ5,R,G]
           Δ
#GSG%WWFFⅡGGG#   [ØØ6,G] [ØØ6,L,G]
         Δ
#GSG%WWFBⅡGGG#   [ØØ6,F] [ØØ7,L,B]
       Δ
#GSG%WWBBⅡGGG#   [ØØ7,W] [ØØ8,R,W]
       Δ
#GSG%WB-BⅡGGG#   [ØØ9,W] [Ø12,R,B]
       Δ
#GSG%WB-WⅡGGG#   [Ø12,B] [ØØ9,L,W]
       Δ
#GSG%WBBWⅡGGG#   [ØØ9,B] [ØØ8,L,B]
     Δ
#GSG%W-BWⅡGGG#   [ØØ8,B] [ØØ9,L,-]
     Δ
#GSG%B-BWⅡGGG#   [Ø12,-] [Ø12,R,-]
       Δ
#GSG%BBWWⅡGGG#   [ØØ9,-] [ØØ9,L,B]
     Δ
#GSG%BBWWⅡGGG#   [ØØ8,%] [Ø13,R,%]
     Δ
#GSG%FBWWⅡGGG#   [Ø14,%] [Ø14,L,%]
   Δ
#GSB%FBWWⅡGGG#   [Ø16,%] [Ø16,R,%]
     Δ
#GSB%FBWWⅡGGG#   [Ø16,B] [Ø16,R,B]
       Δ
#GSB%F-WWⅡGGG#   [Ø19,B] [Ø2Ø,R,-]
       Δ
#GSB%F-BWⅡGGG#   [Ø22,-] [Ø22,L,-]
     Δ
```

Figure 7.2.

```
#GGB%WF-B⌑GSG#   [022,-] [022,L,-]
      Δ
#GGB%WWFB⌑GSG#   [021,-] [021,R,F]
        Δ
#GGB%WWFB⌑GSG#   [019,⌑] [023,L,⌑]
        Δ
#GGB%WWFF⌑GSG#   [024,⌑] [024,R,⌑]
          Δ
#GGB%WWFF#BSG#   [025,⌑] [026,R,#]
          Δ
#GGB%WWFF#BGG#   [026,S] [027,L,G]
          Δ
#GGB%WWFF#BGG#   [027,#] [027,L,#]
        Δ
#GGB%WWFS#BGG#   [026,F] [026,L,F]
      Δ
#GGB%WW-S#BGG#   [008,F] [011,L,-]
      Δ
#GGB%WF-S#BGG#   [012,-] [012,R,-]
        Δ
#GGB%WFSW#BGG#   [010,-] [010,L,S]
      Δ
#GGB%WFSW#BGG#   [008,W] [008,R,W]
      Δ
#GGB%F-SW#BGG#   [011,W] [012,R,F]
      Δ
#GGB%F-WW#BGG#   [012,S] [010,L,W]
      Δ
#GGB%FSWW#BGG#   [010,F] [008,L,F]
    Δ
#GGB%FSWW#BGG#   [013,F] [013,R,F]
      Δ
#GGB%FFWW#BGG#   [015,F] [015,L,F]
    Δ
#GGB%FFWW#BGG#   [015,B] [015,L,B]
  Δ
#GSG%FFWW#BGG#   [017,B] [018,R,G]
    Δ
#GSG%BFWW#BGG#   [018,F] [016,R,B]
      Δ
#GSG%BFWW#BGG#   [016,W] [019,L,W]
      Δ
#GSG%B-FW#BGG#   [021,W] [022,L,F]
      Δ
#GSG%W-FW#BGG#   [022,B] [020,R,W]
      Δ
#GSG%WBFW#BGG#   [020,F] [019,R,F]
        Δ
```

Figure 7.3.

```
#GSB%WFBW⌑GGG#   [021,-] [021,R,F]
        Δ
#GSB%WFBW⌑GGG#   [019,W] [019,L,W]
        Δ
#GSB%WF-B⌑GGG#   [020,W] [022,L,B]
        Δ
#GSB%WW-B⌑GGG#   [022,F] [021,R,W]
        Δ
#GSB%WWFB⌑GGG#   [021,B] [019,R,B]
          Δ
#GSB%WWFF⌑GGG#   [023,B] [024,R,F]
          Δ
#GSB%WWFF⌑BGG#   [024,G] [025,L,B]
          Δ
#GSB%WWFF#BGG#   [026,B] [026,R,B]
            Δ
#GSB%WWFF#BGG#   [026,G] [026,R,G]
              Δ
#GSB%WWFF#BGG#   [028,G] [028,L,G]
            Δ
#GSB%WWFF#BGG#   [028,B] [029,L,B]
          Δ
#GSB%WWFF⌑BGG#   [031,F] [031,L,F]
       Δ
#GSB%WWFF⌑BGG#   [031,W] [031,L,W]
     Δ
#GSB#WWFF⌑BGG#   [031,%] [030,R,#]
     Δ
#GSB#WWFF⌑BGG#   [030,W] [030,R,W]
       Δ
#GSB#WWFF⌑BGG#   [030,F] [030,R,F]
          Δ
#GSB#WWFF⌑GGG#   [030,B] [028,L,G]
          Δ
#GSB#WWFB⌑GGG#   [028,F] [029,L,B]
       Δ
#GSB#WWFB⌑GGG#   [029,W] [008,R,W]
       Δ
#GSB#WF-B⌑GGG#   [011,W] [012,R,F]
       Δ
#GSB#WF-W⌑GGG#   [012,B] [009,L,W]
       Δ
#GSB#WFBW⌑GGG#   [009,F] [008,L,F]
     Δ
#GSB#W-BW⌑GGG#   [008,F] [011,L,-]
     Δ
#GSB#F-BW⌑GGG#   [012,-] [012,R,-]
       Δ
```

Figure 7.4.

```
#GSB#FBWWMGGG#   [009,-] [009,L,B]
     △
#GSB#FBWWMGGG#   [008,#] [013,R,#]
     △
#GSB#FFWWMGGG#   [013,B] [014,L,F]
     △
#GSB%FFWWMGGG#   [014,#] [031,L,%]
   △
#GSB%FFWWMGGG#   [031,S] [031,L,S]
 △
#BSB%FFWWMGGG#   [032,#]  STOP
△
RUN COMPLETE
```

Figure 7.5.

particular algorithm would certainly indicate an efficient input order. However, it is an interesting characteristic of a Turing program that it will run to solution regardless of the input order of the instructions. The authors know of no other programming language in which this is true.

VI. CONCLUSION

The value of the simulation of a Turing Machine is two-fold. First, the simulation represents a practical means of solving algorithms in their most fundamental form, that is, in a basic language which is independent of both computer and computer language. Secondly, for non-numeric problems the processing rates are found to be competitive with the more conventional computer languages.

REFERENCES

1. McNAUGHTON, R. "The Theory of Automata, a Survey." *Advances in Computers,* vol. 2, pp. 379-421, edited by F. L. Alt, Academic Press, New York-London (1961).
2. TRAKHTENBROT, B. A. *Algorithms and the Machine Solution of Problems,* (In Russian, 1960). Available under the title, *Algorithms and Automatic Computing Machines,* D. C. Heath and Co., Boston (1963).
3. LUSTED, L. B., and W. R. STAHL, 1963. "Conceptual Basis of Medical Diagnosis." In press, *Proc. of the Conference on the Diagnostic Problem,* University of Michigan, May 1963.
4. STAHL, W. R., and H. E. GOHEEN. "Molecular Algorithms." In press, *J. Theoret. Biology* (1963).
5. TURING, A. M. "On Computable Numbers, with an Application to the Entscheidungsproblem". *Proc. London Mathematical Society,* Series 2, vol. 42 (1936-37), pp. 230-65.
6. AMDAHL, L. Handbook of *Automation Computation and Control,* edited by E. M. Grabbe, E. Ramo, and D. E. Wooldridge, vol. 2, chapter 17, pp. 40-41, John Wiley and Sons, Inc., New York (1959).
7. The authors gratefully acknowledge the advice of Dr. L. B. Lusted, Senior Scientist, Division of Biophysical Sciences, Oregon Regional Primate Research Center.

THE ROPE MEMORY—A PERMANENT STORAGE DEVICE

P. Kuttner
Electronic Instruments Division
Burroughs Corporation
Philadelphia 7, Pennsylvania

INTRODUCTION

A powerful way of increasing the capability and flexibility of digital computing systems is through the use of permanent storage memories. Such memories are also known as read-only memories or NDRO electrically unalterable memories. As an example of the application of a permanent memory, consider a computer used for control purposes. Generally, such systems are physically small, relatively inexpensive, and are not required to perform a variety but rather a restricted category of computations. The problem of program and constant input and storage in such systems is considerable. Permanent memory can satisfy the input and storage function required in such a system in an inexpensive manner. It can, furthermore, be packaged in such a manner as to minimize volume requirements; in any event, the volume occupied by the store for such an application is less than that required for tape or other inputs. In both large and small scale computers, permanent memory can be used for the storage of supervisory routines, input/output function routines, and, in the case of the rope memory, performing logic.

The rope memory is a true permanent storage device in that the information content of the memory is fixed by construction; any changes in the information content require that the device be rebuilt or exchanged. Memories in which data can be altered by replacing a changeable data element are classed as "semi-permanent memories". While this distinction may be logically tenable, it will not be considered in this paper: the changeability being considered here as simply a special case of rebuilding. Rope memories have been used in a number of computing systems; the use of a rope memory in a computer being designed by the M.I.T. Instrumentation Laboratory for the Apollo space craft was announced recently.[15]

The emphasis of this paper is on rope memories. The relation of a rope memory to a conventional read/write memory can only be based on the question of whether or not a permanent storage device can be used in a particular application. In order to place the properties of rope memories in their proper place in the class of permanent storage devices, some other techniques of this type will be discussed. (No comparison will be attempted with permanent memories based on optical or photographic principles.) These other techniques have received more attention up to now; the discussion to follow is based on the belief that ropes merit similar attention, all the more so since they give the opportunity for a greater range of applications.

REVIEW OF SOME PERMANENT MEMORY SCHEMES

Before entering on a discussion of rope memories, we shall briefly describe various other permanent memory schemes, the description of which will help in placing the information concerning rope memories into proper perspective. The permanent memory schemes to be described below are 1) capacitative type, 2) electromagnetic coupling type, and 3) permanent magnet twistor type. These memories have in common the following features:

1) They are organized in a linear select mode.
2) Each storage element stores one bit of information.
3) Output signals are of the order of a few millivolts.
4) The information state of the memories can be changed by a relatively uncomplicated procedure, i.e., by changing cards which form the basis of information storage.
5) Access is random
6) Some batch fabrication techniques are possible.

At the conclusion of the discussion concerning rope memories, we shall return to these points and examine them in the light of the behavior of rope memories.

Capacitative Type[4, 11, 14, 20]

Schematically, this memory can be represented as shown in Figure 1. A voltage pulse directed along any one of the horizontal word lines will be detected on a sense line if that line is coupled to the word line by a capacitor; otherwise, no pulse will appear on the sense line. An interrogation pulse applied to a word line causes all bits of that word to be read out in parallel.

The word and sense lines are etched on separate printed circuit boards. These boards are then oriented in such a manner that the word and sense lines are orthogonal. A metallized card insulated on both sides by a layer of mylar is sandwiched in between these boards. If a hole is punched in the card at the position where a word and sense line intersect, a small (about 1 pf) capacitor is created, thus coupling the sense to the word line corresponding to a "1" bit. The absence of a hole makes capacitative coupling between the lines impossible. Writing is thus accomplished by punching holes in the metallized card in a pattern in accordance with the information to be stored. The metallized card itself is grounded. Memories of this type have been reported to operate at 5 mc rates, with outputs of about 1 mv for storage capacity of 10^5 bits. Worst case analysis of noise conditions shows that a S/N ratio of 10:1 is attainable.

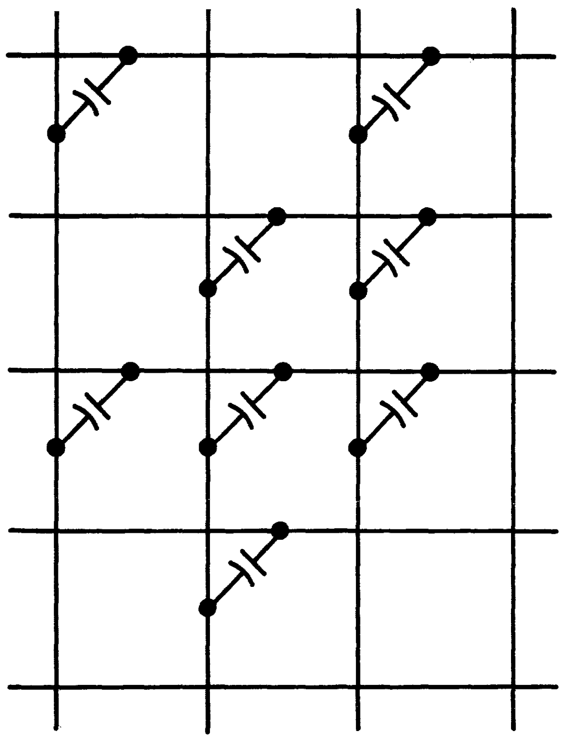


Figure 1. Schematic representation of the capacitative permanent memory.

The techniques for fabricating memories of this type are evidently suited to batch processes that should yield relatively inexpensive stores. Mechanical alignment is a problem that must be considered in the design of this type memory. The information state of the memory can be changed by changing the punched metallized cards.

Electromagnetic Coupling Type[3]

This particular technique is based on the fact that the mutual inductance between two conductors having a fixed geometrical relationship is a function of the medium separating them. If an alternating current, I, is caused to flow in one of a pair of one-turn coils, an output will be generated across the terminals of the second coil which is proportional to M_0 dI/dt. The induced emf in the second coil can be reduced, for a constant geometry, by inserting a shielding material between the coils. Thus, for example, the insertion of a copper plate of 0.04 mm thickness between two coils of 6.5 mm diameter separated by a distance of 0.75 mm causes a 23 db reduction in the induced emf. Binary information can thus be stored as the absence or presence of shield between the coils and can be detected as the absence or presence of an induced emf.

The primary coil is the *excitation* coil; the secondary coil is the *sensing* coil. Since excitation coils and sensing coils can be connected in series as shown in Figure 2, it is possible to manufacture arrays of coils by printed circuit techniques. The excitation coils are etched on one board, the sensing coils are etched on a second board. The orientation of these two boards relative to one another is such that the long axes are orthogonal. Each row of excitation coils represents one word, the sensing coils represent bits. The system is word organized.

Storage of information is effected by sandwiching a prepunched, insulated metal card between a pair of boards containing excitation and sensing coils. Holes are punched at the intersection of sensing and excitation coils at positions which are to store "1" bits. At positions which are to store "0" bits, no holes are punched. The relative outputs have already been discussed above. Mechanical alignment is

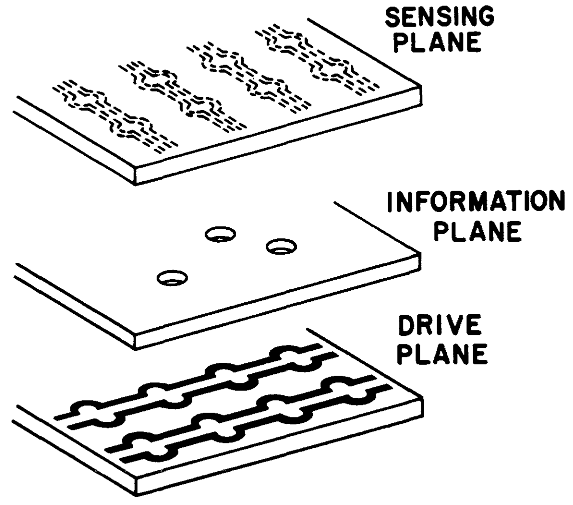


Figure 2. Electromagnetic coupling type permanent memory construction.

extremely critical here, since the holes must have a very definite geometrical relationship to the sense and excitation coils in order not to affect the mutual inductance. Again, stored information can be changed by exchanging cards. A memory of this type is reported as operating at 1.5 mc. Excitation currents used were 200 ma. Outputs are not explicitly reported but can be computed to be about 50 mv without taking attenuation into account. Worst case S/N ratios at the amplifier output are reported to be 30 db.

Permanent Magnet Twistor Type[2, 5, 7, 10]

The twistor is employed in a novel manner in this type of memory. Information is not stored in the twistor, but rather as the absence or presence of permanent magnets. Associated with each twistor element, there is a permanent magnet whose function is to generate an external field which will inhibit the twistor from switching if that element is to store a "0" bit. Alternately, if the twistor element is to store a "1" bit, it is not to be inhibited from switching, hence no permanent magnet is associated with it. Those twistors inhibited from switching will not change state on being interrogated; those twistors not inhibited will switch from one remanent state to the other upon being interrogated. It is thus necessary to follow up each interrogation pulse with a reset pulse.

In constructing a permanent magnet twistor type memory, the first step is to enclose twistor wires and their returns within mylar tape. This assembly is then bonded to solenoid mounting boards. The solenoids themselves are prepared by photoetching. The permanent magnets are a nickel-cobalt alloy, and are formed on a card either by photoetching or electrodeposition. Some 2,816 permanent magnets (bits) can be deposited on a card having an area of 55 in^2, giving a density of about 52 bits/in^2. The twistor wires and tape are continuous; the assembly of wire, tape, and solenoid boards is folded concertina fashion into a stack. Provision is made to guide the magnet card into proper registration with the solenoid board. (It is easy to imagine that alignment problems are critical here, especially in view of the fact that twistor elements may interact if spaced too closely together.) Memories with capacities of

1.44×10^6 bits have been reported, with an operational cycle of 5 μsec. Drives required are 2.0 amperes for the solenoids. Outputs shown for a 4,096 word module are 4 mv for "ones" and 0.8 mv for "zeroes", with a switching time of about 2 μsec.

THE ROPE MEMORY[1, 6, 8, 15, 21]

Principle of Operation

To illustrate the principle of operation of rope memories, suppose that a rope is to be constructed using 2^n cores. It must be possible to select each core uniquely. For purposes of selection, each core is threaded by a selected set of n out of 2n inhibit lines. All cores are also threaded by a common set and reset line. Initially all cores are in the same remanent state. Two inhibit registers are used for the purpose of selecting a given core. One of the registers stores the address of the core to be selected, the other register stores the complemented address of the core. These are the D and C registers respectively. Associated with each of the 2n bits in the registers is a driver which is enabled if the bit is a "1"; one driver is connected to one inhibit line. The set line is enabled simultaneously with the inhibit lines; the set and inhibit pulses are of equal amplitude but opposite polarity. The amplitude of each pulse is greater than the switching threshold of the core. The polarity of the set pulse is such that in the absence of any inhibit pulses it will switch a core fully. The polarity of the inhibit pulses is such as to prevent a core from switching—the inhibit pulses will drive the core further into saturation.

The selected core is not threaded by any of the enabled inhibit lines and is thus switched by the set pulse. During this phase of the operation (called the selection phase), the cores act as null decoders[21]. Referring to Figure 3, we see that the core will be set if, and only if, none of the threaded $\overline{D}n$ or $\overline{C}n$ line drivers is enabled. Hence,

$$\overline{\overline{D}_0 + \overline{C}_1 + \overline{C}_2} = D_0\, C_1\, C_2 = 1$$

$$\overline{\overline{D}_0 + \ldots + \overline{D}_n + \overline{C}_1 + C_2 + \ldots} = D_0 \ldots D_n \,.\, C_1 \,.\, C_2 = 1$$

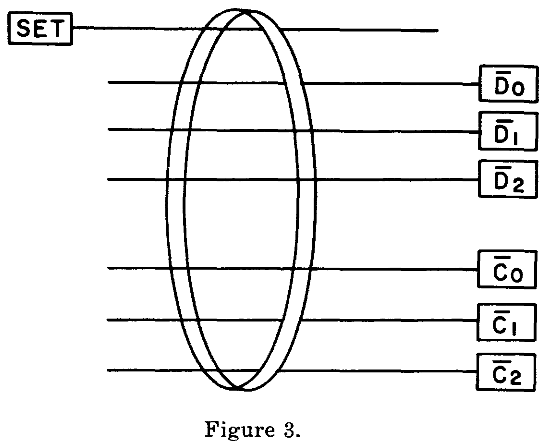


Figure 3.

The reset pulse is of the same amplitude as the set and inhibit pulses. Its polarity is such that it will reset the core that was set during the selection phase of the rope memory operating cycle. During the read phase the core output obtained as a consequence of the application of the reset pulse is used to generate the information pattern. At the end of the reset pulse, all cores are again in the same remanent state. This selection scheme is also known as the Rajhman switch[16].

Each core may store m bits; this requires the use of m sense lines. Information patterns are generated by threading a sense line through a core if the bit to be stored is a "1"; otherwise, the core is bypassed by the particular sense line—these lines correspond to "0" bits as no output will appear across them when the core is reset. The sense lines are common to all cores; a given sense line will thread those cores wherein a "1" is to be stored in the bit position represented by that sense line and will bypass those cores wherein a "0" is to be stored in the bit position represented by that sense line.

Figure 4 illustrates the wiring scheme for a rope memory of eight words (n=3). In the illustration each core stores a three-bit word. The information stored is the address of the word storing it. For a rope consisting of eight cores, there are six inhibit lines; three of these are associated with the direct register, the other three lines are associated with the complemented register. Assume that the data stored in word 010 is to be read out. The operation is as follows:

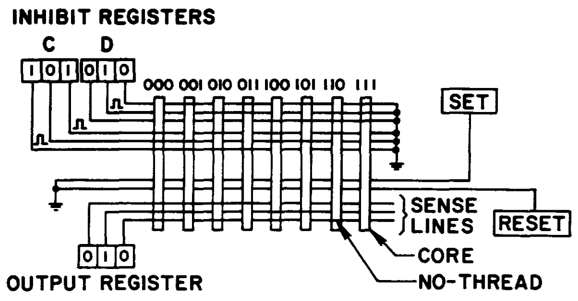


Figure 4a. Rope Memory Wiring Scheme.

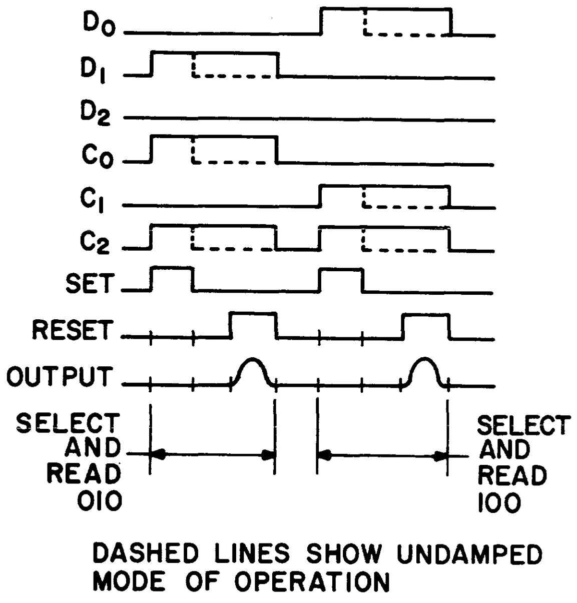


Figure 4b. Timing Diagram for Rope Memory Operation.

1) 010 is stored in the direct inhibit register. 101 is stored in the complemented inhibit register.
2) During the selection phase, the set line and the drivers corresponding to "1" in the registers are enabled.
3) Core 010 will be set since none of the enabled inhibit lines thread it.
4) During the read phase, the reset line is enabled and the information stored in the word is transferred to the output register via the sense lines.

The wiring pattern of the inhibit lines can be generalized as shown in Table I. (The LSB of the address is taken as k=0.)

TABLE I

kth bit of address	D_k	C_k
0	threads	does not thread
1	does not thread	threads

Rope Organization

Rope memories may be organized in two different ways. The m bits stored by a given core may be considered as the m bits of one word. Alternately, each of the m bits stored by a given rope will be referred to as the "one-core-per-ferent words. These two ways of organizing a core will be referred to as the "one-core-per-word" and "one-core-per-output bit" modes of storage respectively. Regardless of the organization, the bits stored by any one core are read out in parallel.

A rope organized on a one-core-per-word basis is a random access memory. It is possible to subdivide the m bits stored per core into n equal parts of n bits each so that a core may be considered as storing m/n words of n bits each. This requires that one of m/n sets of sense lines be selected to read out the proper information; for example, if 256 cores are to store four words each for a total of 1024 words, the address must be 10 bits long, eight to select the proper core and two to select the proper one of four words.

A one-core-per-output bit organization requires that the rope be read out sequentially in order that the bits of a word appear in proper order. While this mode of operation is possible, it is not extremely desirable. For one thing, the time elapsed to fully read out a word becomes somewhat prohibitive: given a cycle time of 8μsec and a k-bit word, 8k μsec would elapse before one word is fully read out. For another thing, the number of bits that a core can store is necessarily related to its inner diameter since this is the factor limiting the number of lines that may be threaded through it. Typically, a core having an I.D. of 0.140" can have 128 sense lines threaded through it. This number of sense lines is certainly adequate for a rope organized on a one-core-per-word basis; it is not necessarily adequate for a rope organized on a one-core-per-output bit basis.

Rope Memory Characteristics

A prototype rope can be seen in Figure 5a. Figure 5b shows a packaged unit. This rope consists of 256 cores, each of which stores four 16-bit words. The storage elements used in this rope are 26 maxwell bobbin cores made of 1/8 mil 4-79 Molybdenum Permalloy. The inner diameter of the bobbins is 0.140″. If driven by ramps having a slope of 400 ma/μsec, the output of the rope is of the order of 200 mv, with an absolute S/N of about 9:1. (See Figure 6—Damped Operation.) The switching times of the cores are somewhat less than 2 μsec; but

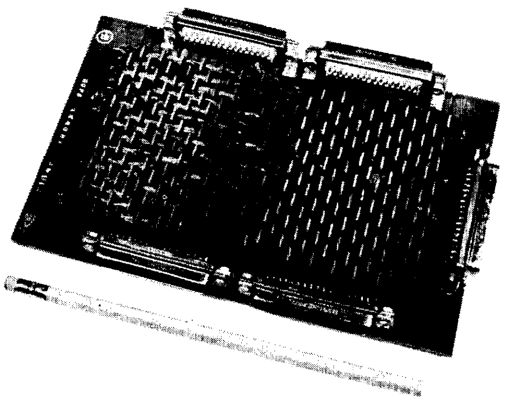


Figure 5a. Prototype Rope Memory Storing 256 Words, 64 Bits per Word.

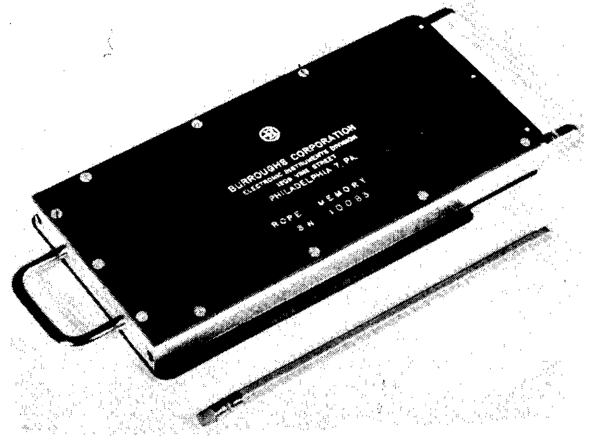


Figure 5b. Rope Memory Package Capacity: 256 Words, 64 Bits per Word.

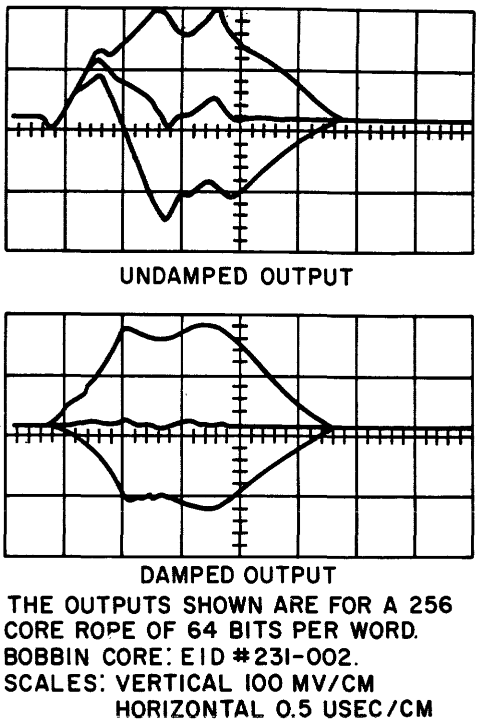


Figure 6. Undamped and damped mode of operation.

since peaking occurs at 2 μsec after the start of the ramp, the cycle time is 8 μsec. The values of resistance and inductance of various lines are tabulated in Table II.

TABLE II

Line	Resistance	Inductance
Set	2 ohms	11 μh
Inhibit	2 ohms	7 μh
Sense	10 ohms	12 μh

These data are suggestive of the performance of a rope memory. Various factors pertaining to rope memory operation are discussed below.

Drive Currents—It is possible to drive ropes with either square pulses or ramps as mentioned above. Owing to the large inductance of the selection and sense lines, a fast rise time pulse makes the L di/dt term large, which introduces 1) considerable back voltage, 2) air coupling noise. While a ramp increases the cycle time of the memory, it 1) introduces relatively little back voltage, 2) decreases air coupling noise, 3) gives greater uniformity of core outputs.

The constraints placed on the various drive currents are relatively minor:

1) Each of the set, reset and inhibit pulses must exceed the switching threshold of the core.
2) The amplitude of the inhibit pulses must be greater than or equal to the amplitude of the set pulse.
3) The set pulse may not start before or end after the inhibit pulses, and must terminate before the reset line is enabled.

There is no requirement that the set and inhibit pulses have the same shape: in fact, it is possible and desirable to apply essentially DC on the inhibit lines; the set pulse may be either a square wave or a ramp, as long as the other constraints are met. The desirability of DC on the inhibit lines will be discussed below in the section on noise cancellation. The reset pulse ought to be a ramp for the reasons advanced above. The relation of the core output to the ramp pulse is shown in Figure 7.

Since the selection scheme associated with rope memories is such that one and only one core is not inhibited from switching during the selection phase and only that core is reset during the read phase, the core may be overdriven in order to increase operating cycles. This will also increase the core output. Clearly, power consumption is increased too.

No general statement can be made regarding the amplitude requirements of the various

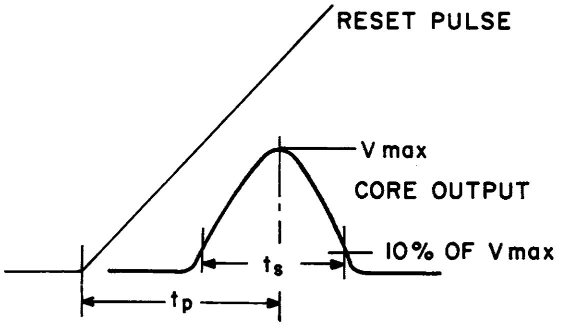


Figure 7. Relation between the core output and the ramp used to generate it.

pulses. As is true in regular coincident current or linear select memories, not all rope memories are made with the same type of core and the magnitude of the threshold switching field is clearly a function of the core used. One vital tradeoff appears in choosing a core for a rope memory: the storage capacity, the required drive currents, and the output voltage are a function of core size. The storage capacity and the required drive currents are directly proportional to the inner diameter of the core; hence, the drive currents increase as the number of bits to be stored increases. It thus becomes axiomatic that for a given required capacity the smallest core making the memory manufacturable is to be selected as the storage element.

Outputs and Noise Cancellation—The output of a particular rope for given drive conditions was given above. This magnitude output is typical for the given slope and memory capacity. The memory capacity governs the output insofar as the length of the sense lines, and hence their attenuation, varies in direct proportion to the number of cores common to a set of sense lines.

Noise problems in rope memories can be rather severe. Fortunately, there are two very simple methods by which noise can be minimized to give S/N ratios of about 10:1. Before discussing these schemes, the causes of the noise in the rope will be discussed.

In the first instance, noise is introduced in the sense lines due to the air coupling between the sense lines and the drive lines. This component is proportional to L di/dt. Secondly, during reset time the $(2^n—1)$ unselected cores are driven into saturation. This contributes an amount of noise which is proportional to $(2^n—1)\,\frac{d\phi_k}{dt}$ where ϕ_k is the flux excursion between the remanent and saturation points of the core. This noise can be many times the output of the selected core.

The air flux coupling can be reduced by doubling back the sense line causing it to return at its starting point. This doubling back minimizes the area in which coupling can occur. The noise flux can be cancelled by providing sense line cancellation and/or by leaving the inhibit lines on during the read phase as is suggested in Figure 4b, i.e., by using essentially DC. Noise line concellation is provided in such a manner that one-half of the unselected cores threaded by a given sense line are threaded in a sense opposite to that which is given to the remaining half of the linked unselected cores. This will, in the worst case, leave one noise output uncancelled. Clearly, each sense line will have a unique cross-over point. Leaving the inhibit lines on during the read phase of the cycle does not interfere with the resetting of the selected core; it will, however, minimize the $d\phi_k/dt$ term. This mode of operation is defined as "damped". The outputs of a rope in the damped and undamped modes of operation are shown in Figure 6.

Rope Storage Elements and Speeds—Basically, two types of storage elements can be used in rope memories: bobbin cores and ferrites. The ferrites may be either square loop or linear. The advantage in using bobbin cores is that they require lower drives than ferrites, are relatively insensitive to large variations in the thermal environment, and give higher outputs per unit of area due to their higher flux density. Ferrites offer a wider opportunity for experimenting with different geometries and reducing parts of the manufacturing technique to a batch fabrication process[9].

The ropes we have manufactured have been made with bobbin cores exclusively in order that they might meet required environmental specifications. It is possible to operate bobbin cores in a temperature range extending from —70°C to +150°C. Using bobbin cores, we feel that the cycle times can at best be reduced to 6 μsec without resorting to overdriving. Using linear ferrites in novel geometries, cycle times of 2-4 μsec are attainable.

Rope Memory Applications

The function of data storage is common to all permanent memories. The data to be stored may typically consist of programs, constants, supervisory routines or abstracts; e.g., a capacitative memory developed by the Academy of Sciences of the USSR is used to store abstracts of scientific literature[14]. Rope memories are clearly capable of performing these functions. The decision to use or not to use rope memories in these applications must then be based on such factors as cost, immunity to environment, speeds, selection techniques, and circuit requirements.

The rope has a built-in feature which is lacking in other types of permanent memories and which gives it a flexibility that extends its usefulness considerably. The rope is essentially a coding switch, and may thus be used to perform logic and lookup functions in a very efficient manner.

The use of ropes to perform logic (microprogramming) is straightforward and has been discussed by Walendziewicz in his paper on the Burroughs D210 Magnetic Computer[21]. In that computer, ropes are used both to perform logic and as fixed storage devices. The computer itself is used as a control element in various space and missile applications. Yii[23] has proposed and put into practice a scheme whereby every minterm in a Boolean function is represented by a core in a rope, and a wiring table is generated by negating the function and then applying De Morgan's Theorem. Using Yii's scheme, complements of the inputs are not always required.

The use of a rope memory as a dictionary in automatic language translation has been proposed[8]. If a source word is encoded in a standard six-bit code, the encoded source word will automatically select one core which can be made to store the target word. This overcomes the necessity for conducting a search. In this application, the D and C lines represent the encoded source word and its complement. If in addition to the target word, the complemented target word is also wired into the cores, the rope may serve as a two-way dictionary, e.g., Russian-English and English-Russian.

Rope memories can be used to effect information retrieval and association. For information retrieval purposes, a rope could operate in the one-core-per-word mode. The descriptors can be encoded in binary format. The binary coded descriptor then uniquely addresses a given core (or cores, in the case of multiple entries). Each core in turn stores the location of the required document. Content addressing can also be realized with rope memories by interchanging the role of the sense and inhibit lines. The sense lines corresponding to bits used as the associative criteria are driven during the selection phase of the cycle. The particular combination of sense lines will cause all those cores storing information in accord with the search bits to be switched. During read time the addresses of all cores so switched can be determined by linking each core to a detector.

ROPE MEMORY MANUFACTURE

Rope memory manufacture is unique in that 1) no batch fabrication techniques are possible at the moment, 2) with the exception of the set, reset, and inhibit lines, there is no regular wiring pattern, 3) the test of the final assembly must establish that all cores are within specification at the end of the manufacturing process and that the information wired into the rope is correct.

Up to the moment, we have used only discrete cores as storage elements in rope memories. In the near future. we hope to be able to produce storage elements on a batch basis, using ferrites in conjunction with thick magnetic films[9].

The main labor involved in the manufacture of a rope memory is in the wiring. This process has been automated to the fullest extent possible in order to reduce unit costs and to minimize the chances for human error. Rope memories are manufactured in modules of 256 or 512 cores. Initially, the set, reset, and inhibit lines are wired in. At this stage, the rope is tested to see that these lines were wired in correctly. This is done by driving each core in turn with a single turn winding and monitoring the output on the inhibit lines to determine whether or not a given inhibit line threads the core. After the rope passes this test, it is obvious that each core can be selected properly. The next step consists of wiring in the sense lines. For each sense line, there are as many thread, no-thread decisions as there are cores in the rope. All the thread, no-thread decisions for a sense line for 256 cores are encoded on a 90-column card. This card is used not only to govern the wiring phase of the manufacturing process, but also to test the correctness of the wiring. After each sense line is inserted, each core is addressed in turn and its output monitored across that sense line. The output is compared to the information stored on the 90-column card. The sense line tester is shown in Figure 8. (The details of the manufacturing equipment and processes are the subject of a paper presently being prepared by the personnel responsible for its design.)

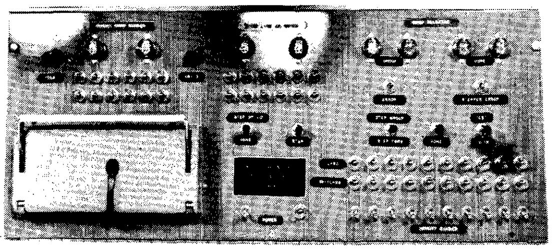


Figure 8. Sense line tester.

At the completion of the memory assembly, the rope is tested again. This final test consists of again testing each sense line as described above and/or reading out each word (core) in turn and comparing the wired word output against the desired word output, which is also stored on a 90-column card. Each rope is also tested to insure that the outputs of the whole assembly of cores are consistent with one another; i.e., that the cores meet their specification. This latter test is the third test that the cores are subjected to: they are first tested in a "raw" state to determine if they meet specifications, and they are again tested after mounting on the assembly matrix prior to being wired.

Packaging techniques are primarily dependent on the connector type used. Generally, ropes should be pluggable, which demands that a rugged connector be used. The choice of the connector is dependent on the size of the rope (256 or 512 cores) and the maximum number of sense lines. Our experience with packaging cores suggests that there are no difficulties in encapsulating a complete rope to protect it against environmental stresses.

ROPE MEMORY SYSTEM

As mentioned above, a number of rope memory systems have been built and placed into operation. A rope presently being designed and constructed is outlined in block diagram form in Figure 9. The purpose of this section is to briefly outline the approach used in the design of this system.

The description of the rope memory system under construction is as follows:

STORAGE CAPACITY	256 words, expandable in modules of 256 words. Maximum number of bits per core: 64. Provision will be made for storing up to four words per core.
ADDRESS INPUT	Either serial or parallel. Only "true" address needs to be furnished. An eight-bit address input is required if one core stores one word. If cores store four words, a ten-bit address is required.
INFORMATION OUTPUT	Either serial or parallel.
STANDARD LEVELS	0 and -6 volts.
CYCLE TIME	8 μsec.
POWER REQUIREMENTS	115 v AC, 60 cycles. Actual power will depend on memory configuration.
MECHANICAL	19″ rack mount. Minimum height including power supply: 21″. Rope memory will be interchangeable.
VERIFIER	A built-in verifier will allow each word to be addressed manually so that word content can be verified. Provision will be made for testing the memory under worst case drive conditions.

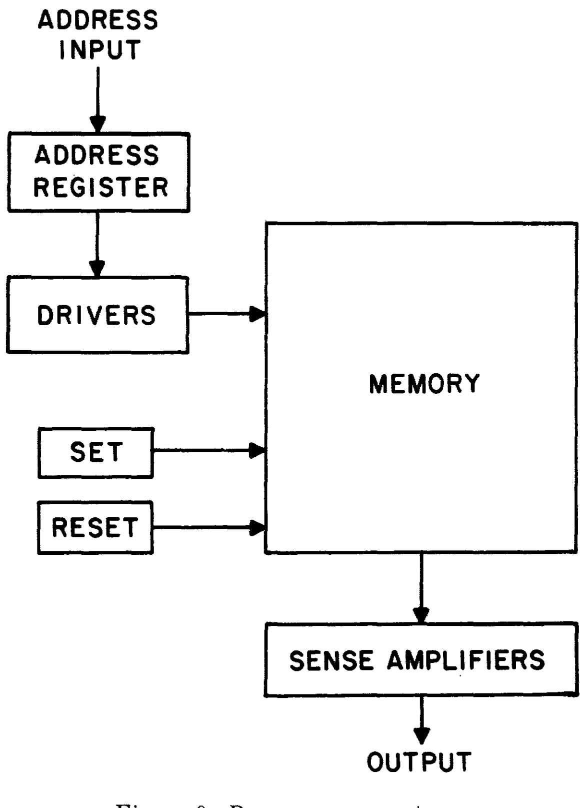


Figure 9. Rope memory system.

The eight-bit address of the core to be selected is the input to the rope memory system. The complement of the address is generated internally. If four 16-bit words are to be stored per core, giving a total storage capacity of 1024 words, the address must consist of ten bits: the eight low-order bits select the core, the remaining two bits are used to identify which of the four words stored by that core are to be read out.

Inspection of Figure 9 shows that no decoding circuits are necessary to select a core. All the information required for selection is contained in the address and its internally generated complement.

Selection may be accomplished in one of three different ways: 1) by having one driver associated with each of the direct and complemented address register bits and enabling those drivers whose corresponding bit position in the register is a "1"; 2) by using eight drivers and steering the current through the appropriate inhibit lines by means of switches controlled by the address register; 3) by using sixteen switches and activating eight of these to switch the proper inhibit lines from a zero level to a DC level able to generate the proper inhibit signal. The system under construction uses the last mentioned scheme. In addition, a set and reset driver are provided. The only restriction on the inhibit currents is that their amplitude be greater than that of the set pulse during the time the set pulse is turned on; no restrictions are placed on the inhibit current waveforms. A damped mode of operation is used.

The output is fed into a flip-flop shift register. If four words are stored per core, the selected word will be gated to the output register by using the information contained in the additional two bits of the address.

The block diagram of a memory system utilizing an electromagnetic coupling type permanent memory[3] is shown in Figure 10. The block diagram has been simplified for comparison purposes. A system of this type is inherently faster than a rope memory system since it depends on a coupling response proportional to di/dt and not on the switching time of a magnetic core. Since the output is dependent on di/dt, the rise time of the current pulse must be carefully controlled; in the rope system such careful control is not required as explained above. The input to that system is a nine-bit code. Six bits are used to select one of 64 channels. The remaining three bits are used to select one of eight words stored per channel—this corresponds to selecting one of four words stored per core in the rope memory system. There are 41 bits per word. The selection is accomplished by fragmenting the input as shown, and decoding each of the three segments

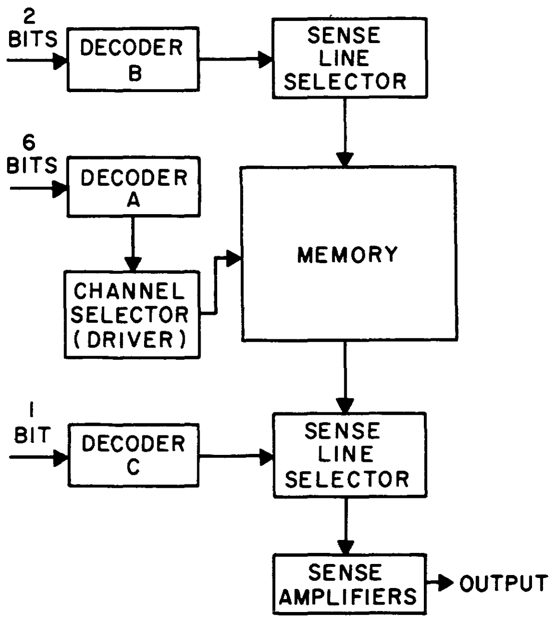


Figure 10. Electromagnetic coupling type permanent memory system.

in order to select the proper word. The address must be decoded as in any linear select scheme, a step not necessary in the rope memory system.

Due to the unique selection scheme employed by the rope memory and the fact that the tolerances on the inhibit drivers are very loose, together with the relative simplicity of the sense amplifiers due to the high output levels of the cores, the component count in such a system can be reduced by from 30% to 50% as compared to systems using other permanent memory schemes.

SUMMARY OF CHARACTERISTICS

A summary of various characteristics of the permanent memory schemes considered is given in Table III. Unfortunately, certain critical data is available only for the rope memory, and while it can be guessed what the comparable data is for other memory types, a comparison made on such a basis cannot be considered objective. For example, no data is available on the cost per bit of the various memory types discussed, nor is power consumption mentioned. In the rope memory, both cost per bit and power consumption are a function of core type used. Generally, as the flux is increased (for bobbin cores) both price and power consumption increase. Within limits, the cost per bit of a rope memory is the same as the cost per bit of a coincident current memory. Power requirements have been discussed above. Storage density in rope memories is between 500 and 800 bits per cubic inch.

The cycle times of all but the capacitative type memory appear capable of some improvement. As mentioned above, the use of linear ferrites as storage elements should make it possible to operate rope memories with a cycle time of 2-4 μsec. With regard to peripheral circuitry necessary to operate the various memories, the rope offers the greatest simplicity due to its unique selection scheme.

By adopting special packaging techniques, the addition of a limited number of data words is feasible in rope memories. Information can always be deleted by shorting out the unwanted core with a single-turn winding; up to ten new words can probably be put into a 256 core rope

TABLE III

SUMMARY OF CHARACTERISTICS

Permanent Memory Type	Changing Data Feasible?	Mechanical Alignment	Cycle Time	Output	Magnetic	Automatic Reset Required	Access	Bits Per Storage Element	Addressing
ROPE	NO	Not applicable	6-10 μsec	100-200mv	Yes	Yes	Random	64+ depending on core	Null decoder
CAPACITATIVE	YES	Critical	200 μsec	1mv	No	No	Random	1	Linear Select
EM COUPLING	YES	Critical	1 μsec	50mv*	No	No	Random	1	Linear Select
PM TWISTOR	YES	Critical	5 μsec	4mv	Yes	Yes	Random	1	Linear Select

* This value is computed and does not take attenuation into account.

to replace information deleted in this manner. While the other schemes allow wholesale alteration of information, the ease with which this can be done must be somewhat tempered by the mechanical alignment which is critical.

ACKNOWLEDGEMENTS

The work done with rope memories which forms much of the basis of this paper has depended on the cooperation of a number of people. It is with pleasure that I acknowledge the efforts of Messrs. W. Hennessey, A. Dorn, D. Pilsetnieks, and D. Clemson, all of the Electronic Instruments Division of the Burroughs Corporation. Messrs. Hennessey, Dorn and Pilsetnieks are responsible for the design of the rope memory manufacturing equipment and packaging techniques. Mr. Clemson is largely responsible for the realization of the test system, the coordination of the manufacturing procedure, and the work on the rope memory system described in the paper.

BIBLIOGRAPHY

1. ALONSO, R., and LANING, J. H., "Design Principles for a General Control Computer", (R276 Instrumentation Laboratory, M.I.T., April, 1960).
2. BARRET, et al., "A Card-Changeable Permanent-Magnet Twistor Memory of Large Capacity", *IRE Trans. on EC, EC-10,* 451, (1961).
3. ENDO, I., and YAMATO, J., "The Metal Card Memory, A New Semipermanent Store", *Large-Capacity Memory Techniques for Computing Systems;* ACM Monograph Series, Yovits, Ed., The Macmillan Co., New York, 1962.
4. FOGLIA, et al., "Card Capacitor—A Semipermanent, Read Only Memory." *IBM Journal of Research and Development, 5,* 67 (1961).
5. GIANOLA, et al., "Large-Capacity Card Changeable Permanent Magnet Twistor Memory". *Large-Capacity Memory Techniques for Computing Systems;* ACM Monograph Series, Yovits, Ed., The Macmillan Co., New York, 1962.
6. GETLER, W., "Space Role Seen for New Computer", *Missiles and Rockets, 12,* 31 (1963).
7. KRYLOW, et al., "Semipermanent Memory: Latest Use for Twistors", *Electronics, 36,* 80 (1963).
8. KUTTNER, P., "Magnetic Permanent Storage: The Rope Memory". Presented to the 1963 ACM National Conference.
9. KUTTNER, P., "Thick Film Memory", Internal Burroughs Report, A1-267, October, 1960.

10. LOONEY, D. H., "A Twistor Matrix Memory for Semipermanent Information", Proceedings, Western Joint Computer Conference, 1959.

11. MACPHERSON, D. H., and YORK, R. K., "Semipermanent Storage by Capacitative Coupling", *IRE Trans. on EC, EC-10,* 446 (1961).

12. MEYERHOFF, ed., *Digital Applications of Magnetic Devices,* p. 417, ff., John Wiley and Sons, New York, 1960.

13. MORGENSON, E. O., JR., "Wired-Core Matrix Memories", *Instruments and Control Systems, 35,* 75 (1962).

14. No Author Given, "Miniaturized Data Storage System", *Design News, 17,* 212 (1962).

15. No Author Given, "Core-Rope Memory", *Computer Design, 2,* 16 (1963).

16. RAJCHMAN, J., "Static Magnetic Matrix Memory and Switching Circuits", *RCA Review, 14,* 183 (1952).

17. RENARD, A. M., and NEUMANN, W. J., "Unifluxor—A Permanent Memory Element", Procedings, Western Joint Computer Conference, 1960.

18. RYAN, R. D., "A Permanent High Speed Store for Use with Digital Computers", *IRE Trans. on EC, EC-3,* 2 (1954).

19. SCHNEIDER, et al., "MADDAM—The First Completely Miniaturized Operating Computer", Proceedings, XII International Astronautical Congress, 1961.

20. TAKASHI and WATANABE, "Capacitance Type Fixed Memory", *Large-Capacity Memory Techniques for Computing Systems,* ACM Monograph Series, Yovits, Ed., The Macmillan Co., New York, 1962.

21. WALENDZIEWICZ, E. T., "The D210 Magnetic Computer", Paper presented to the Spaceborne Computer Engineering Conference, October, 1962.

22. WIER, J. M., "A High Speed Permanent Storage Device", *IRE Trans. on EC, EC-4,* 16 (1955).

23. Private discussion with R. YII, Electronic Instruments Division, Burroughs Corporation.

A 300 NANOSECOND SEARCH MEMORY

C. A. Rowland and W. O. Berge
*Univac Division of Sperry Rand, St. Paul, Minn.**

SUMMARY

This paper describes a 24 digit, 128 word, transistorized magnetic thin film search memory with a 300 nanosecond read cycle time. With further improvement such a memory could work with a 100 nanosecond cycle time. A search or associative memory is one in which an input word is compared simultaneously with all stored words; an output is produced on all words that differ from the input word. Variations of this memory permit search between limits by setting up a "don't care" condition for some of the stored bits; however, the emphasis in this paper is on the equality search.

The writing system described is capable of doing a complete rewrite of all 128 words in 13 milliseconds; much faster writing systems are possible. Applications are in sorting, cataloging, information retrieval, translation, and searching.

INTRODUCTION

A search memory is one in which an input word is compared simultaneously with all stored words. In a conventional memory one word line is selected and driven while all digit lines are driven and the word lines are sensed. A search may be made for equality or a limited type of between limits search may be made. The search memory described here utilizes thin films of cobalt-iron and of nickel-iron (Permalloy). The cobalt film acts as the memory element and the Permalloy film acts as the sensor. The coercivity of the cobalt film is not so high that transistor writing circuits are out of the question. A search of all 128 words can be completed in 300 nanoseconds. Writing is relatively slow and must of course be done one word at a time; approximately 13.6 milliseconds are required to write all 128 words in the particular model described here. Writing times of 4 microseconds/word are feasible.

Theory of Operation

The search memory described here utilizes BICORE non-destructive readout memory elements. The BICORE element is a sandwich of a high coercivity cobalt-iron film and a low coercivity permalloy film. Such a film pair has been described in various publications.[1, 2, 3, 4] The demagnetizing field of the high H_c film (cobalt) biases the low H_c film (Permalloy) to one of two saturated conditions. If the applied read field aides the demagnetizing field of the high H_c film a small signal will be produced by the low H_c or sensing film. A pictorial representation of the operation is shown in Figure 1.

A small transverse field aids in producing rotational high speed switching of the sensor or low H_c film. The read bias field (mostly transverse) can be adjusted to optimize the "1" to "0" ratio. The total read field must be small enough so that the data film does not creep (i.e., gradually demagnetize) and it must be large enough to produce fast switching of the sensor film. The following are the approximate parameters of the film elements used:

*Sponsorship of this work was under the Bureau of Ships.

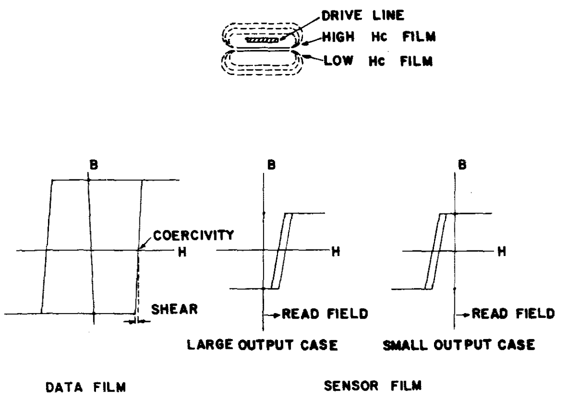


Figure 1. Theory of operation.

	Permalloy	Cobalt Iron
Thickness	2500 A°	2500 A°
Coercivity	1.1 oe	30 oe
Anisotropy	5.0 oe	40 oe
Shear Field	2.0 oe	4.4 oe

The optimum read bias field and read drive field are interrelated and are a function of the rise time of the read drive pulse. It was found experimentally that the best results for a 24 digit search memory were obtained with a transverse bias field of about 5 oersteds and a read drive pulse producing about 4 to 7 oersteds longitudinal drive with a 10 nanosecond rise time. These conditions result in a fast rotational switch of the Permalloy film. The magnitude of both the transverse bias field and the longitudinal read field is less than 25% of the magnitude of the cobalt film coercivity; this is well below the creep region for the data film (cobalt).

In order to produce a satisfactory non-coincidence to coincidence signal ratio for a 24 digit search memory in which all digits are examined simultaneously it is desirable to use two NDRO or BICORE memory cells per bit. The second or dummy cell is always set to produce a small output and thus is used to cancel a small output from the active cell. In order to produce an output for both 1's and 0's in the case of anticoincidence it is necessary to have a complement non-complement memory. The use of a complement non-complement memory permits the search memory to be set to produce an anticoincident (large) signal for either a "1" input or for a "0" input. Also particular bits of a given word may be set to a "don't care" condition where neither a "1" or "0" input produces an output, i.e., both cells produce a small output; an ignore word condition may be set by producing an output for both 1's and 0's for one or more of the digits. The "don't care" condition is useful for searching between limits. The "don't care" condition may be stored as mentioned or it may be obtained by not pulsing the digits where "don't care" is desired. A type of between limits search may be made by examining the high order digits and ignoring the low order digits. Then, a hit on 01011XX would indicate that the numbers producing the hit were greater than or equal to 01011XX but less than 01100XX.

A total of four BICORE memory cells are required for each bit of memory. An arrangement of memory cells for one bit of search memory is shown in Figure 2. The digit lines are the driven lines and the word lines are the sense lines. The dummy cells are always set to produce a small cancelling output. For an equality check one of the active cells is set to produce a large anticoincidence signal and the other is set to the complement and thus produces a small output. If the input word being searched for contains a "1" in a given digit the "1" line for the digit is driven and the "0" line is not driven. Then if a stored word contained a "1" for this digit a negligible output would be produced from this digit for this particular word; however, if the stored word contained a "0" a large output would result when the "1" digit line was pulsed. Because of the good anticoincident to coincident signal ratio one anticoincident signal is roughly 4 times the amplitude of 24 coincident signals so that amplitude discrimination in the sense ampli-

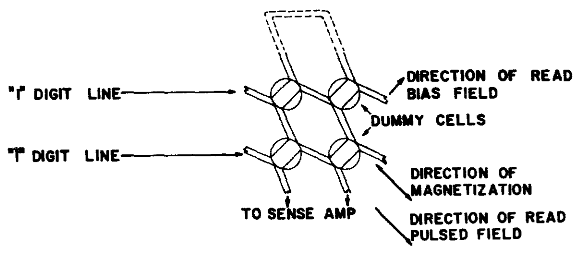

"1" DIGIT LINE
"1" DIGIT LINE
DIRECTION OF READ BIAS FIELD
DUMMY CELLS
DIRECTION OF MAGNETIZATION
TO SENSE AMP
DIRECTION OF READ PULSED FIELD

Figure 2. Arrangement for one bit including complement non-complement feature.

fier is relatively simple. The search takes place on all words simultaneously and a sense amplifier is required for each word.

Physical Characteristics

The magnetic thin film memory elements are .035 inches in diameter and are spaced on .07 inch centers. The word and digit lines follow a zig zag path as indicated in Figure 2. These lines are .021 inch wide over the films and .012 inch wide between the films. The drive and sense lines return under the .006 inch glass substrates on which the films were deposited. Each film substrate contains 32 x 24 films; thus 16 substrates are required for a 128 word, 24 digit search memory. Eight substrates are placed on each of two memory arrays. One memory array is shown in Figure 3A; each array is 4¾ in. by 11½ in. overall without the wiring fanouts and 7½ by 13⅜ in. overall with the fanouts. Figure 3B shows a closeup of the array and one substrate not aligned.

The characteristic impedance of the digit and word lines are approximately 50 ohms. The propagation velocity is approximately 4 inches per nanosecond.

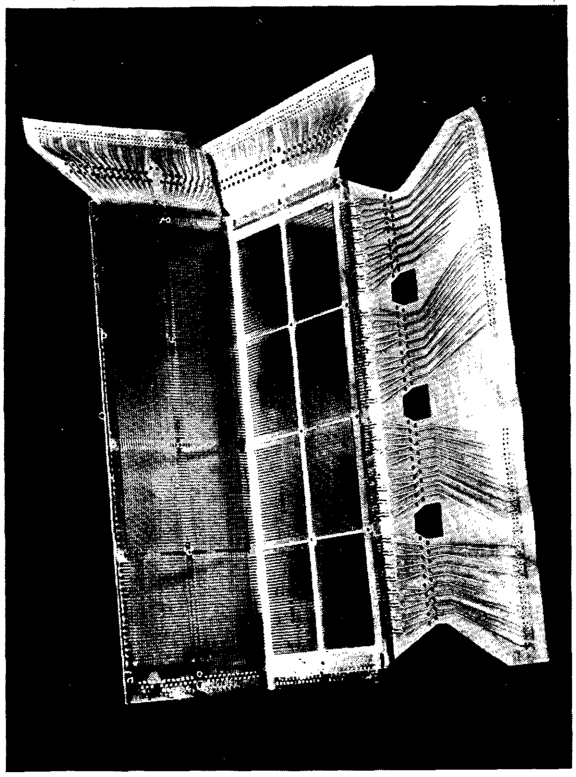


Figure 3A. Memory array.

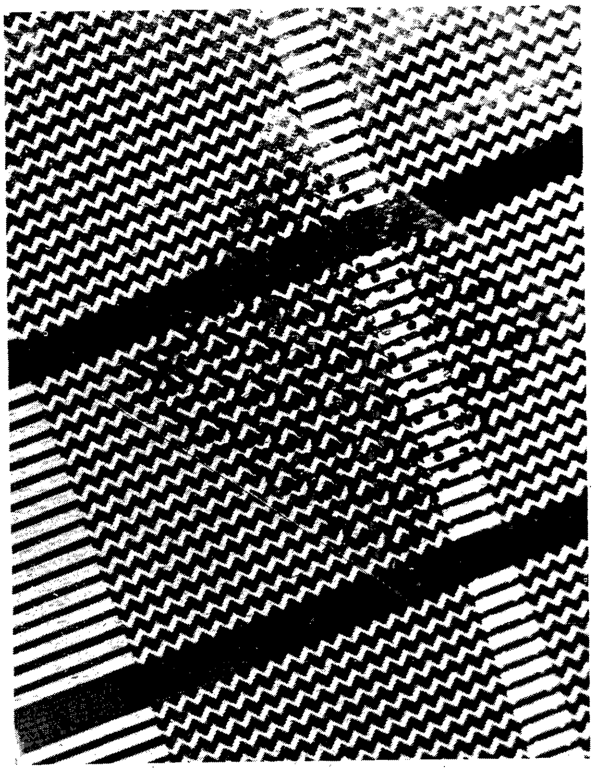


Figure 3B. Close up of memory array and one substrate.

In order to provide a means of providing read bias field, write bias field and an erase field, two bias coils surround the two layers of substrates and the two memory arrays. These coils are perpendicular to each other so both direction and magnitude of the bias field can be controlled. The bias coils are single layer etched coils having 10 turns per inch. The overall dimension of the bias coils are 6½″ × 13¾″ × 9⁄16″ and 7⅜″ × 13¼″ × ⅝″.

The following circuit elements are required for the 128 word, 24 digit search memory:

48	Digit Read Drivers
48	Digit Write Drivers
128	Sense Amplifiers
	Word Write Circuits
	Bias Circuits for Erase, Write, and Read
	Address Encoding
	Address Decoding
	Read Control
	Write Control

Figure 4 shows: (1) a module containing one digit write driver (48 required), (2) a module

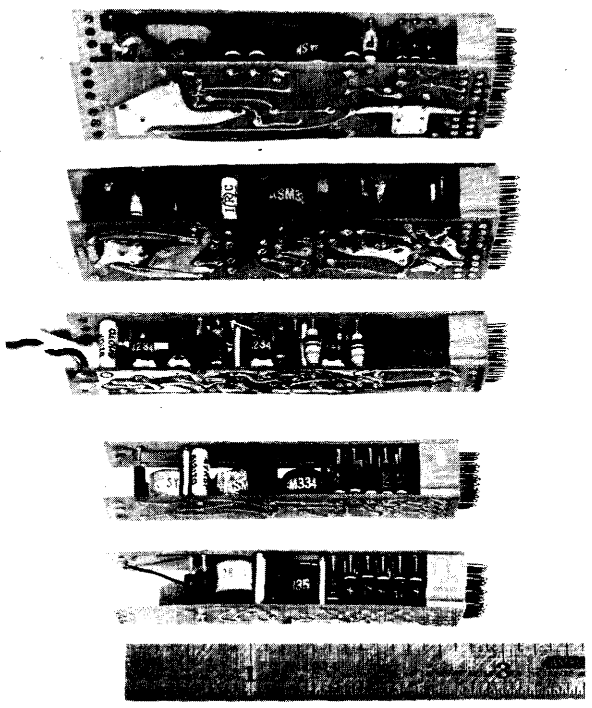


Figure 4. Photo of memory circuit modules

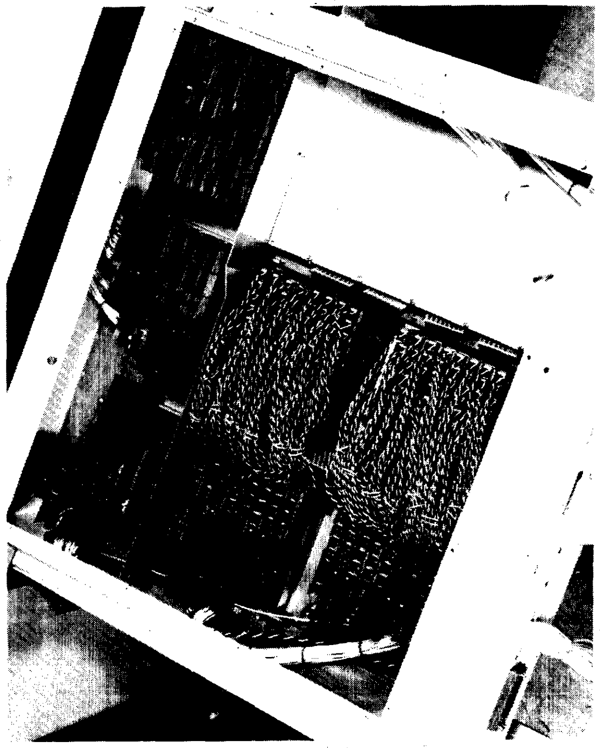


Figure 5. Photo of complete Search Memory.

containing a complement and a non-complement read driver (24 required), (3) a module containing a sense amplifier (128 required), (4) a word write pulse generator (8 required), and (5) a module containing a word write switch (16 required). Figure 5 shows the complete 128 word search memory including all associated logic circuits. The complete search memory measures 26 in. $\times$ 24 in. $\times$ 7 in.

Digit Circuits

The read driver must be capable of supplying 0.3 amp. pulses with less than 10 nanoseconds rise time and 30 nanoseconds duration into a 50 ohm load and be capable of operating at any repetition rate up to 10 megacycles. The write digit driver must supply a 2 amp. pulse of about 2 microseconds duration into an R-L load of approximately 2 ohms and 3 microhenries "average" inductance at a 10 kilocycle rate; the inductive load is mainly the average of the inductance of the read transformer which saturates during the write pulses.

The digit circuits and interconnections are shown in Figure 6. The read driver contains

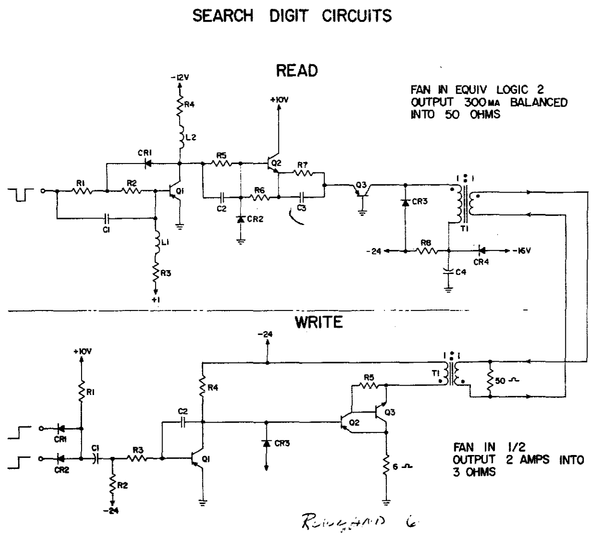


Figure 6. Schematic of digit circuits.

three stages—a grounded emitter PNP stage followed by an NPN grounded collector stage followed by a PNP grounded base stage. The read driver is not designed to be on 100% of the time; to prevent damage due to accidental turn on of the read driver, capacitors C_1 and C_2 and resistor R_8 are chosen so that no damage to the driver results from accidental turn on. The write driver is an NPN saturated inverter followed by a complementary Darlington circuit. The —12 clamp voltage on the first inverter and the size of R_7 determine the approximate digit write current.

The digit lines are floating so that when reading, the capacitive coupling between the digit lines and the word lines will be roughly balanced out. The digit and word lines cross the films on top and return underneath with the read pulse being positive going on top and negative going underneath. The fact that the capacitance between the lines at the top crossing and at the bottom crossing is approximately equal produces a nearly balanced condition. Since the word and digit lines are parallel over the films it is also necessary to balance out the inductive coupling; the dummy line is convenient for balancing out this inductive coupling between the digit lines and the sense lines.

Word Write Circuit

The word write circuitry consists of a diode-transformer matrix, 8 pulse current generators and 16 saturating switches. Each intersection of the word write matrix is included within a

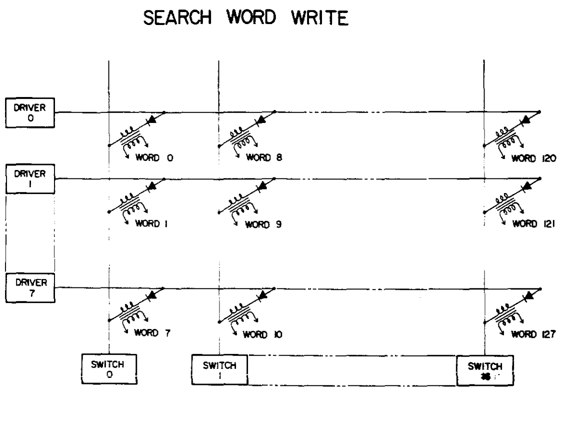


Figure 7. Matrix arrangement for word-write.

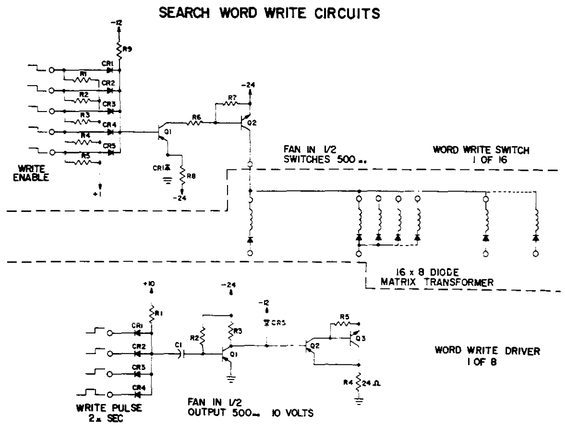


Figure 8. Word-write circuits.

sense amplifier module. Figure 7 shows the matrix arrangement. For writing into the word lines the current required is 2 amps. into a resistive load of 0.6 ohms.

The current-pulse-generator circuits and the switch circuit used for generating word write current are shown in Figure 8. A 4:1 current step up transformer is used at each matrix intersection so it is necessary for these circuits to generate and switch 0.5 ampere to produce 2 amperes of write current. The pulses used have a duration of 2 microseconds. A diode AND circuit is included on the inputs to the pulse current generator and the switch; these AND circuits are used for address translation and to inject the 2 microsecond write pulse. The current generator is similar to the digit write driver in that an inverter with a clamped output is used to drive a complementary Darlington pair of transistors. The switch circuit is simply a PNP grounded emitter stage followed by an NPN grounded emitter stage. Capacitor C_1 is used to prevent damage to the pulse current generator by accidentally leaving the circuit on 100% of the time.

The write circuits for this model were designed to write in one direction only for simplicity and because there was no emphasis on writing speed. The writing is accomplished in the following manner:

(1) First all cells are erased by a large current (about 6 amps.) through each of the bias coils. This sets all cells to produce a small output so that all bits are set to the "don't care" condition.

(2) The erase field is removed and a write bias field is applied by sending about 2 amperes of current through each bias coil in the same direction as the erase current.

(3) Then writing is accomplished by applying write currents to one word at a time by turning on one word write switch, one word write driver, and either the complement or non-complement write driver for each of the digits. The fields applied by the digit and word lines are each ⅔ of that necessary to write. The bias field is —⅓ of the field required to write.

The writing system described here is relatively unsophisticated in that the whole memory must be rewritten each time a change is made. Schemes to selectively alter one word can be devised. With the circuits used here having very little emphasis on writing speed, an erase requires 0.5 milliseconds, 0.3 millisecond is required to establish the write bias, and writing each word requires 0.1 millisecond. The total write time for 128 words is 13.6 milliseconds. Selective writing at 4 microsecond intervals is feasible with circuits designed to operate at the higher duty cycle.

Sense Amplifier

The sense amplifier must have the following capabilities:

1. A very good common mode rejection.
2. Ability to discriminate one anticoincident signal from 24 coincident signals.
3. Ability to recover from 24 anticoincident signals in 100 nanoseconds.

The circuit used is shown in Figure 9. To provide common-mode rejection the input is through a common-mode transformer into a differential amplifier. A finer balance is obtained by adjusting the variable resistor on the input to the amplifier. The second stage of the sense amplifier has enough dynamic range for 24 anticoincident signals. This stage feeds into a back-to-back backward diode circuit which rectifies the signal and attenuates the large signals. Because the signal is bipolar and because the backward diode circuit is fairly symmetrical very little bias shift is encountered. The backward diode circuit feeds an emitter follower which is followed by the final inverter stage. The memory output pulse is provided for the anticoincident case although it would be more desirable to provide an output for the coincident case; an output for the coincident case can be provided by using the sense amplifier output to inhibit a strobe by using a 2-input logic AND circuit.

Waveforms

Figure 10 shows typical outputs at three different points on the sense amplifier: (1) Clipped output of 2nd class A stage, (2) Input to the 3rd stage emitter follower, and (3)

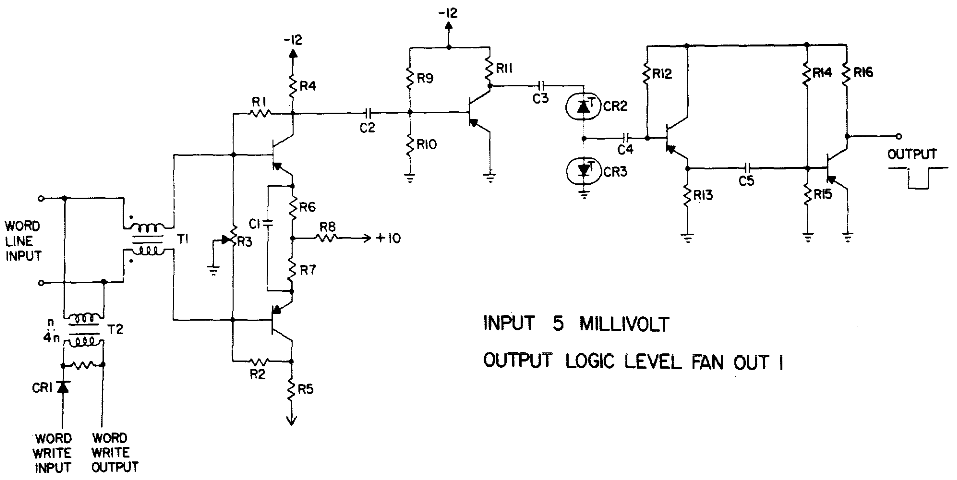


Figure 9. Sense-amplifier schematic.

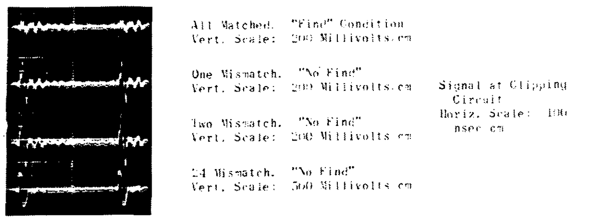


Figure 10A. Waveforms at backward diode clipping circuit.

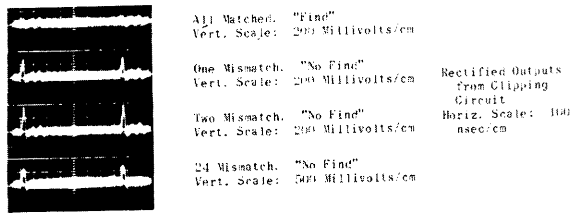


Figure 10B. Rectified and clipped outputs from backward diode clipping circuit.

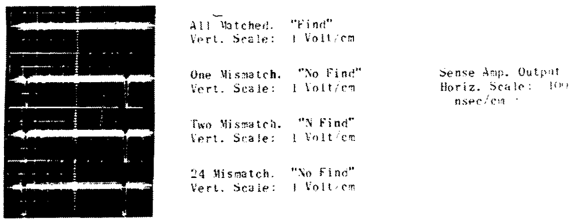


Figure 10C. Outputs from sense amplifier.

Sense amplifier output. The outputs are shown for the following conditions: (a) matched, (b) one mismatch, (c) 2 mismatches, and (d) 24 mismatches. Ringing on some of the word lines prevents operating at the maximum rate of 10 megacycles. However, with a little more improvement operation at this rate will be possible.

Applications

Applications of search type memories are in sorting, cataloging, information retrieval, translating, and searching. A specific application is in radar track correlation. A search memory can compare a radar return with all the tracks stored in one search time which may be as fast as 0.1 microsecond. Automatic abstracting is another specific application for search type memories. In this case the search memory is used as a dictionary that contains a list of key words. The contents of the document are compared with this dictionary and if any of these key words are found in the document, they are placed in an abstract.

CONCLUSION

The BICORE thin-film memory cell is a practical device for making search or associative or content addressable type memories. With presently available transistors a complete search on all words may be completed in 300 nanoseconds and with reduction in ringing of the word lines a 100 nanosecond cycle time should be realized.

REFERENCES

1. Oakland, L. J., and Rossing, T. D. "Coincident Current Non-Destructive Readout from Thin Magnetic Films." J. Appl. Phys., Vol. 30, Suppl., Pp. 545–555; April, 1959.
2. Petschauer, R. J., and Turnquist, R. D. "A Nondestructive Readout Film Memory." Proc. WJCC; May, 1961.
3. Pohm, A. V., and Mitchell, E. N. "Magnetic Film Memories, A Survey." IRE Trans. of Electronic Computers, Vol. EC-9, Pp. 308–314; Sept., 1960.
4. Raffel, J. I., Crowther, T. S. Anderson, A. H. and Herndon, T. O. "Magnetic Film Memory Design." Proc. IRE, Vol. 49-1, Pp. 155–164; January, 1961.
5. Turnquist, R. D., Hogenson, C. O., and Christensen, V. E. "A Compact 166-Kilobit Film Memory." 1962 IRE Convention.

A NEW TECHNIQUE FOR USING THIN MAGNETIC FILMS AS A PHASE SCRIPT MEMORY ELEMENT

Bruce Kaufman and Eduardo Ulzurrun
The National Cash Register Company
Electronics Division
2815 West El Segundo Boulevard
Hawthorne, California

INTRODUCTION

The use of thin magnetic film memories for high speed computers has been well established. More recently, efforts have been made to adapt thin-film memory elements for operation in a phase script mode by using drive fields at two different frequencies to switch and read the state of the memory element. The resulting memory system is uniquely suited for operation in a parametron computer where information exists as a phase state rather than an amplitude state.

Work by E. Goto at the University of Tokyo,[1] and a disclosure by W. E. Proebster of IBM[2] illustrate some of the techniques proposed. An alternating field, at half the subharmonic frequency of the parametron machine, is applied in the transverse direction. The simultaneous application of an appropriately phased sinusoidal field in the easy direction, at twice the frequency of the transverse field, will cause the magnetic vector to creep to the opposite state, or remain in its initial state, depending on the phase of the easy direction field.

WRITING OPERATION

If two sinusoidal and orthogonal magnetic fields at frequencies f and f/2 are simultaneously applied to the film, the resultant applied field

$$\vec{H} = \vec{H}_{\text{easy}} + \vec{H}_{\text{transverse}}$$

creates an imbalance that is able to switch the magnetic vector by creeping. Operation in this mode has the inherent limitation that many RF cycles are required to change the state of the magnetic film, resulting in a long write time. This was a major drawback in the "dual-frequency" operation utilizing ferrite cores, although the magnetic principles involved were somewhat different.[3]

It would be desirable to cause $\vec{H}$ to cross the switching threshold in one direction and to remain well below the threshold when it is reversed. This may be accomplished by increasing the amplitude and asymmetry of the transverse field, perhaps by adding some second harmonic in such a way as to cause a flat part on one half cycle and a peak on the other. However, there is still a problem of specifying and controlling tolerances of the creep threshold, as yet a requirement beyond the state of the magnetic film art.

In addition to these problems, there are serious circuit complications with the use of a sinusoidal word field in a dual-frequency memory. An economically feasible system requires some sort of matrix selection scheme for the word lines. Early ferrite core dual-frequency memories[3] required one word driver per word—a prohibitive expense for a large (>1,000 words) memory. A row-column selection system allowing linear select operation would be more desirable. The circuitry to provide sinusoidal word currents for such a matrix is very complex.

A new technique has been developed by the writers which utilizes appropriately timed unipolar pulses applied in the transverse direction. These are at the same rate as the accompanying sinusoidal field applied in the easy direction. This technique has the advantage that the pulses may be shaped and timed (Figure 1) so that the trajectory of the applied H-vector in the H_t-H_e plane has only one crossing point on the switching threshold curve as may be seen from Figure 2. This eliminates undesirable creeping. The obvious additional advantage is that, since only uni-polar pulses are required in the word line, simple row-column selection circuits (commonly used for a linear select memory) are suitable. Since pulses of only one direction are supplied to the word line, only one line isolation diode per word is required.

A major advantage of the new mode is that creep phenomena are not required for writing. Rotational switching occurs within the duration of one uni-polar word pulse. This eliminates one major drawback of previous dual-frequency memories—limited speed writing.

READ OPERATION

For sensing the state of the memory element, only the field in the transverse direction is applied. This operation is similar to that found

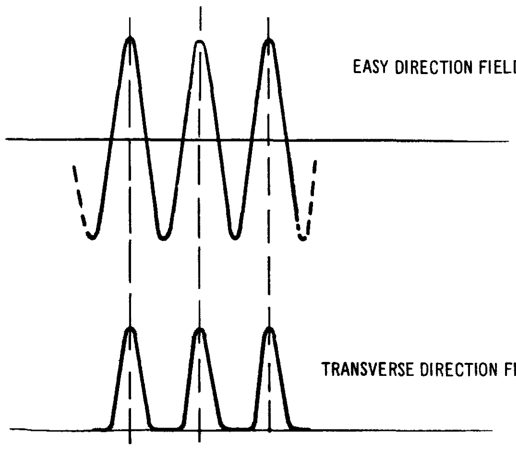


Figure 1. Unipolar H_t Pulse and Sinusoidal H_e.

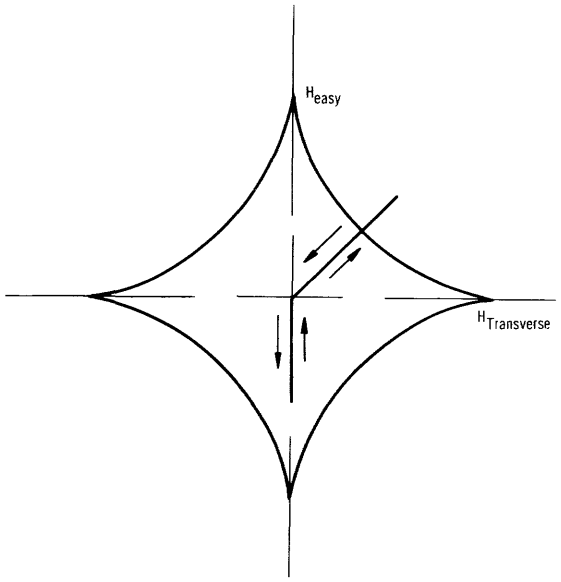


Figure 2. Locus of Applied Field During Write Cycle with Unipolar $\vec{H}_t$.

in conventional film memories. The read field may be the same uni-polar pulse required for writing. On the rise of the transverse field, the magnetic vector rotates clockwise or counterclockwise, depending on the direction of magnetization of the film (Figure 3). On the fall of the field, it returns to its original position. The flux change produces an output signal as in Figures 4a and 4b. It may be seen that,

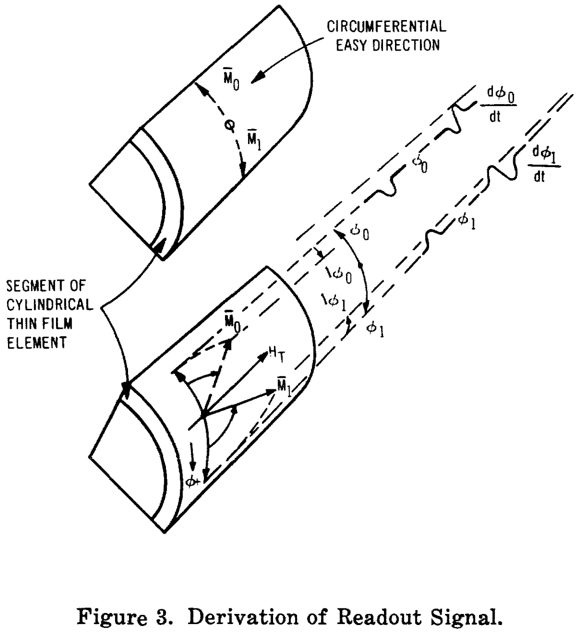


Figure 3. Derivation of Readout Signal.

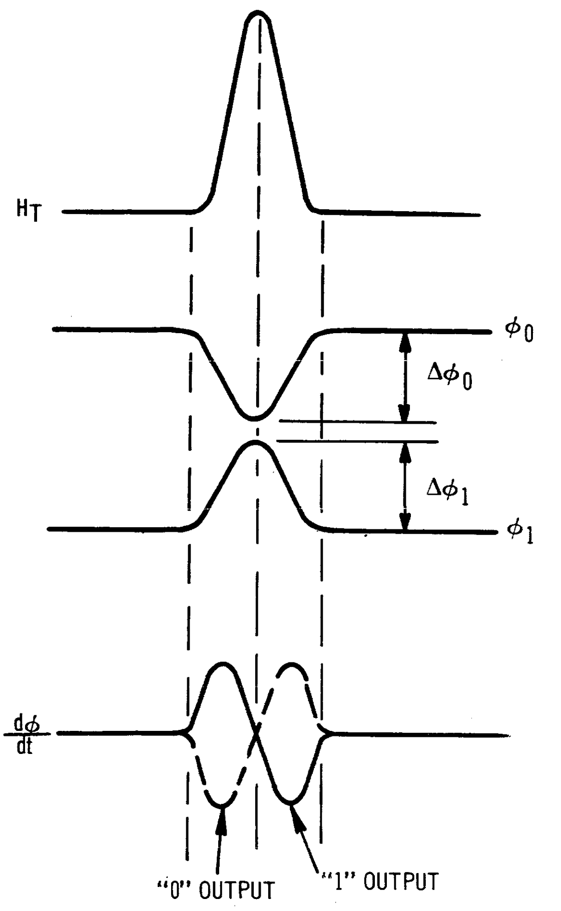


Figure 4a. Readout Signal Waveforms.

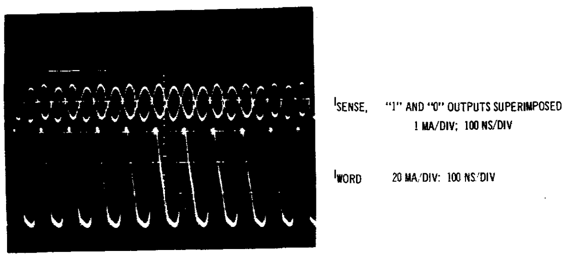


Figure 4b. Phase Script Readout Waveforms.

depending on the state of the film (1 or 0), the two outputs will be mirror images of each other. Since a train of uni-polar pulses is applied for readout, a train of pulse pairs at the same PRF appears on the readout line. It may be readily seen that this waveform contains phase information which may be used to lock a parametron operated as a sense amplifier.

It should be noted that this technique is equally suitable for the various physical forms of thin-film memory elements, i.e., films deposited on cylindrical wires with either an axial or circumferential easy direction, or on a flat film array. It is also noteworthy that this technique of operation preserves the inherent capability for NDRO operation since the read field is applied only in the transverse direction, and if it can be kept below the worst-case creep threshold of the film, nondestructive readout may be obtained.

One design requirement of the memory element is a certain degree of NDRO capability, since a finite time is required to lock a parametron sense amplifier to the desired phase. If creep, due to a transverse field, is possible, it must be such that the output signal is available for a sufficient number of read pulses (typically 10-20) to allow the sense amplifier parametron to lock into the required phase state.

USE OF COMMON PARAMETRON FOR SENSE AMPLIFIER AND DIGIT DRIVER

A major advantage of this type of memory is the potential use of a single parametron for both the sense amplifier and the digit driver. The operation is as follows: the flux change caused by the word current flowing in the word line at read time causes readout signals to appear on the sense line either in "one" phase or "zero" phase. These are then fed by an appropriate matching network to a parametron operating as a sense amplifier. When properly clocked, this readout signal forms the "seed" to lock the sense amplifier parametron and allow it to build up to steady-state with the phase of the readout signal. This parametron may be designed so that it can supply sufficient current into the digit line to automatically rewrite (in conjunction with applied word current which is simultaneously present) the information which had been read out. This action yields automatic read-restore operation, i.e., any read cycle automatically becomes a read-restore cycle, greatly simplifying the memory organization and giving, in essence, the system equivalent of NDRO operation without stringent creep and threshold requirements on the film memory element. However this technique poses practical problems because of a phase difference between readout signal and required digit current.

Japanese workers[3] had considered the above technique in early dual-frequency core memories; however, it was not commonly practiced because the optimum phase angle for the digit current normally developed by the parametron was different from that of the memory readout locking signal, and a phase correction was necessary. Separate parametrons for the digit driver and the sense amplifier, coupled by buffer parametrons to absorb the phase error, were usually employed.

The use of a single word current source, switched under address control to the various word lines (as used in a linear selection memory), allows a very simple arrangement to provide the phase correction. The uncompensated phase angles of the various signals may be seen in Figure 5. Notice that the fundamental component of the readout signal is at 90° phase angle (Figure 5d) with respect to the normal buildup phase of the parametron (Figure 5a). This is in the worst possible phase for locking the parametron and minimum sensitivity would result.

The readout signal must, therefore, be shifted by 90° before it appears as a "seed" signal for the parametron. While it is possible to shift the phase of each readout signal or of each writeback digit current, a much simpler arrangement is to apply the phase correction to the word current itself. This may be accomplished by delaying the burst of unipolar pulses used for readout by the necessary amount (25 nsecs at 10 mc) to compensate for the 90° phase shift. This has the effect of shifting the sense output 90°. The burst of word pulses for the write operation is supplied without the 25 nsec delay. This is preferable to operating on the digit currents since only two word current sources are necessary; one with the correct phase for read and a second with the correct phase for write, while a phase fixup in the digit circuitry would be necessary at each digit plane in the memory.

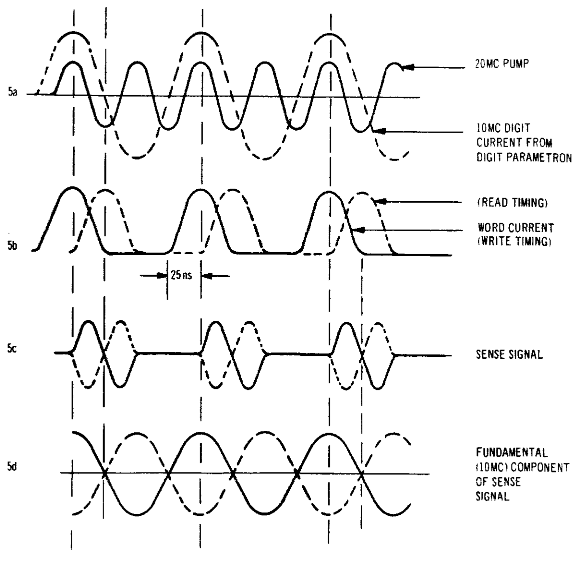

5a
20MC PUMP
10MC DIGIT CURRENT FROM DIGIT PARAMETRON
5b
(READ TIMING)
WORD CURRENT (WRITE TIMING)
25ns
5c
SENSE SIGNAL
5d
FUNDAMENTAL (10MC) COMPONENT OF SENSE SIGNAL

Figure 5. Timing Relations for Read-Out Phase Correction.

MEMORY STACK DESIGN

Considerations of demagnetizing fields and their effect on the threshold characteristics of the memory element led to the use of a cylindrically-oriented permalloy plated wire. To produce the plated memory elements, a 10-mil beryllium-copper substrate is electroplated in a continuous process with a 10,000 Å coating of cylindrically-oriented permalloy. The H_c of the plated wire (in the cylindrical direction) is approximately two oersteds. The transverse loop shows an H_k of approximately four oersteds, with good closure of the loop to saturation.

Memory organization with a cylindrical easy direction requires that the word current generates an axial field. An efficient approach is an array of very small solenoids of the type developed previously for the Rod memory.[4] The memory stack thus consists of planes of encapsulated solenoids stacked together, with the plated wires inserted into the solenoid apertures, and connected end to end in an appropriate noise-cancelling arrangement. The plated wires themselves become the digit lines. Readout is obtained from, and digit drive is applied directly to, the plated wire lines.

The small diameter of the memory element (0.01″) requires $\approx$ 60 ma/oe for generation of a circular field (digit), which is easily provided by cylindrical thin-film parametrons.[5] The word solenoids require approximately 15 ma/oe to generate the transverse field. Figure 6 is a diagram of the stack arrangement, and Figure 7 is a photograph of a stack of 512, 26-bit words.

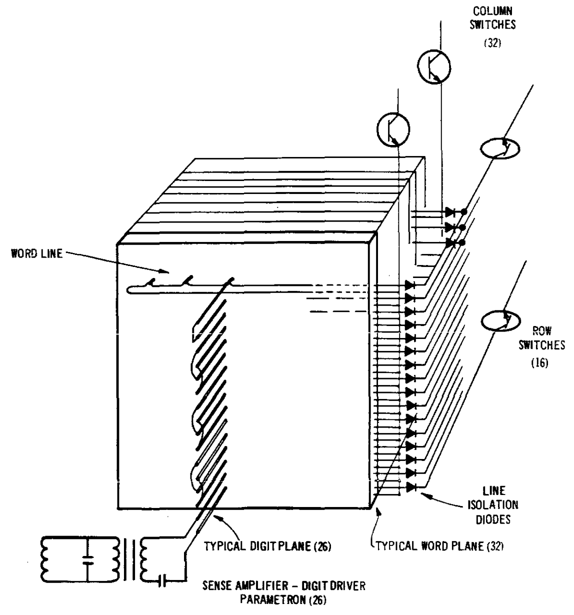


Figure 6. Stack Arrangement.

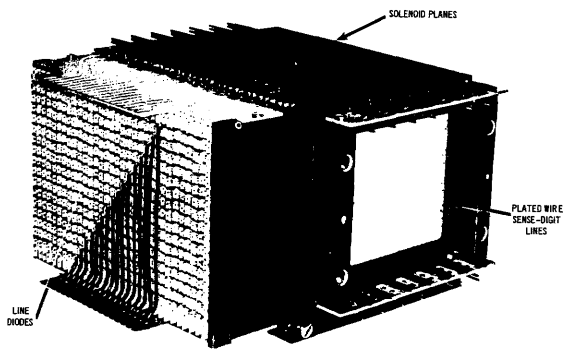


Figure 7. Phase Script Memory Stack.

The word lines are selected on a row-column basis, both ends of the lines being switched. To reduce loop inductance in the word line, a connection to the far end of the word line at the last solenoid is brought back directly under all solenoids of the same word. This places the row end of the word line close to the column end.

The sense-digit line must be treated as a transmission line, and phase errors resulting from reflections must be considered. With a conventional pulse memory, the sense line is normally terminated in its characteristic impedance. In this case the line has a resistive input impedance. An unusual advantage results from the use of parametrons as digit drivers. Since the digit current is truly sinusoidal, the line then may be treated on a single frequency basis rather than considering all spectral components of a pulse. The far end of the line may be short-circuited and a standing wave deliberately set up on the line. Since the line is short compared to the wavelength at 10 megacycles, the standing wave ratio is low (approximately 1.1:1). Thus, only a 10% difference exists between digit current in the near and far ends of the line. This has been shown to be within acceptable limits. The spatial current distribution is such that the current has the same phase angle everywhere on the line. This tolerance is perhaps more important than that of amplitude.

Since the line now has no terminating resistance to be driven, only sufficient power to overcome line losses is required. This greatly reduces (by an order of magnitude) the power necessary from the digit driver and allows the use of a simple parametron for this purpose. The inductive input impedance of the shorted line is tuned out with a capacitor, therefore a resistive load is presented to the parametron.

An interesting situation occurs when the digit line is driven in the manner described above. With reference to Figure 8, one may see that the digit current circulates in a loop, i.e., in the upper segment of the line, it circulates

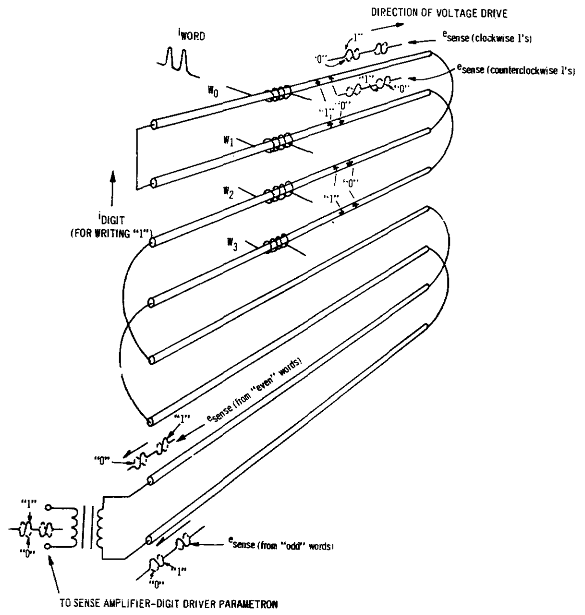

DIRECTION OF VOLTAGE DRIVE
i_{WORD}
"1"
"0"
e_{sense} (clockwise 1's)
e_{sense} (counterclockwise 1's)
w_0
w_1
w_2
w_3
i_{DIGIT} (FOR WRITING "1")
e_{sense} (from "even" words)
e_{sense} (from "odd" words)
TO SENSE AMPLIFIER-DIGIT DRIVER PARAMETRON

Figure 8. Digit Drive and Sense Output Polarity Relations on Sense-Digit Line.

in one direction, and in the next segment of the line, it circulates in the opposite direction. This causes opposite directions of the magnetic vector for storage of a "one" and a "zero" on adjacent pairs of plated wires. This is desirable if the input transformer for the sense-digit parametron is a differential transformer (required to reduce common mode noise). With reference to Figure 8, one sees that correct sensing of the output signal occurs and no compensation is necessary for the fact that the direction of a "one" (or "zero") alternates for "odd" and "even" words, as a result of the balanced sense-digit line.

Digit-plane to digit-plane coupling is often a source of difficulty in fast memories. A high degree of cancellation of this type of coupling is obtained in two ways: by sufficient spacing between adjacent digit planes and also by connecting the lines in such a way that the total flux induced in one line by the current in the other tends to cancel.

Four different line configurations (shown in Figure 9) are used in such a way that any pair of the four are non-interacting. By using groups of these four configurations, interaction occurs only between lines separated by four spaces, greatly reducing the effect. Each digit line, in turn, is balanced so that capacitive noise from the word lines is also cancelled by the differencing action of the parametron input transformer.

Another very unique advantage with the use of sinusoidal signals for driving and sensing is

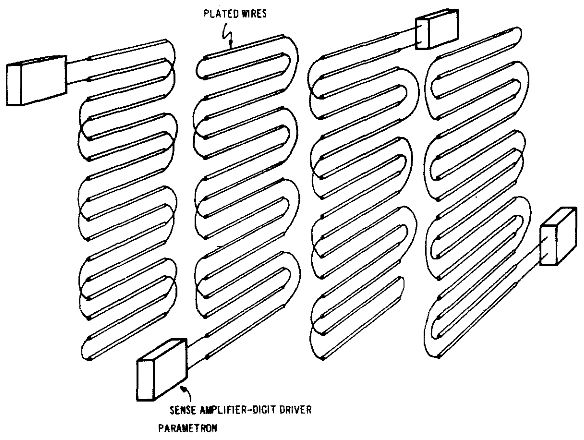


Figure 9. Non-Interacting Digit Planes.

the complete absence of pattern sensitivity for transformer coupling. A sine wave burst, the wave form of the sense signal and the digit current, may be passed through a transformer without level shift since there is no dc-base line component generated. This allows much greater freedom in the use of transformers for coupling in and out of the stack, greatly improving isolation and reducing ground noise.

MEMORY SYSTEM DESIGN

Figure 10 is a complete block diagram of the memory. In the interest of improved speed and since the actual word currents are pulses rather than sine waves, conventional row and column decoding to a matrix of saturated transistor switches is used. Diode logic is used to decode the addresses which are developed by parametron flip-flop registers driving phase-to-dc converters.

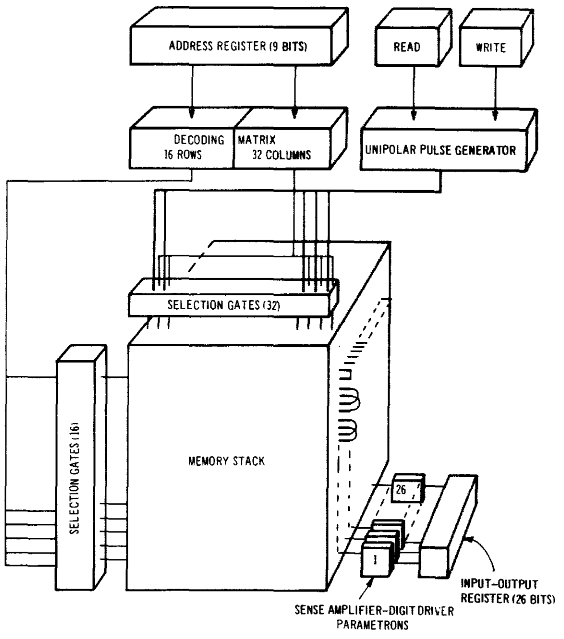


Figure 10. Block Diagram of 512 Word, 26 Bit/Word System.

Since only one word line at a time is energized, the read-write current source is simplified; the pulse current required is on the order of 100 ma at a 10 mc PRF. Figure 11 illustrates the system techniques of generating the read and write word trains with appropriate 25 nsec delay to allow optimum phase fixup for reading.

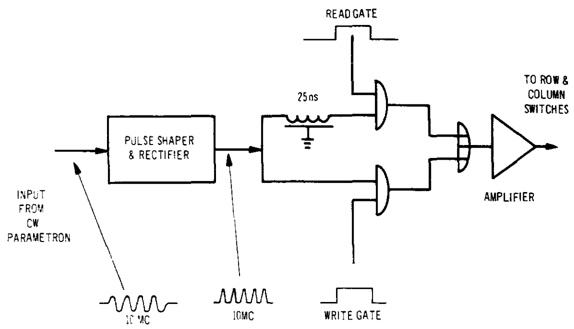


Figure 11. Word Current Shaper and Generator.

The digit parametrons are special, cylindrical thin-film parametrons, pumped at 20 mc (10 mc subharmonic) and operated in such a way as to produce approximately 100 ma pp of digit current in the line. The digit parametron also operates as a sense amplifier and is part of a three-parametron chain which forms the input-output register. Figure 12a is a schematic representation of the digit circuit.

For readout, it is necessary to insure that the sense output is the only locking signal present. This is conveniently done by cancelling the input signal from the Beat I parametron by appropriate logical gating, as shown in Figure 12b.

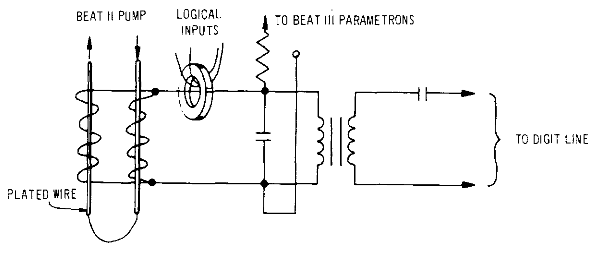


Figure 12a. Sense Amplifier-Digit Driver Parametron Schematic.

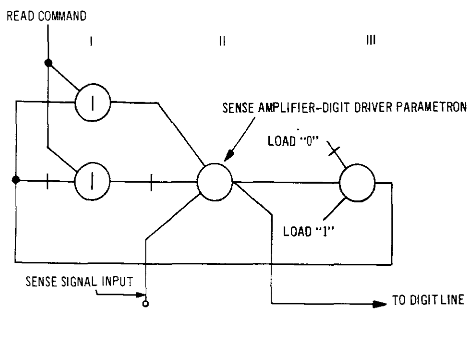


Figure 12b. Sense Amplifier-Digit Driver and Input-Output Register Logical Diagram.

If a "write only" command is to be executed, information is loaded into the register element at the appropriate time to cause the digit driver parametron (Beat II) to be at steady-state in the proper phase when the write word current is turned on.

Figure 13 is a detailed timing diagram for the memory.

The basic memory cycle is five microseconds to be compatible with the 200-kc data rate of the parametrons described in a companion paper.[5]

CONCLUSIONS

The principles of a new technique for using a thin magnetic film as a phase-script memory element have been presented. The use of cylindrically-oriented permalloy plated wire results in a simple, inexpensive memory element, with many design advantages.

Departures from previous phase-script memory techniques are the use of transverse switching with a uni-polar word pulse of the same PRF as the sinusoidal digit drive for fast writing. This eliminates the necessity for magnetic "creeping" and long write times previously encountered in other phase-script memory systems. The use of the uni-polar waveform for the word pulse eliminates the need for a word pulse PRF half that of the digit current, makes a simple row-column linear-select matrix possible, and allows generation of phase-script readout based on the fundamental harmonic component of the complex sense signal which may be used to lock a parametron operated as a sense amplifier. By a simple system artifice, an unfavorable phase relation between the readout sense signal and the required digit current may be corrected and an automatic read-restore cycle obtained by the use of a common parametron for the sense amplifier and digit driver.

The design of a 512 word (26 bits/word) memory has been described. The use of small multi-turn solenoids encapsulated in planes and assembled in arrays to form the memory stack allows reduction of word currents to moderate

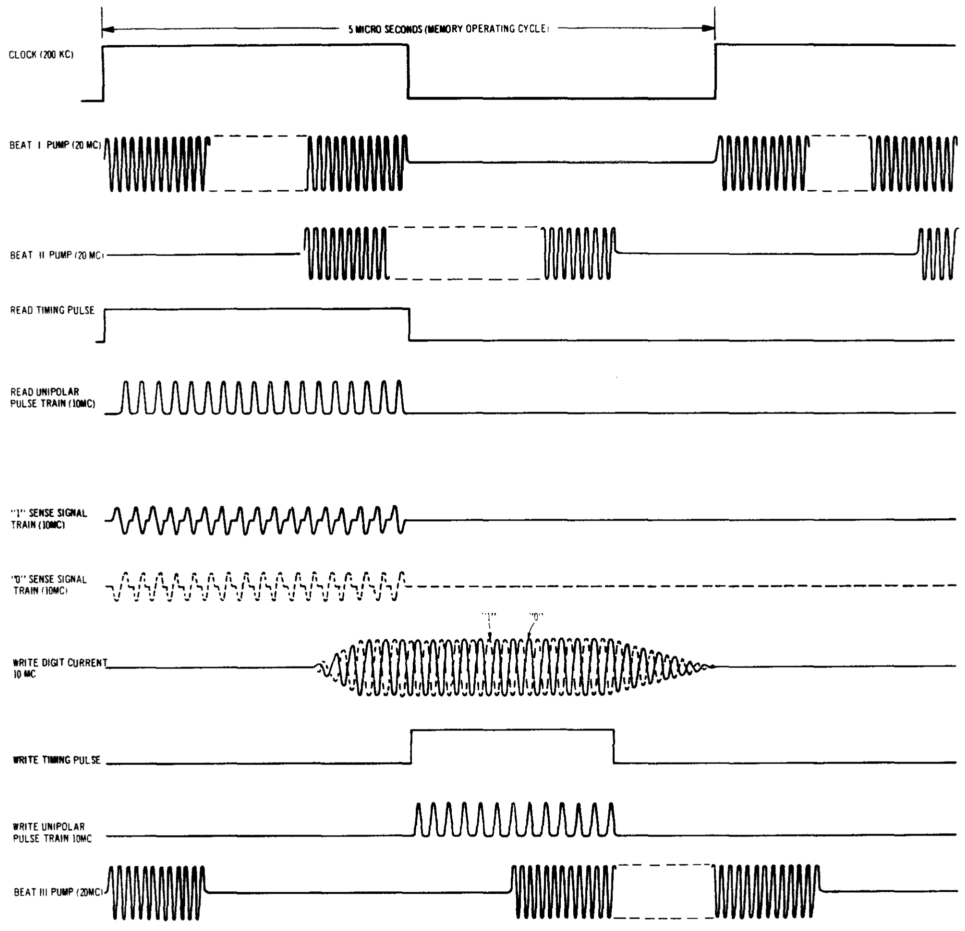

5 MICRO SECONDS (MEMORY OPERATING CYCLE)
CLOCK (200 KC)
BEAT I PUMP (20 MC)
BEAT II PUMP (20 MC)
READ TIMING PULSE
READ UNIPOLAR PULSE TRAIN (10MC)
"1" SENSE SIGNAL TRAIN (10MC)
"0" SENSE SIGNAL TRAIN (10MC)
"1"
"0"
WRITE DIGIT CURRENT 10 MC
WRITE TIMING PULSE
WRITE UNIPOLAR PULSE TRAIN 10MC
BEAT III PUMP (20MC)

Figure 13. Memory Cycle Timing.

levels allowing simple transistor circuitry to be used. Diode logic has been used for the word decoding to achieve higher speed. It is entirely possible that an all-parametron address decoding could be utilized at the expense of speed.

Considerable sophistication has been shown necessary in the design of the sense-digit line, but the single frequency operation of this line allows many simplifications over conventional pulse memories.

It is believed that concepts leading to a low cost, moderate speed, yet highly practicable thin-film memory, that is ideally suitable for parametron computing systems, have been presented.

ACKNOWLEDGEMENTS

The authors would like to thank Mr. A. Kolk and Mr. L. Douglas for preparing the plated wire to their specifications, Mr. C. Britton for his work on the packaging, and Mrs. J. Greenwell and Miss M. Roberts for their skillful assembly efforts.

REFERENCES

1. E. GOTO, Personal Communication.
2. W. E. PROEBSTER, "Parametron Thin Film Storage", *IBM Technical Disclosure Bulletin,* Vol. 2, No. 6, pp. 116-118 (April 1961).

3. S. YAMADA, T. BESSHO, T. KOSHIBA, "Magnetic Core Memory of the Parametron Digital Computer, Musasino—1", *Journal of Electrical Communication Engineers of Japan,* November 1958.
4. D. A. MEIER and A. J. KOLK, "The Magnetic Rod—A Cylindrical Thin Film Element", *Large Capacity Techniques for Computing Systems,* Edited by M. C. Yovits, Macmillan Co., pp. 195-212, 1962.
5. B. A. KAUFMAN, W. G. PFEIFFER, V. K. RANDERY, and A. J. KOLK, "Engineering Charateristics of Cylindrical Thin Film Parametrons for Use in Digital Systems", Proceedings of 1963 Fall Joint Computer Conference, Las Vegas, November 1963.

LAMINATED FERRITE MEMORY

R. Shahbender, C. Wentworth, K. Li, S. Hotchkiss, and J. A. Rajchman
RCA Laboratories
Princeton, N. J.

I. INTRODUCTION

This paper describes a random access magnetic memory consisting of a monolithic sheet of ferrite with embedded conductors made by a simple batch fabrication technique: lamination of ferrites. Its tightly packed elements with closed flux paths of only two to three mils in equivalent diameters are the smallest yet realized by any technique. This smallness is being exploited for high speed. Cycle times as short as 100 nanoseconds have been demonstrated. The smallness of the elements leading to modest drive requirements of only a few tens of milliamperes combined with the inherently low cost "integrated" batch fabrication technique opens the possibility of low cost memories of very large capacities: tens or hundreds of millions of bits. A significant cost reduction for the whole memory system is possible because this type of "integrated" magnetic structure lends itself particularly well to be integrated with integrated semiconductor driving and sensing circuits.

A few general remarks may help to set the subject in perspective within the fast advancing art of computer memories. The usefulness of a digital computer depends to a large extent on the speed and storage capacity of its random access memory. In general, the maximum attainable capacity decreases with increasing speeds.[1] Capacities in the range of a few million bits at a cycle time of a few microseconds have been obtained with ferrite cores.

Recently, the feasibility of a 100 ns cycle time, i.e., 10 Mcps repetition rate, ferrite system was established[2] and microferrite memories with a cycle time of 375 ns are commercially available.[3] These results have extended the usefulness of the well established ferrite technology to high speeds which were the original goals motivating much of the efforts with thin magnetic metallic films. The result in the case of ferrite is obtained by miniaturization of the element and the density of appropriate "printed" microscopic windings. Impulse or partial switching is used in addition to miniaturization to further reduce the magnetic size of the storage element below its physical size. In a sense, the laminated technology presented here is a culmination of our recent efforts[2, 3] to reduce the element size by providing a single means of making small elements and all necessary windings.

Elements with closed magnetic paths can provide a relatively higher ratio of sense voltage to drive current than open flux path elements. Furthermore, there are no limitations to geometrical shapes due to demagnetizing effects, and there are no fringing fields limiting packing density. To obtain some of these benefits with thin film technology, paired patches are resorted to in many of the more recent approaches.[4]

The successful development of integrated magnetic structures with elements having closed magnetic flux paths includes the aperture ferrite plate,[5] the FLEA permalloy sheet,[6] and the waffle iron memory.[7] It is believed that the laminated ferrite memory is a further step in this integration providing smaller elements at much lower cost.

The paper describes the fabrication methods, the results obtained with the magnetic structures and some speculations on the possibility of integrated semiconductor drives.

II. ORTHOGONAL ARRAY STRUCTURE

The laminate technology permits a great flexibility of geometries of the windings. The simple orthogonal winding structure shown in Fig. 1 is particularly suited for high speed and is described in detail in this paper. In this structure two orthogonal sets of conductors, an X-directed set and a Y-directed set, are embedded in a sheet of square loop ferrite. The conductors are electrically insulated from one another by ferrite. Each cross-over point, between an X and a Y conductor, is a storage location. For high speed the array is operated using impulses switching in a word organized, two cross-overs per bit mode, analogous to the word organized two cores per bit[5] mode of conventional arrays. The word current (read-write) is applied to an X-winding, and the Y-windings are used for both digit and sense.

The experimental arrays fabricated to date have a capacity of 256 crossovers (128 bits equivalent to 16 words of 8 bits each). The overall thickness of the ferrite laminate is approximately 5 mils and the conductor spacing, for both the X and Y conductors, is 10 mils. The conductor cross-sectional dimensions are approximately 2.5 $\times$ 0.7 mils.

III. MEMORY FABRICATION TECHNOLOGY

The basic operations involved in the fabrication of a laminated memory array are: "Doctor Blading," "Laminating and Sintering," and "Conductor Screening."

1. *Doctor Bladed Ferrites*

The "doctor blading" technique (DB for short) for fabricating sheets of ferrites consists of preparing a slurry of the desired ferrite powder and appropriate organic binders. The slurry is spread in an even layer on a glass

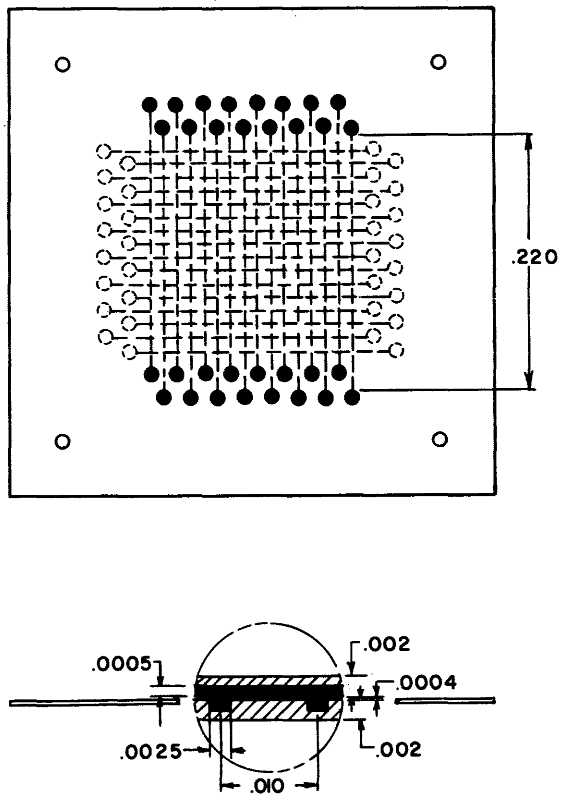


Figure 1. Laminated Array Structure.

Figure 2. Doctor Blading Ferrite.

substrate by the sweeping action of a blade, called a "doctor blade," held at a constant distance above the glass surface as shown in Fig. 2. The sheet is air dried and peeled off the glass surface.

The 60-cycle hysteresis loops for toroids of the same composition fabricated conventionally (dry pressing) and by DB, and fired simultaneously are shown in Fig. 3. The saturation flux for both types of cores is essentially the same. The coercive force for the DB cores is somewhat higher than that for the dry-pressed cores, when both are fired under the same conditions. However, by modifying the firing schedule for the DB cores, the resulting characteristics are the same as those of dry-pressed cores. The pulse behavior of DB cores is again essentially the same as that of dry-pressed cores (equal switching coefficients).

2. *Laminated Ferrites*

Monolithic structures with embedded conductors are fabricated by laminating together

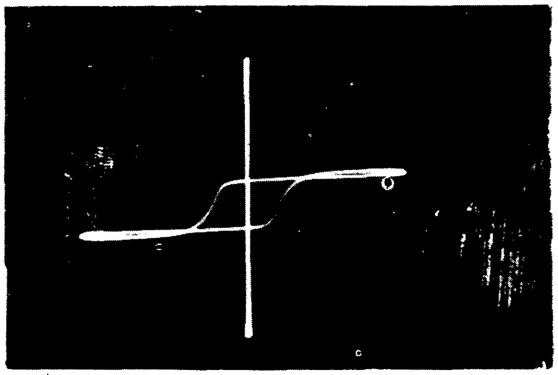


Figure 3A. Hysteresis Loop of Pressed Ferrite Core.

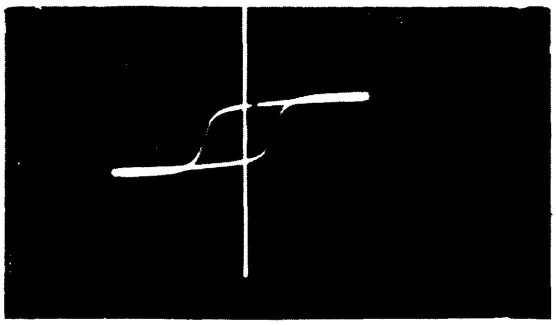


Figure 3B. Hysteresis Loop of Doctor-Bladed Ferrite Core.

the required number of sheets. This is accomplished by pressing green sheets together at moderate pressures and temperatures which are not too critical. A simple hydraulic press with heated platens is adequate for this purpose. The ferrite sheets are sandwiched between two aluminum blocks as shown schematically in Fig. 4, and placed between the

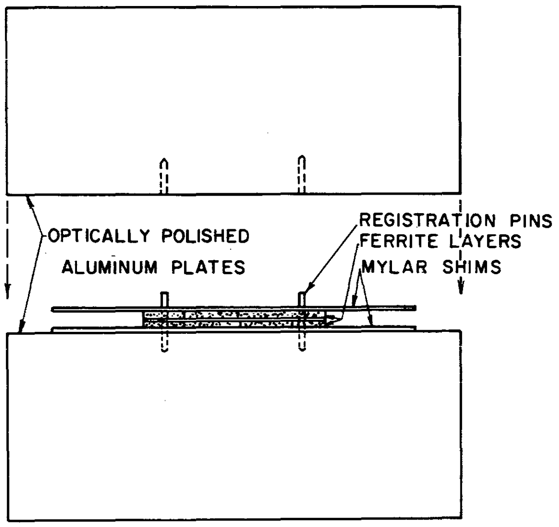


Figure 4. Laminating Jig.

heated platens of the press. Pressure is applied for a few minutes after the block temperature, as monitored by a thermocouple, has stabilized. The laminated sheets are next sintered in a controlled temperature furnace.

Multiple lamination was tested with many layers. For example, a sample in which a total of 60 sheets, each of 2.5 mils thickness, was successfully laminated and fired. Fig. 5 shows a micrograph of a section through the sample which contains an 8 $\times$ 8 matrix of embedded conductors whose ends can clearly be seen. Inspection of the micrograph shows a uniform ceramic body with no delamination marks. Magnetic testing of this and similar samples in which the flux is switched across the laminating plane leads to the same results as when the flux is switched in the plane of the original doctor bladed ferrite sheets. In other words, a truly uniform isotropic material is obtained.

Experiments with dimensional stability were made using sheets 5″ $\times$ 10″ and thicknesses

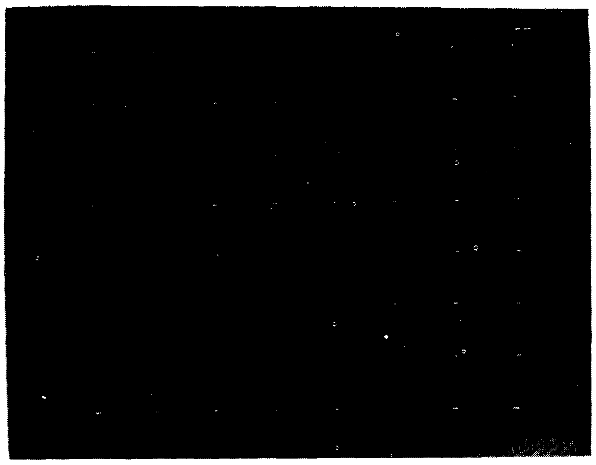
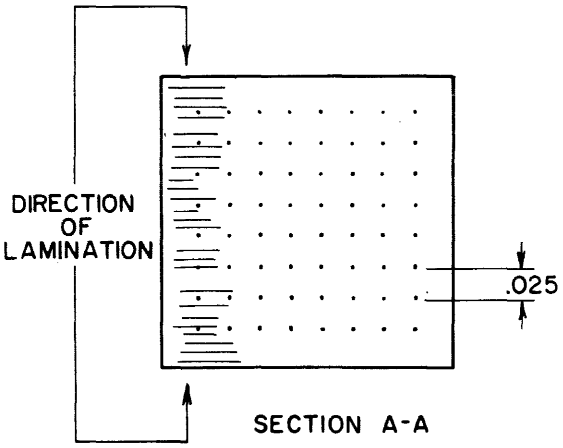


Figure 5A. Thick Laminate.



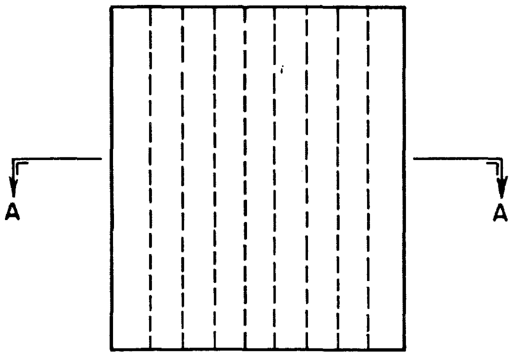


Figure 5B. Thick Laminate.

varying from 0.1 to 10 mils. It was found that properly sintered samples cut from these sheets could be relied on to have all their dimensions within ±5 per cent. Better tolerances were obtained under special conditions. For example, the sample of Fig. 5 had spacing between conductors as shown in Fig. 6. It is

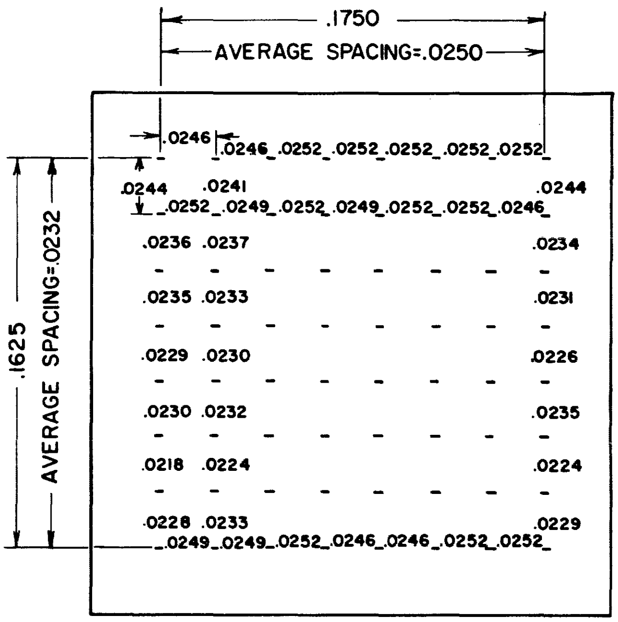


Figure 6. Dimensions of Thick Laminate.

seen that variations in spacings are only ±0.5 mils for a nominal spacing of 0.025″.

3. *Conductor Screening*

The technique developed for forming conductors as an integral part of a "green" ferrite sheet is similar to the familiar "silk screening" process. A photo-formed metal mask is laid on a glass substrate and a paste consisting of the required metal powder and a binder is squeegeed through the mask onto the glass substrate. The mask is then removed leaving the required conductor pattern on the glass surface. Ferrite is doctor bladed on the glass substrate over the conductor pattern. When peeled off the glass the ferrite sheet contains the conductors intimately embedded in it and flush with its surface. Similar ferrite sheets with conductors, or spacer sheets without conductors, are then laminated, as described above.

On firing, the conductors sinter along with the ferrite.

Conductor dimensions as small as 1.3 × 0.5 mils in cross section shown in Fig. 7, with a resistance of 2 ohms per inch, have been obtained. For most of the arrays tested, the conductor cross section is 2.5 × 0.7 mils, shown in Fig. 8, and their resistance is 3 ohms per inch.

IV. ORTHOGONAL ARRAY FABRICATION

A number of experimental arrays 16 × 16 were made. The orthogonal array structure, as shown in Fig. 1, is fabricated by laminating three sheets of doctor bladed ferrite in the order shown in Fig. 9. The top and bottom sheets, with a "green" thickness of approximately 2.5 mils, contain conductors spaced 13 mils apart.

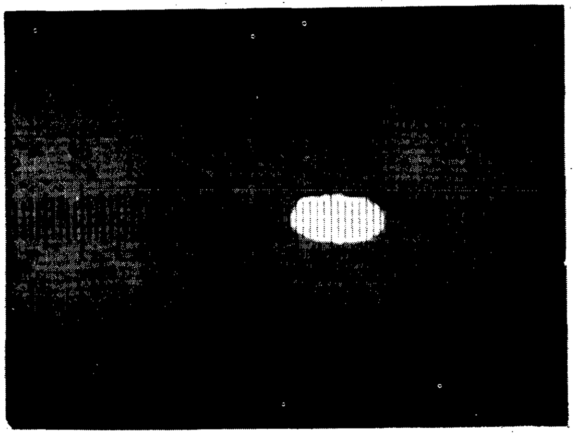


Figure 7. 1.3 × 0.7 Mil Conductor Section.

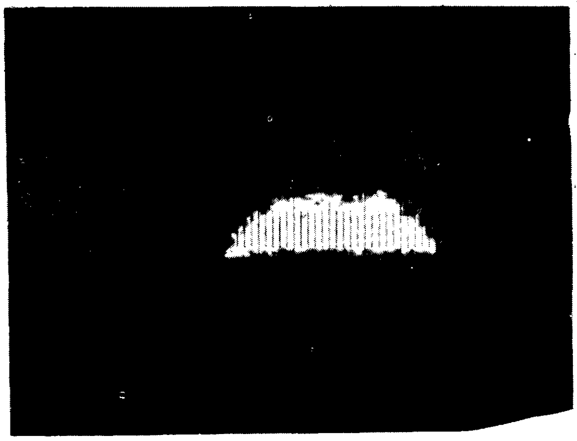


Figure 8. 3.0 × 0.8 Mil Conductor Section.

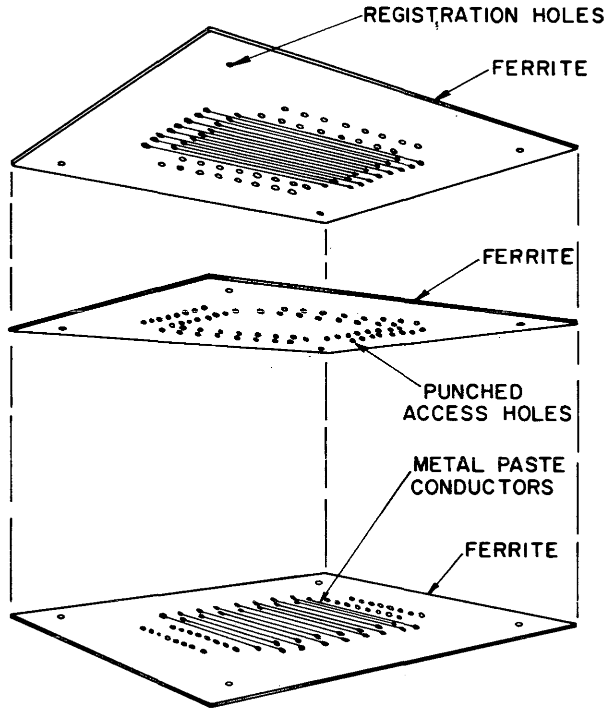


Figure 9. Laminate Details.

The center sheet is 0.5 mils thick and contains no conductors. Sixteen mil diameter holes are gang punched in each sheet in the patterns shown in Fig. 9. The rows of holes are used as access to the embedded conductors. The corner holes are used for registry during assembly and match the pin locations in the laminating jig of Fig. 4. After sintering, the ferrite laminate shrinks to the dimensions shown in Fig. 1 (overall thickness approximately 5 mils, conductor spacing 10 mils).

Interconnections to the embedded conductors are provided via the plastic film with etched conductors shown in Fig. 10. The conductors, spaced on 10 mil centers, overhang the plastic film and are positioned over the access holes. They are then manually soldered in place. Fig. 11 shows a 16 × 16 array mounted on a printed circuit board ready for testing. Fig. 12 is a magnified X-ray photograph of an array. Fig. 13 is a partial cross section showing some of the embedded conductors which shows the

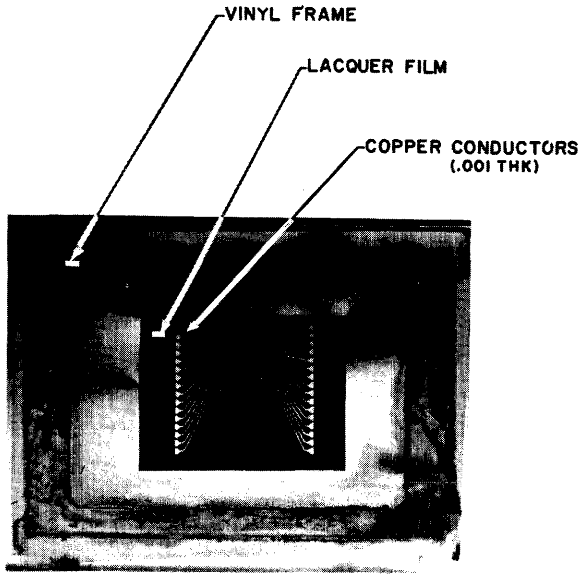


Figure 10. Interconnecting Conductors.

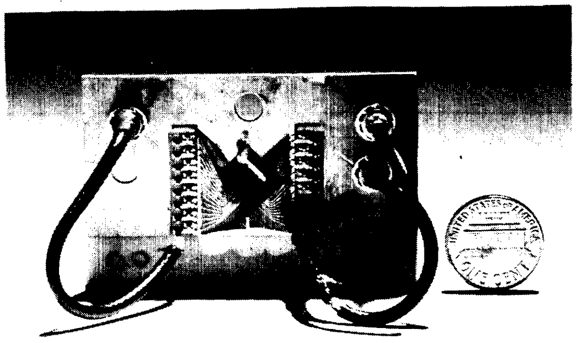


Figure 11. 16 × 16 Array Mounted for Test.

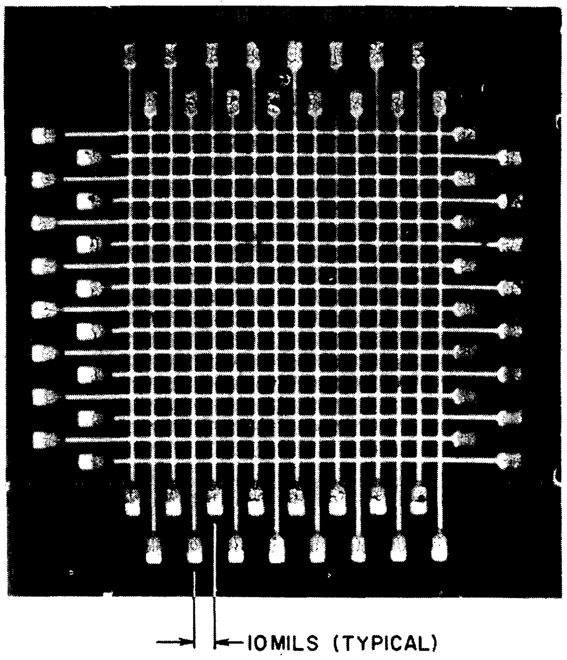


Figure 12. Magnified X-Ray Photo of Array.

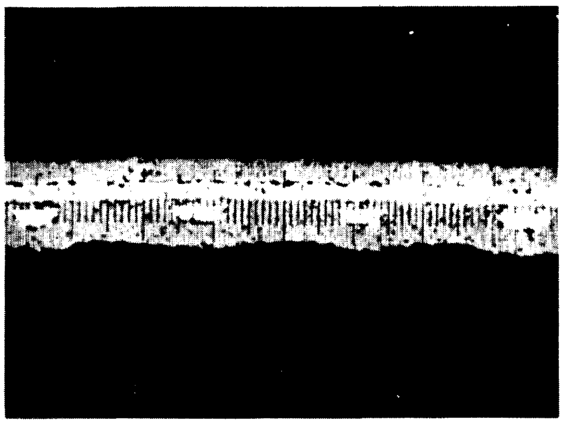


Figure 13. Micrograph of Partial Cross Section of Array.

monolithic nature of the ferrite without any trace of the lamination.

V. OPERATING MODE

The memory is word organized and two crossovers per bit are used. In this mode all bits of a selected word are subjected to the same read-write current pulses. The best digit drive techniques[2] found consist of applying a unipolar digit pulse to either one or the other of the two crossovers of a bit in time coincidence with the write pulse. In other words, if the two crossovers of a bit are labeled A and B, crossover A is digited with say a positive pulse to write a binary "1" and crossover B is digited with a positive pulse, also, to write a binary "0."

Sensing is differential in that the output sense voltage obtained during the read is the difference between the two voltages induced along the digit sense windings linking the two crossovers of a bit. Different polarities of voltage correspond to the two binary states.

Because of the orthogonal disposition of word and digit conductors, word read-write currents switch flux along a selected word conductor that does not link the digit conductor. A digit pulse applied to a digit conductor in time coincidence with a write pulse switches a component of flux mutual to both the word and digit conductors at the corresponding crossover point. The application of a read pulse switches this mutual flux and induces a sense voltage in the digit winding. The polarity of the induced sense voltage is determined by the polarity of the applied digit current.

Operation of a bit may be visualized with the aid of the vector diagrams shown in Fig. 14. The flux switched by the word write current in the vicinity of a crossover point may be represented by the vector ϕ_w. The magnitude of this vector is proportional to the flux contributing to the bit operation. The direction of this vector is normal to the plane of the flux itself, and using the right hand rule convention, is in the direction of the current establishing the flux. The application of a digit current in time coincidence with the write pulse causes the vector ϕ_w to tilt to ϕ_t in the direction of the digit current, i.e., to have a component in the digit direction, as shown in Fig. 14.

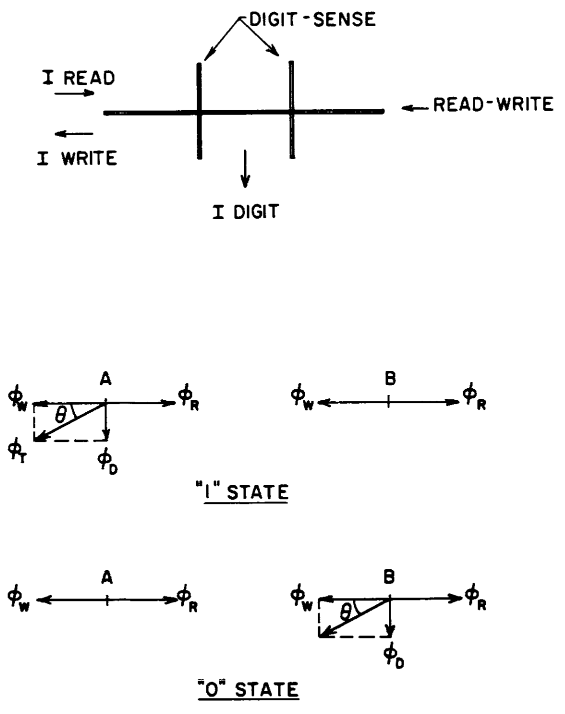


Figure 14. Flux Vector Diagram.

Application of the word read current reverses the flux vector ϕ_t to the position ϕ_r resulting in a sense output. Increasing the amplitude of the digit current relative to the write current increases the tilt angle and correspondingly the sense output. The amplitude of the digit current is limited by its disturbing effect on the stored information.

VI. SYSTEM ANALYSIS

The high speed performance of the orthogonal array structure may be predicted from the known ferrite characteristics and some simplifying assumptions. Starting with the switching equation[8]:

$$T_s\ (H - H_c) = S_w$$

where T_s = switching time
H = applied magnetic reversing field
H_c = coercive field of ferrite used
S_w = switching coefficient of ferrite used

An estimate of the required word writing current may be made. For optimum operation the write current magnitude is adjusted to completely switch along the perimeter of the word conductor and partially switch beyond. From the switching equation the optimum write current I_w is:

$$I_w = 2\pi R_o \left[\frac{S_w}{T_s} + H_c \right]$$

where $2\pi R_o$ = cross-sectional perimeter of the word conductor.

The amount of inelastic flux partially switched by the word write current per crossover (flux vector ϕ_w in Fig. 14) may be estimated by assuming an inverse relationship between flux switched and switching time. In other words, if B_r is the remanent saturation flux of the ferrite, then the flux B partially switched by I_w at a radius r is:

$$B = K \frac{B_r}{r}$$

The proportionality constant K is determined from the condition that at

$$r = R_o, \qquad B = B_r.$$

The total inelastic flux switched by I_w per crossover is:

$$\phi_w = Kl \int_{R_0}^{R} \frac{B_r}{r} dr = KlB_r \, Ln \frac{R}{R_0}$$

where R = outside radius to which flux is switched
l = spacing between conductors

The back voltage V_w induced on the word drive line per crossover due to the write current switching the inelastic flux from $-\phi_w$ to $+\phi_w$, and assuming a peaking factor of 2, is:

$$V_w = \frac{2(2\phi_w)}{T_s}$$

For the sample assumptions, the back voltage V_{we} due to the elastic flux switched by the word write current is:

$$V_{we} = \frac{4\phi_w}{T_s} \frac{1-S}{S}$$

where S = squareness ratio of the ferrite defined as the ratio of the remanance flux to the saturation flux

The digit current causes the flux vector ϕ_w (Fig. 14) to tip through an angle θ, which for small θ, is:

$$\theta \simeq \frac{I_d}{I_w}$$

The flux contributing to the sense output is:

$$\phi_s = \phi_w \tan \theta \simeq \phi_w \frac{I_d}{I_w}$$

The expected sense output voltage V_s due to the switching of the flux ϕ_s by a read current I_r from $+\phi_s$ to a neutral state (assuming the same peaking factor of 2) is:

$$V_s = \frac{2\phi_s}{T_s} \frac{I_r}{I_w} = \frac{2\phi_w}{T_s} \frac{I_d I_r}{I_w^2}$$

The value of digit current I_{dm} at which disturb effects are initiated is determined by the coercive force of the material. For this digit current to disturb stored information it must switch flux linking the word conductor. The minimum path length P along which this occurs is defined by the conductor dimensions and their separation:

$$P = 2w + 4t + 2d$$

where w = conductor width
t = conductor thickness
d = separation between conductors

The disturbing digit current is given by:

$$I_{dm} \propto H_c P$$

Digit current values greater than I_{dm} may be used if a slight disturb in sense outputs can be tolerated. This is especially true in a system where the sense outputs are bipolar.

The back voltage induced on the word line, per crossover, due to the read current I_r switching both inelastic and elastic flux is:

$$V_r = (V_w + V_{we}) \frac{I_r}{I_w} = \frac{4\phi_w}{T_s} \cdot \frac{I_r}{I_w} \cdot \frac{1}{S}$$

To the above must be added the resistive drop in the conductor to compute the required driving voltage.

For the ferrite selected for the laminated memory,[2] and for 100 nanoseconds cycle time, the following values may be used in the preceding equations:

$H_c = 1$ oersted
$S_w = 0.3 \times 10^{-6}$ oersteds-seconds
$B_r = 10^3$ Gauss
$T_s = 30 \times 10^{-9}$ seconds
$S = 0.7$
$w = 2.5$ mils
$t = 0.5$ mils
$R_0 = \frac{6}{2\pi}$ mils
$R = 1.75$ mils
$d = 0.4$ mils
$l = 10$ mils

Substituting these values in the above equations leads to

$I_w = 132$ ma
$\phi_w = 3.6 \times 10^{-2}$ maxwells
$V_w = 4.8 \times 10^{-2}$ Volts
$I_{dm} = 16$ ma

Assuming a read current of 300 ma, and a digit current of 20 ma, the sense output and back voltage per crossover are

$V_s = 8.3 \times 10^{-3}$ Volts
$V_r = 1.56 \times 10^{-1}$ Volts

The characteristics of the digit-sense winding may be estimated by computing L, the distributed inductance per crossover, and C, the distributed capacitance per crossover, in addition to the crossover capacitance. These quantities are:

$$L \propto \mu l \ln \frac{R}{R_0}$$

$$C \propto \frac{\epsilon A}{d} + \frac{\epsilon W l}{R}$$

where

μ = small signal relative permeability of ferrite

ϵ = relative dielectric constant of ferrite

A = w^2 = crossover area between a word and a digit conductor

The characteristic impedance Z_0, and delay per bit τ_d for the sense digit line are approximately given by:

$$Z_0 = \sqrt{\frac{L}{C}} \text{ ohms}$$

$$\tau_d = \sqrt{LC} \text{ seconds}$$

For the ferrites used in the laminated memory and at frequencies compatible with 10 Mcps operation, the following values may be used:

$\mu = 100$

$\epsilon = 10$

leading to:

$L = 3 \times 10^{-9}$ Henry

$C = 6.6 \times 10^{-14}$ Farad

$Z_0 = 214$ ohms

$\tau_d = 1.4 \times 10^{-11}$ seconds/bit

VII. EXPERIMENTAL DATA

A. *Ferrite Composition*

The composition selected for the high speed laminated array is the same as that in use for the previously developed high speed ferrite memories.[2, 3] The switching coefficient for this material is 0.3 oersteds-microseconds, the coercive force is 1.0 oersted, the Curie temperature is 100° C, and the squareness ratio is 0.7. The magnetic characteristics achieved with this material using doctor blading and laminating techniques are identical to those of conventionally processed toroids.

With different materials other operational characteristics could be obtained. For example power requirements can be very low using a switchable ferrite with low coercity, e.g., half an oersted. Also, recently developed wide temperature range material could be used.[9]

B. *Array Operation*

Three basic pulse programs are used to evaluate the performance of an array. These are:

1. Alternate "One-Zero" read-write.
2. Alternate "One-Zero" read-write with disturbs.
3. Mixed "One's and Zero's" read-write with disturbs.

These programs are provided by a set of transistor drives (described in detail in reference 2) triggered by commercially available 10 Mcps logic building blocks. The read-write and digit pulses are of adjustable amplitude, 30 nanoseconds wide at the base with a maximum repetition rate of 10 Mcps.

Fig. 15 shows the circuit layout used to test the array operation. Individual words are connected to the read-write drivers, and digiting is accomplished via the resistors connected to the sense-digit lines. A sense output transformer with a 1:1 turns ratio (10 turns on each side) wound on a ferrite bead is used to obtain the difference signals. The reactance of the transformer is considerably higher than the impedance to ground presented by the sense-digit lines for 16 × 16 arrays. Thus the shunting effect produced by the transformer

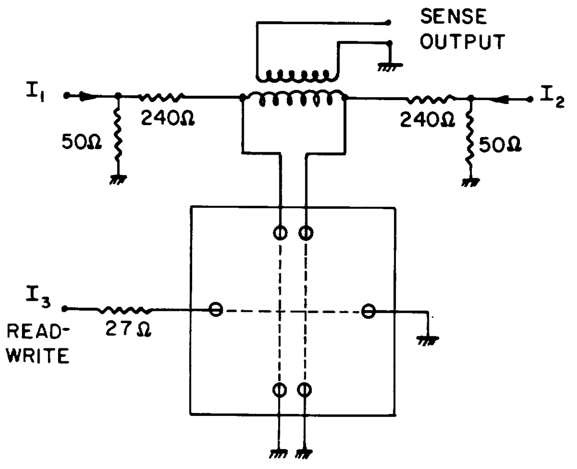


Figure 15. Memory Array Test Schematic.

in series with the undigited side on an applied digit pulse is negligible.

1. *Alternate "One-Zero" Read-Write*

In this program an alternating pattern of "One and Zero" is cyclicly written into, and read out of the memory bit at a cycle time of 100 nanoseconds and a repetition rate up to 10 Mcps. Fig. 16 shows the pulse program used

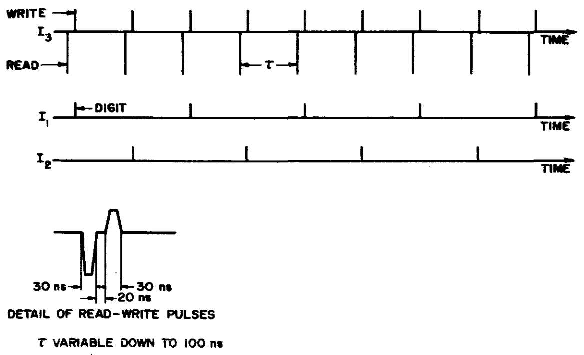


Figure 16. Alternate One and Zero Pulse Program.

and Fig. 17 shows oscillograms of the voltage developed at the sense output terminals under constant drive conditions and at three different repetition rates: 0.5, 2.0, and 7.2 Mcps. The "one" and "zero" outputs are shown superimposed at the lower repetition rates. Inspection of Fig. 17 shows that the sense voltage is independent of repetition rate. The operating current levels and sense outputs are:

Read Current:	370 ma
Write Current:	220 ma
Digit Current:	21 ma
Undisturbed Sense Output:	±9 mv

Back Voltage (including IR drop) per crossover = 340 mv

2. *Alternate "One-Zero" Read-Write with Disturbs*

In this program an alternating pattern of "One and Zero" is cyclicly written into the memory bit. Each write operation prior to reading is followed by the application of disturb pulses aimed at destroying the stored information. Fig. 18 shows the pulse program and Fig. 19 shows oscillograms of the voltage

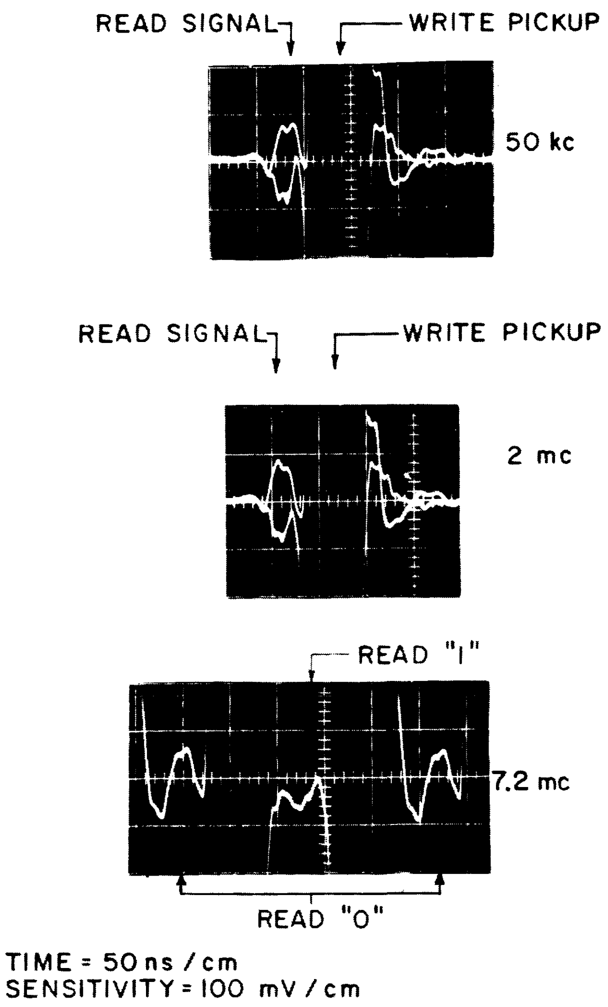


Figure 17. Sense Output Voltages at 0.05, 2, and 7.2 Mcps (Pulse Program of Fig. 16).

developed at the sense output terminals for the same current drive levels as above.

The disturbed sense voltages are ±6 mv; a decrease of 3 mv due to the digit disturbs. Experimentally it was determined that 3 × 10^5 disturb pulses produced no additional disturb effects as compared to only 100 pulses.

3. *Mixed "One-Zero" Read-Write with Disturbs*

The pulse program described under Item 2 above is not the most pessimistic. A further deterioration of the sense signal occurs if the following pulse program is applied: a large number of "one's," each followed by disturbs in the "one" direction, is written and read from

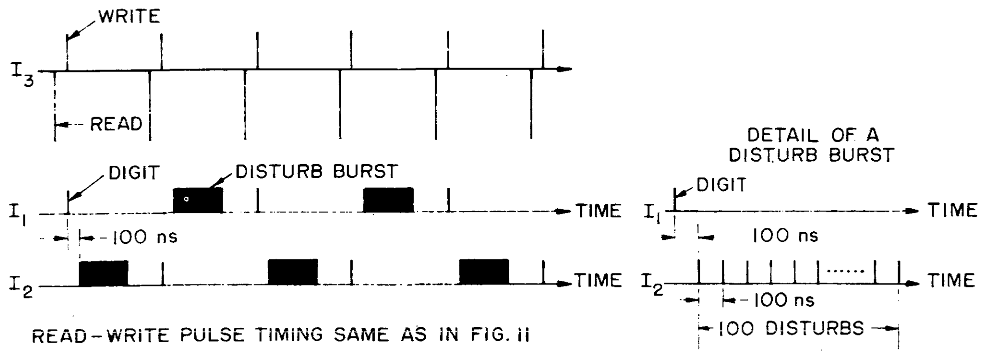


Figure 18. Alternate One and Zero with Disturbs Pulse Program.

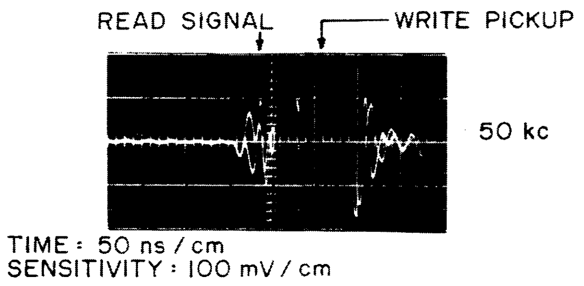


Figure 19. Sense Output Voltages with 10 Mcps Disturbs (Pulse Program of Fig. 18).

the bit; this is followed by a single zero, which is then disturbed in the "one" direction. This zero is then read out and the output voltage recorded. The mirror image of this program is then applied, and a "one" read out and recorded. The program is shown in Fig. 20, and Fig. 21 shows oscillograms of the voltage developed at the sense output terminals for the same current levels as above. The sense voltages obtained are ±5 mv. Comparing Figs. 17, 19, and 21, a progressive decrease in output voltage is noted. It is felt that the pulse program of Fig. 21 represents a sufficiently stringent test so that a memory array that yields the acceptable output of Fig. 21 under the conditions of the test will perform satisfactorily in a computer under random conditions.

C. *Uniformity of Fabricated Arrays*

The uniformity of the electrical and magnetic characteristics of laminated arrays is obviously of importance in determining the feasibility of the fabrication process. The measured electrical resistance of 32 (2 × 16) embedded conductors in a recently fabricated sample have a total variation of 0.2 ohms (15 conductors have a resistance of 0.7 ohms, 11 conductors have a resistance of 0.8 ohms, and the remaining 6 conductors have a resistance of 0.9 ohms). Fig. 22 shows the superimposed sense output signals obtained from the same bit position in eight successive words. Approxi-

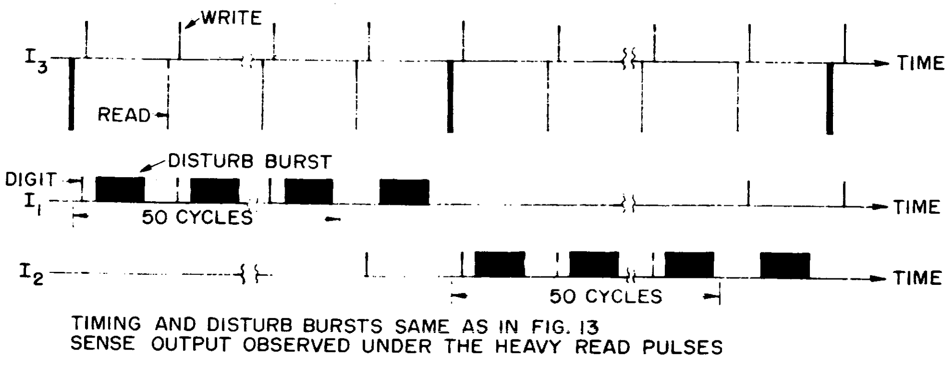

WRITE
I_3
READ
TIME
DISTURB BURST
DIGIT
I_1
50 CYCLES
TIME
I_2
TIME
50 CYCLES
TIMING AND DISTURB BURSTS SAME AS IN FIG. 13
SENSE OUTPUT OBSERVED UNDER THE HEAVY READ PULSES

Figure 20. Mixed One and Zero with Disturbs Pulse Program.

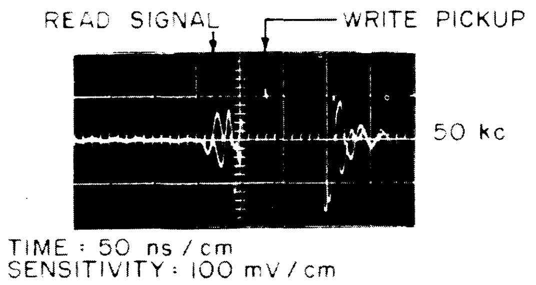


Figure 21. Sense Output Voltages with 10 Mcps Disturbs (Pulse Program of Fig. 20).

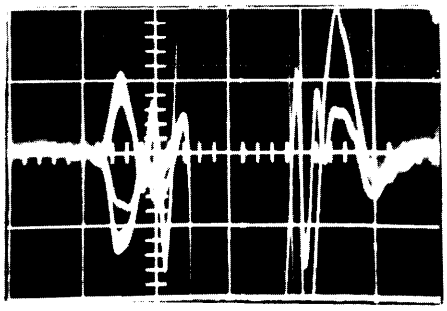


Figure 22. Superimposed Sense Outputs from Corresponding Bit of Eight Words.

mately 80 per cent of the 128 bits in the array have acceptable outputs above a certain arbitrary minimum while the remaining bits could possibly be used, but would present undue system difficulties. The non-uniformity in output is attributable entirely to a non-uniform conductor cross section, and reasonable care in the fabrication of the conductors should provide the required uniformity.

D. *Propagation Characteristics of Sense-Digit Line*

The experimentally determined characteristics of the sense-digit lines in a laminated array are summarized in Table I. This data is obtained by propagating a pulse along a line consisting of the series connection of 16 sense-digit conductors in an array. The total embedded conductor length is 3.32 inches. This is equivalent to 332 bits spaced 10 mils apart. The 16 word conductors in the array are grounded as shown schematically in Fig. 23. The characteristic impedance of the line is experimentally determined from the condition

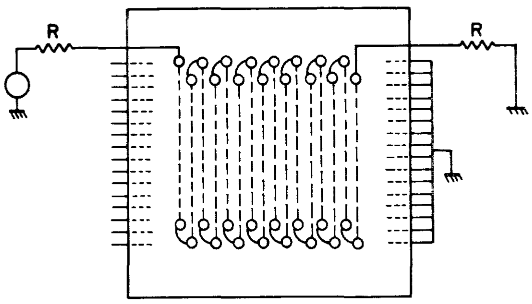


Figure 23. Long Bit Sense Digit Line.

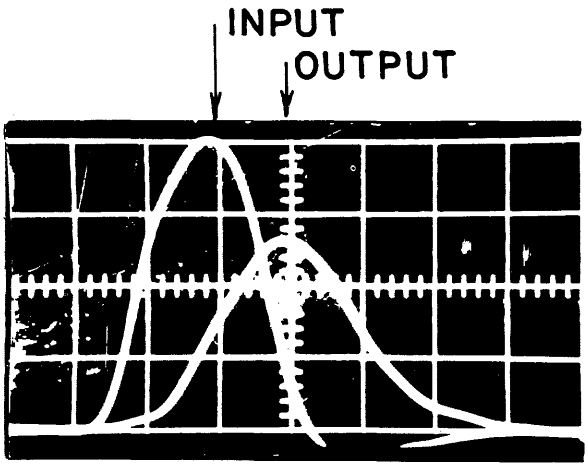


Figure 24. Pulse Propagation Characteristics.

of minimum reflection. The delay per bit is computed from the total delay between the peak of the output pulse and that of the input pulse, shown in Fig. 24, divided by 332, the number of equivalent bits on the line. The pulse attenuation is due to two factors: the attenuation αdc due to the series dc resistance of the line (measured value of 10 ohms), and αac due to the losses in the ferrite reflected as an equivalent ac line resistance. As can be seen in Table I, the ac losses exceed the dc losses.

TABLE I

Z_o	=	200 ohms
αac	=	5.7×10^{-3} db/bit
αdc	=	1.2×10^{-3} db/bit
τ_d	=	3.3×10^{-11} sec, bit

VIII. FUTURE DEVELOPMENTS

The fabrication techniques described are obviously applicable to the fabrication of a large

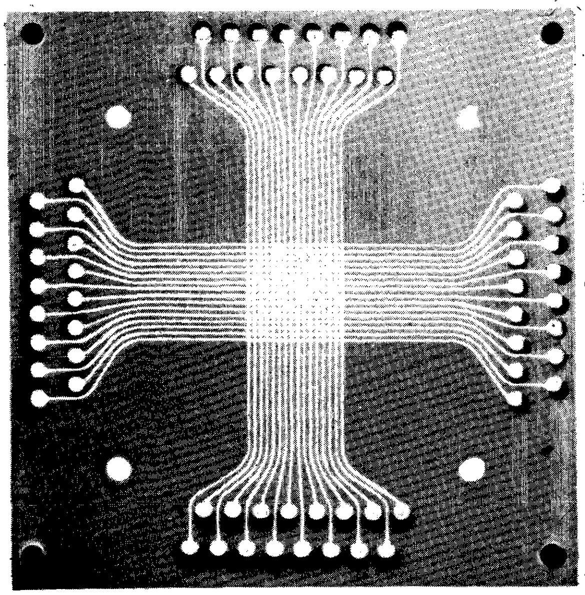


Figure 25. 5 Mil Spacing X-Ray.

class of ferrite devices and systems with integrated windings. The memory arrays fabricated and tested have not been especially optimized. For example, higher bit packing densities can be achieved quite readily. Fig. 25 is an X-ray photograph of an array with 5 mil conductor spacing. The advantages to be gained from this tighter packing is a reduction in back voltages, in propagation delays, and in signal attenuation. A further decrease in bit spacing, down to a few mils, is possible with the small cross sectional conductor dimensions shown in Fig. 7. The reduction in conductor cross section will result in a reduction in drive current requirements without a deterioration in sense output. For memories of short word lengths, say 20 to 30 bits, a thicker ferrite laminate may be used to advantage to give higher sense outputs with correspondingly higher back voltages.

Conversely, long words are readily realized by reducing the laminate thickness. This reduces both the sense output and signal attenuation.

For relatively slow speed operation, say a one microsecond cycle time, the required drive currents are under 50 ma as has been experimentally verified. The sense output is approximately one millivolt. The low drive currents and the relatively high sense outputs are well matched to the capabilities of integrated electronic circuitry. Word drivers, triggered by address decoding trees, can be fabricated and assembled as integrated arrays. Similarly, integrated sense amplifiers may be built to detect the binary signals . The physical structure and physical dimensions of a laminated arrays are also well suited for interconnection to arrays of integrated semiconducting elements. Combining laminated arrays with integrated electronic circuitry promises to result in a substantial cost per bit reduction over the present state of the art. This in itself will make economical random access ferrite memories with capacities in excess of 10^7 bits. The expected cycle time of these memories will be predominantly determined by the characteristics of the integrated semiconducting circuits and may conceivably be less than 0.5 microseconds.

IX. CONCLUSIONS

The results presented in this paper show conclusively that the techniques described are well suited for the low cost fabrication of memory arrays of micro size at exceptionally high packing densities. The physical and electrical characteristics of these arrays may be tailored to meet all important aspects of digital random access memories, including high speed (100 nanosecond cycle), large capacities (in excess of 10^7 bits), nondestructive read-out, wide temperature range operation, etc. The drive requirements and sense outputs for these arrays as well as their ease of fabrication place them in a favorable position with respect to other integrated magnetic arrays, e.g., FLEA memories, twistor memories, thin magnetic film memories, or the waffle iron memory.

X. ACKNOWLEDGMENTS

The authors wish to express their appreciation to their many colleagues at RCA Laboratories and the Product Division of the Corporation who were involved in this program.

Dr. Bernard Schwartz, RCA Laboratories, suggested the basic ceramic techniques used for the fabrication of these memory arrays. The authors wish to express their appreciation to him for his help during the early phases of the work.

The authors wish to express their thanks also to Messrs. R. Noack and A. Monsen, RCA Laboratories, who fabricated and wired many of the sample arrays.

XI. REFERENCES

1. J. A. RAJCHMAN "Computer Memories—Possible Future Developments," *RCA Review*, 23, No. 2, pp. 137, June, 1962
2. R. SHAHBENDER, T. NELSON, R. LOCHINGER and J. WALENTINE, "Micro-aperture High-Speed Ferrite Memory," *RCA Review*, 23, No. 4, pp. 539, December, 1962
3. H. AMEMIYA, H. P. LEMAIRE, R. L. PRYOR, T. R. MAYHEW, "High Speed Ferrite Memories," *Fall Joint Computer Conference*, pp. 184, 1962
4. A. V. POHM, R. J. ZINGG, G. A. WATSON, T. A. SMAG, R. M. STEWART, JR., "Large High Speed, DRO Film Memories," *Proceedings of the INTERMAG Conference*, pp. 9-5-1, April 1963
5. J. A. RAJCHMAN, "Ferrite Apertured Plate for Random Access Memory," *Proceedings of the IRE*, Vol. 45, pp. 325-334, March 1957
6. G. R. BRIGGS and J. W. TUSKA, "Design and Operating Characteristics of a High-Bit Density Permalloy Sheet Transfluxor Memory Stack," *Proceedings of the INTERMAG Conference*, pp. 3-4-2, April 1963
7. T. R. FINCH and A. H. BOBECK, "The Waffle Iron Store," *International Solid-State Circuits Conference Digest of Technical Papers*, pp. 12, 1963
8. N. MENYUK and J. B. GOODENOUGH, "Magnetic Materials for Digital Computer Components," *J. Appl. Physics*, Vol. 26, No. 1, p. 8, 1955
9. E. G. FORTIN and H. LESSOFF, "Temperature-Stable Ferrites for Memory Applications," *Proceedings of the INTERMAG Conference*, pp. 1-3, April 1963

A LARGE CAPACITY CRYOELECTRIC MEMORY WITH CAVITY SENSING*

L. L. Burns, D. A. Christiansen, and R. A. Gange
RCA Laboratories, Princeton, N. J.—Walnut 4-2700

This paper describes a cryoelectric high speed Continuous Sheet Memory (CSM) with 16,384 bit locations and complete cryotron addressing matrices. This memory is of the coincident current, random access type. The entire memory plane is fabricated via vacuum deposition techniques and is contained on a 2 inch by 2 inch substrate. In addition to a planar density of 10,000 bits per square inch, and complete cryotron decoding trees, the memory contains a novel sense structure in the form of a geometrical cavity. In addition to other advantages, the new cavity sense eliminates the major alignment problems. In one type of organization, each substrate or memory plane may be regarded as comprising the nth bit of 16,384 words; all n bits of a word may be read out in a parallel manner. Thus a word length is equal to the number of memory planes. This number of planes is at the discretion of the designer. The continuous sheet memory structure with decoding matrices and cavity sense offers a potential means toward the physical realization of a high speed random access memory of a capacity beyond a billion bits with present day technology. Since all present day computers of general utility are memory limited, the complexity of the mathematical operations performed by these computers is restricted by the size and/or access time of the memory. This restriction is indicative that large capacity high speed memories will play a major role in future computers. Many studies have been made of possible techniques that can be utilized for high speed, large capacity random access memories. These techniques have been in the areas of magnetics, semiconductors, and superconductivity. Microminiature magnetic cores, magnetic metal sheets, and deposited magnetic film show high promise toward the realization of a high speed random access medium capacity memory of about 10^7 bits, but do not appear to show promise toward large capacity memories of the order of 10^9 bits. Semiconductors, including transistors and tunnel diodes, also indicate a similar capability toward high speed memories of medium capacity. The present art of transistors is indicative that a very large array of the latter would be costly and cumbersome. Tunnel diodes are two terminal devices and have a relatively complicated memory cell structure. In addition such memories suffer an unavoidable delay due to the high amplification necessary because of signal attenuation; the latter is proportional to the number of words in the memory. A semiconductor device which has been proposed is the cryosar. This device is a two terminal element and awaits future development. In the area of superconductors, film cryotrons and persistors do not appear to offer the potential toward realizing a very large capacity memory of the order of 10^9 bits. The former has a relatively small bit density and registration difficulties; reasonably high uni-

*This work was supported in part by Rome Air Development Center, Griffiss Air Force Base, Rome, New York, under Contract AF30(602)-2722.

formity is required and its relatively large power dissipation in large arrays places an upper bound on the number of cryotrons which may be utilized with present day technology. In addition, an inherent gain problem exists. A major disadvantage of the persistor is the incomplete shielding which exists between the sense and drive lines. Thus arrays of only limited size can be made. A two-hole memory cell described by J. W. Crowe in 1957[1],[2] was anticipated to hold high promise towards the realization of a high speed large capacity memory until it was found that the two-hole cell had an inherent uniformity problem due to the strong dependence of the critical field on the nature and edges of the holes. The continuous sheet memory eliminates the holes of the two-hole memory cell and with the cavity sense geometry represents perhaps the first realistic solution towards the physical realization of a practical high speed random access memory with a capacity in excess of a billion bits. This CSM with cavity sense eliminates the major registration problems encountered with present memory packing densities of 10,000 bits per square inch. In addition, the cryotron decoding matrices are an integral part of the CSM structure. Interrogation of the memory results in negligible power dissipation and no half-select noise has been observed. Present forms of the CSM have a signal-to-noise ratio of about 20:1 and low drive current requirements.

SUPERCONDUCTIVITY

The phenomenon of superconductivity was discovered in 1911 by the Danish physicist Kamerlingh Onnes. Three years after Onnes succeeded in liquefying helium, while attempting resistivity measurements on metals in the new temperature range of 4.2 degrees K, he found that the resistance of mercury completely disappeared. Onnes labeled the new phenomenon "superconductivity". Materials which have been observed to exhibit this property have been called "superconductors". The temperature at which the superconductive state manifests itself is called the "critical temperature". Many superconductors have been discovered since 1911 with critical temperatures falling between .07 degrees K and 18.5 degrees K for a molybdenum-niobium and noibium-tin mixture respectively.

Soon after the discovery by Onnes, it was found that the superconductive state could be destroyed by an external magnetic field, the lowest value of which is called the "critical magnetic field". In addition, Onnes, while experimenting with powerful electromagnetics, found superconductors to have a "critical current". In 1916, Silsbee hypothesized that the destruction of the superconductive state by a critical current passing through a specimen occurred when the magnetic field associated with that current was equal to the critical field. The Silsbee hypothesis is now known to apply only in specific cases. It was thought that a superconductor was in essence an "ideal" conductor until Meissner and Ochsenfeld discovered that a pure superconductor will expel all the magnetic field from its interior as it passes from the normal to the superconducting state. This implies perfect diamagnetism.

The application of an external magnetic field throughout the transition of a multiply connected superconductor from the normal to the superconducting state will result in flux being trapped in the vacancies of the superconductor; upon the removal of the magnetic field, persistent "screening" currents will be established (about the voids) supporting the trapped flux which exists through the voids. In general, flux may be trapped when the superconductor contains impurities, vacancies, and/or normal regions. The depth to which the screening currents substantially exist is called the "penetration depth".

THE CONTINUOUS SHEET MEMORY

Much insight may be gained into the CSM by first briefly discussing the two-hole memory cell whose configuration is illustrated in Fig. 1. The cell consists of a lead (Pb) film with two "D" shaped holes. A strip exists over and is insulated from a superconducting bridge which

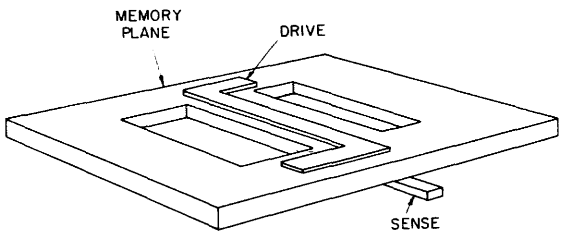


Figure 1. Two-Hole Memory Cell.

separates the two holes. A sense line exists beneath the bridge. A critical magnetic field is made to exist in the region between the bridge and the strip due to a current of sufficient magnitude in the latter. The critical field causes the bridge to change from the superconducting to the normal state permitting the magnetic field to link the bridge. The removal of this current causes the flux which links the superconducting bridge to be trapped in the two holes. The removal of the current gives rise to persistent currents (about the holes) which support the trapped flux. Readout is accomplished by applying a current in the strip of a polarity opposite to that which previously existed so that the magnetic field of the read current constructively adds to the stored magnetic field. The magnitude of the magnetic field of this current is such that when constructively added with the stored field produces a resultant field of a magnitude sufficient to cause the bridge to assume the normal state. As the bridge assumes the normal state, the applied magnetic field links the sense line, and gives rise to a sense pulse indicating the presence of a stored "1". In a similar manner, flux of a polarity opposite to that which previously existed is trapped in the two holes subsequent to the removal of the read current. Note that this constitutes the storage of a "0". The readout is therefore destructive. Assuming a "0" to be stored, the magnetic field of a read current will destructively add with the stored field causing a resultant magnetic field of insufficient magnitude to switch the bridge. The superconducting bridge prevents the applied field from linking the sense line, and no sense pulse is observed. Fig. 2 illustrates a read-write cycle together with the stored current and sense pulse wave forms of the two-hole cell.

The two-hole cell can be arranged to function in an orthogonal coincident current mode as illustrated in Fig. 3. A major disadvantage of this cell is that the magnitude of applied magnetic field necessary to switch the bridge is strongly dependent on the shape and edges of the holes implying an inherent uniformity problem.

The continuous sheet memory eliminates the uniformity problem while retaining the advantages of the two-hole memory. Figure 4 illustrates a single cell CSM without cavity sense consisting of a superconducting tin film, a lead sense line, and lead drive strips. The latter are orthogonal to, and insulated from one another, and are insulated from the storage plane over which they pass. The sense line is beneath the storage plane, and is oriented diagonally to and directly under the intersection of the drive lines. Coincidence of the "X" and "Y" drive currents at the intersection of the drive strips produce magnetic fields which when vectorially added, produce a field of a magnitude sufficient to switch the region of storage plane beneath the intersection. In a manner similar to that of the two-hole memory, the region beneath the intersection of drive strips returns

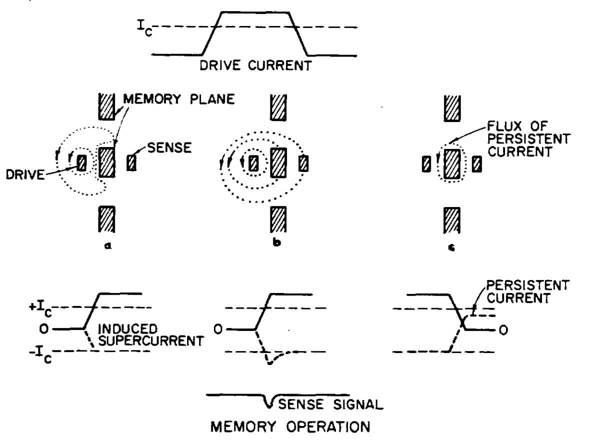


Figure 2. Operation of Two-Hole Memory Cell.

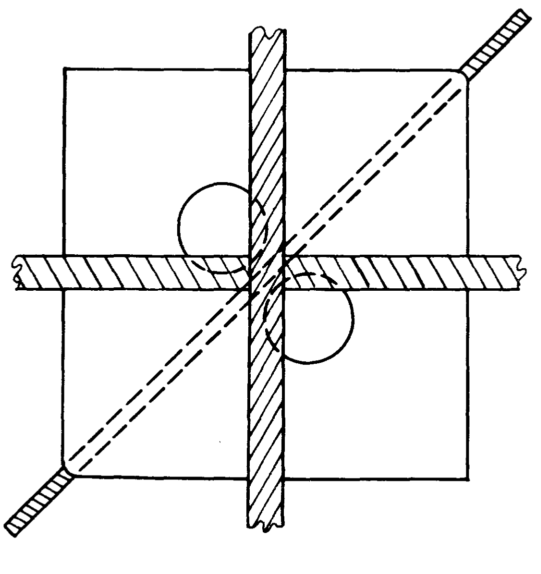


Figure 3. Persistent Current Memory Cell.

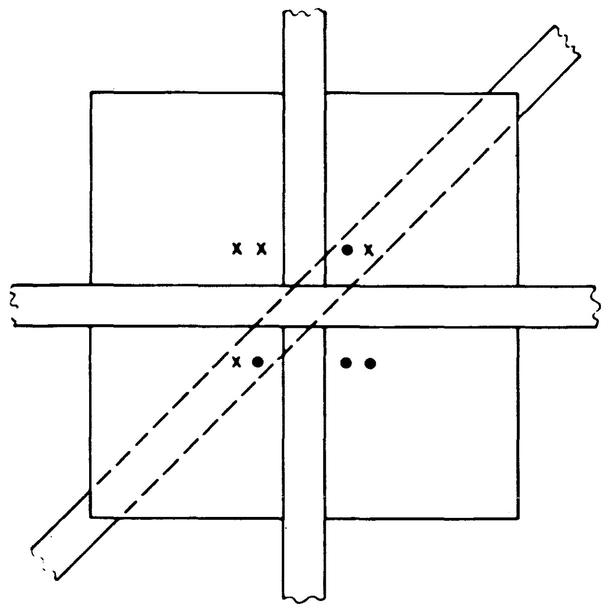


Figure 4. Continuous Sheet Memory Cell.

to the superconducting state forcing the flux linking it to be trapped in the storage plane. Upon removal of the drive currents persistent currents are established which support this flux. If subsequent drive currents are polarized so as to have their magnetic fields constructively add to that of the persistent currents stored in the tin film, the magnitude of the resultant field near the intersection at which the drive currents are coincident will be sufficient to cause the region of storage plane beneath the intersection to assume the normal state, and a voltage will appear across the sense line. If the drive currents are polarized so as to have their magnetic field subtract from that of the stored current, the net field acting at the region of film beneath the aforementioned intersection will be of a magnitude insufficient to cause the onset of the resistive state of this region and no voltage will appear across the sense line.

Of particular interest is the remarkable shielding ability of the storage plane. Theoretical estimates[3] indicate that for a storage plane whose thickness is equal to the penetration depth, the field on the surface of the storage plane in proximity to the sense line is less than 1 percent of the field which exists at its other surface. The superconducting storage plane's remarkable shielding of the drive signal from

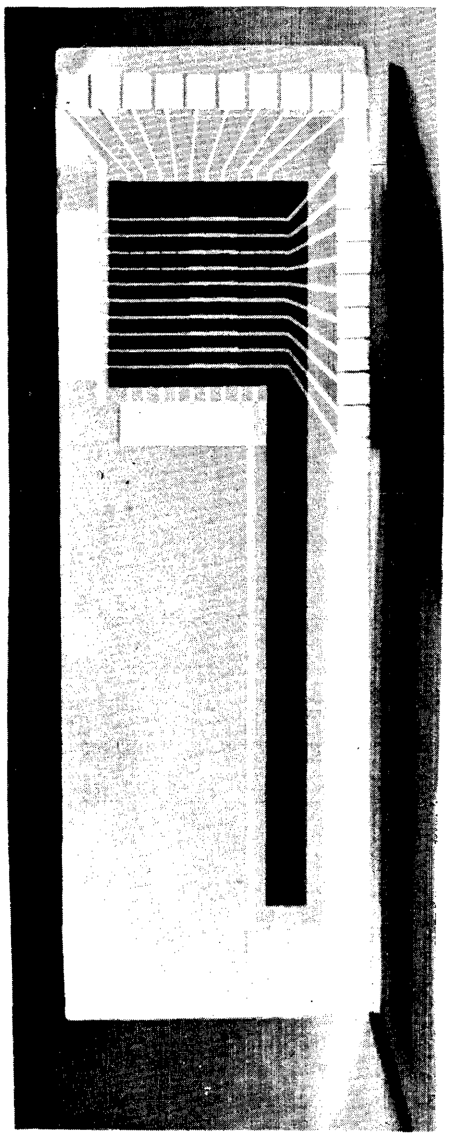


Figure 5. 100 Bit Memory Plane.

the sense circuit has been experimentally verified. In addition, early experiments on the continuous sheet memory[4] have indicated that the cell-to-cell drive current uniformity can be held to within 1 percent. These experimental data are typified by tests announced by L. L. Burns et al. in their paper at the ONR symposium in 1960 on a CSM containing 100 bit locations with 10 "X" and 10 "Y" lines (10 $\times$ 10 array). This is shown in Fig. 5. In essence, the 10 $\times$ 10 array consisted of a superconducting tin film, a zig-zag sense line, and "X" and "Y" superconducting lead (Pb) drive lines. As with the single continuous sheet memory cell, the drive

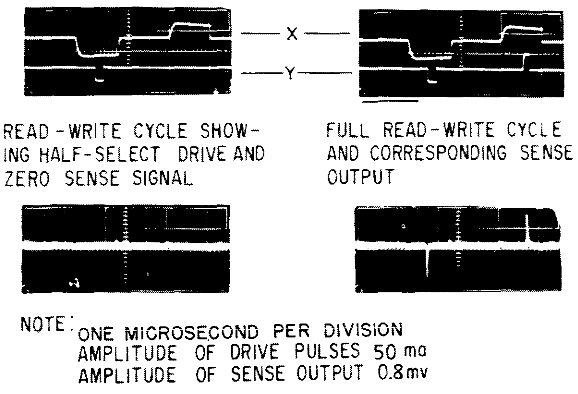


Figure 6. Waveforms for 100 Bit Memory Plane.

lines are orthogonal to and insulated from one another. Both sets of drive lines are insulated from the tin storage plane. The zig-zag sense line is beneath the storage plane and is oriented diagonally to and directly under each drive line intersection. The drive currents and sense voltage waveforms for the 10 by 10 array are as shown in Fig. 6. Drive current amplitudes of 30–80 ma and sense voltages between .8 and 8 millivolts were reported.[4]

The major disadvantage of the zig-zag sense line is that its length becomes prohibitively large with increasing memory size. The zig-zag sense line si about 20 feet in length for a 16,384 bit memory plane with a packing density of 10,000 bits per square inch; a length of sense line of this order of magnitude necessitates a delay between the onset of the memory read currents and the ensuing memory sense output pulse of about 30 nanoseconds. In addition, the excessive inductance of a zig-zag sense line of this order of magnitude tends to reduce the amplitude of the sense output voltage. The following paragraphs describe a cavity sense structure which eliminates the aforementioned disadvantages of the zig-zag sense line.

The cavity geometry offers the potential of very large high speed memory structures not before possible with existing technology.

The continuous sheet memory with the cavity sense structure consists of a storage plane, a "sense tongue" and "X" and "Y" drive lines. The latter are orthogonal to, and insulated from one another, and are insulated from the memory plane. The sense tongue is beneath the memory plane. Both the sense tongue and memory plane films are electrically connected along one edge and together they constitute the cavity geometry. If the drive currents are polarized such that their magnetic fields constructively add to that of the stored currents, the resultant magnetic field which exists at the region of memory plane beneath the intersection at which the drive currents are coincident will be sufficient to destroy the superconducting state of this region; as a consequence, a change of the flux occurs within the cavity beneath the aforementioned intersection. The time varying magnetic and electric fields give rise to a sense pulse at the output tabs. Fig. 7 depicts a three dimensional illustration of the cavity structure.

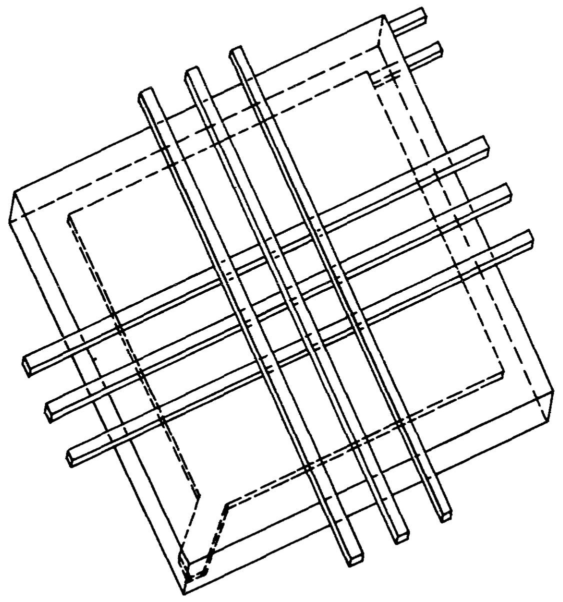


Figure 7. Memory with Cavity Sensing.

A CSM 4×4 array with cavity sense was designed and tested. Fig. 8 illustrates this array. Over 60 such memories have been fabricated and tested with the accumulation of extensive experimental data. Fig. 9 shows typical memory drive currents and cavity sense output waveforms.

Fig. 10 is a photograph of the 16,384 bit continuous sheet memory with cavity sense and cryotron addressing matrices. Interrogation of a given cell is achieved via the introduction of drive currents at the input nodes of the "X" and "Y" selection trees subsequent to the exist-

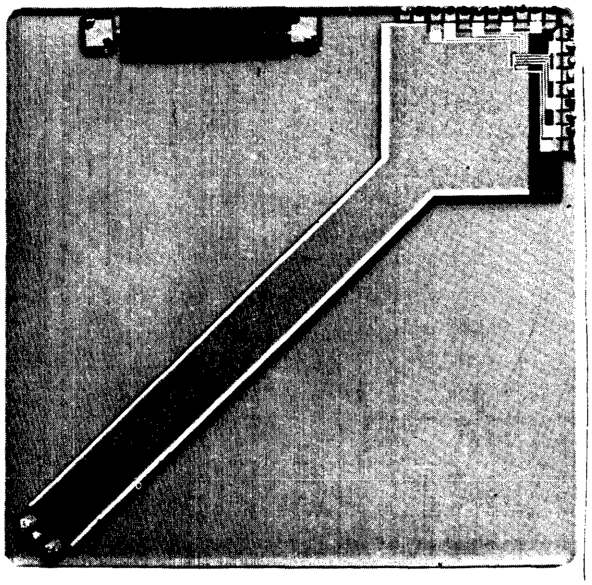


Figure 8. 16 Bit Test Sample.

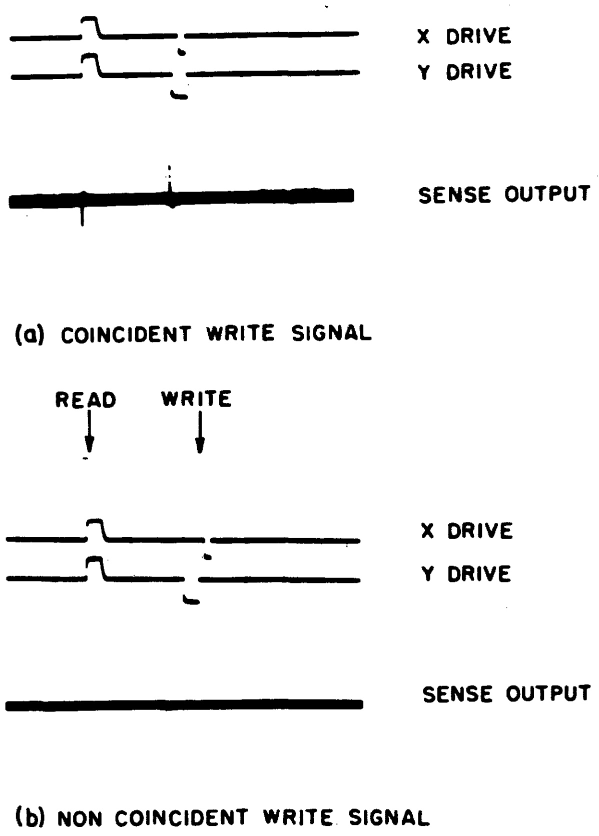


Figure 9. Oscilloscope Photographs of Drive and Sense Signals.

ence of control currents to the cryotrons of these trees. The selected cryotrons in either tree maintain a resistance in all but the desired

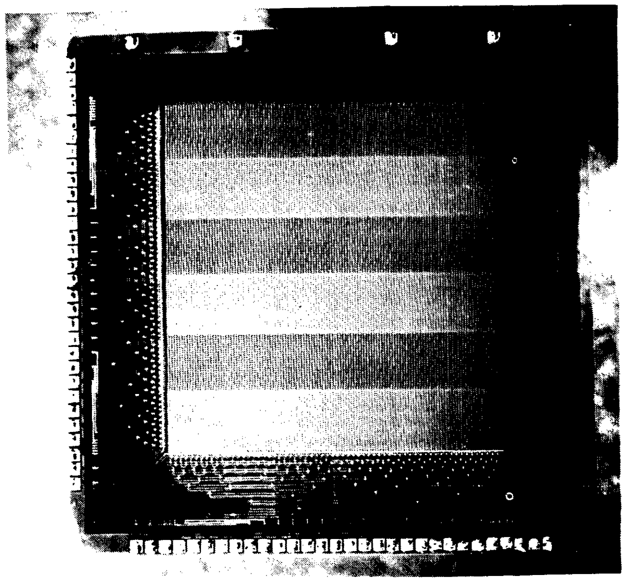


Figure 10. 16,384 Bit Memory with Cryotron Address Matrix.

(superconducting) paths, thereby causing coincidence of the "X" and "Y" drive currents over the storage planes at the intersection of the two selected paths. Analysis indicates that for a separation of drive strip to memory plane of 3000 Angstroms and a tin cryotron gate thickness of 5000 Angstroms with unity crossing ratio, a time constant of the selection tree of about .5 μsec may be realized. Fig. 11 depicts 5 levels of a seven-level decoding tree.

The 16,384 bit memory was fabricated via multilayer vacuum deposition techniques.[5] A total of 16 masks were used in the 26 step

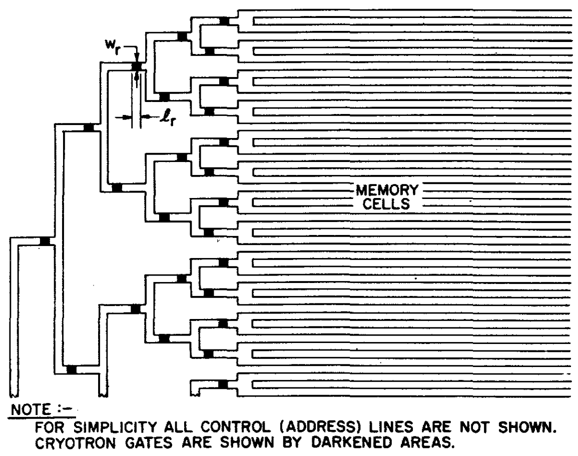

w_r
ℓ_r
MEMORY CELLS
NOTE :-
FOR SIMPLICITY ALL CONTROL (ADDRESS) LINES ARE NOT SHOWN.
CRYOTRON GATES ARE SHOWN BY DARKENED AREAS.

Figure 11. Cryotron Tree.

evaporation. The evaporation masks were 2 mils thick and were fabricated via a photoetching process; the distance between the mask and the 43 mil thick 2″ × 2″ substrate was about 5 mils. The width of the memory drive lines was 5 mils and silicon monoxide was used for the insulation. Figures 12, 13, and 14 are photographs of the vacuum system with mask changer used for the fabrication of this 128 × 128 array. Figure 15 illustrates an exploded view of the memory plane with the decoding hardware and cavity sense structure. Figure 16 is a photograph of the apparatus used for the testing of the 16,384 bit memory plane. Figure 17 is a block diagram of the electrical interconnection of this apparatus. Currents of approximately 500 ma were used to address the cryotrons of the selected trees. Drive currents of about 120 ma were used to interrogate the memory cells with an overall selection tolerance of about 20 percent. Sense voltages of about one millivolt were observed.

CONCLUSIONS

Large capacity high speed computer memories containing upwards of one billion (10^9)

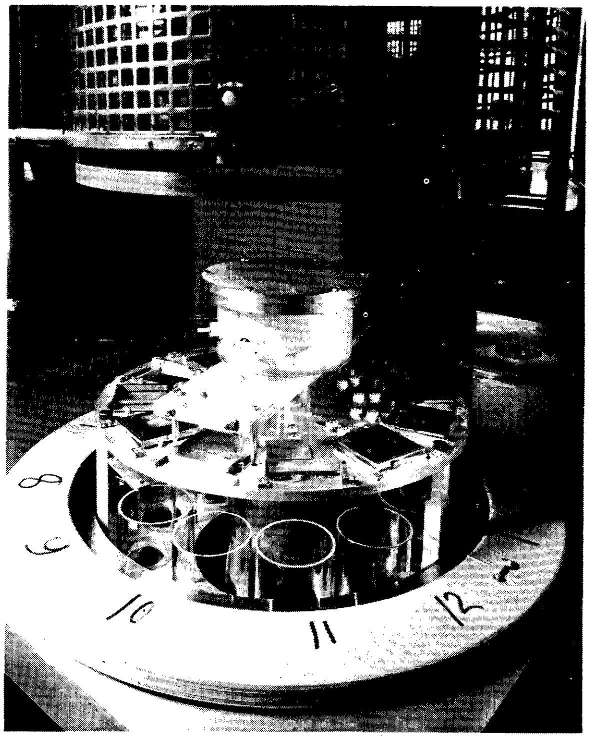


Figure 13. Evaporator Number 2 Showing Masking Arrangement.

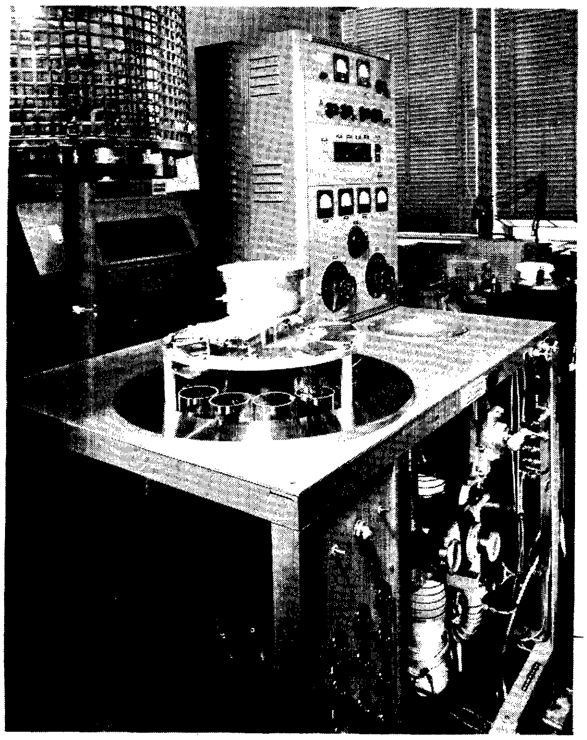


Figure 12. Evaporator Number 2.

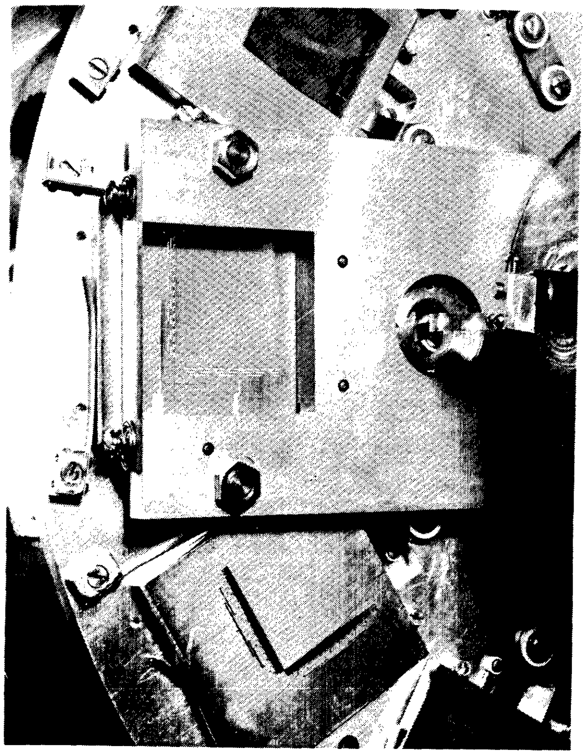


Figure 14. Cryotron Tree Mask.

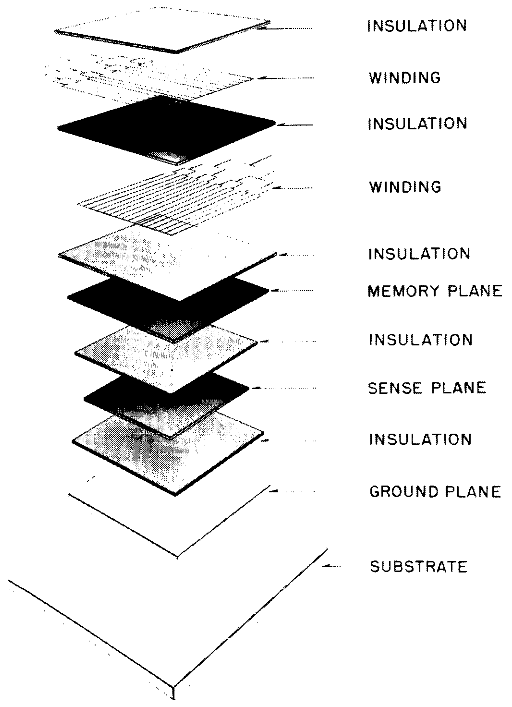


Figure 15. Exploded View of 16,384 Bit Cryoelectric Memory Plane.

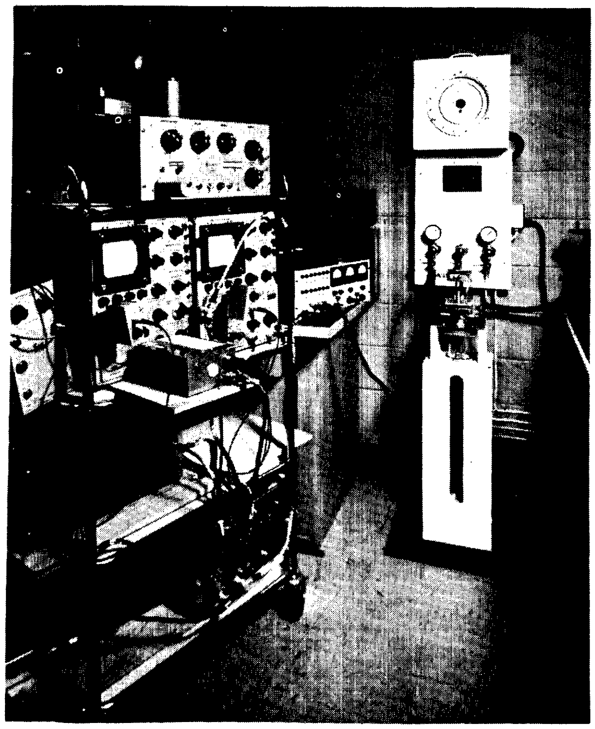


Figure 16. Test Station.

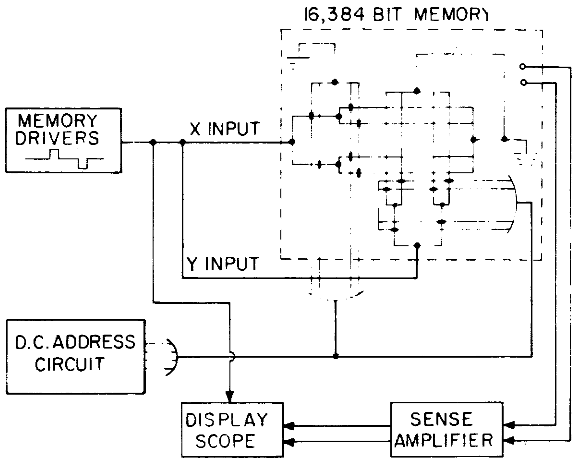


Figure 17. Memory Testing Apparatus.

bits are demonstrated to be feasible with the techniques described. For many years superconductivity has been considered to be the only known phenomenon that could possibly be used for building really large high speed memories. The work described has shown that indeed superconductivity does hold this promise. A modest beginning has been made in the construction of operable planes containing 16,384 memory positions and 508 cryotrons in the associated X and Y selection trees. The principal advantage of superconductivity over magnetic techniques has been shown to be realizable in practice, i.e., the perfect shielding effect of a superconducting plane does, in fact, completely eliminate half select noise from unselected cells thus allowing the construction of very large memory arrays. Only mechanical restrictions on how large an area can be evaporated uniformly coupled with limitations in registration of successive evaporation mask patterns limit the size of memory arrays. Planes containing over one million bits are contemplated for the near future.

Cavity sensing of the stored information provides an elegant means of memory readout and greatly simplifies the construction of the required evaporation masks. Tests and measurements have shown that time delay effects are also reduced by cavity sensing while the magnitude of the available sense output voltage is not significantly affected.

REFERENCES

1. "Trapped-Flux Superconducting Memory", J. W. CROWE, *IBM Journal*, October, 1957. 1957.
2. "An Analysis of the Operation of a Persistent-Supercurrent Memory Cell", R. L. GARWIN, *IBM Journal*, October, 1957.
3. "Field Distributions and Thermodynamics of a Lead Strip over a Superconducting Tin Ground Plane", R. A. GANGE. To be published.
4. "Coincident-Current Superconductive Memory", L. L. BURNS, G. A. ALPHONSE, G. W. LECK, *IRE Transactions on Electronic Computers*, Vol. EC-10, No. 3, September, 1961.
5. "Vacuum Techniques for Fabricating Integrated Cryoelectric Computer Devices", G. W. LECK, American Vacuum Society, 10th National Vacuum Symposium, October 16-18, 1963, Boston, Mass.

FIXED, ASSOCIATIVE MEMORY USING EVAPORATED ORGANIC DIODE ARRAYS

M. H. Lewin, H. R. Beelitz, and J. A. Rajchman
RCA Laboratories
Princeton, N. J.

INTRODUCTION

The use of fixed or read-only memories to store permanent or semi-permanent data has been widely treated in the literature.[1] Associative or content-addressable memories, discussed more recently,[2] provide the important capability for rapid parallel searching and retrieval of stored information. A fixed, associative memory can be used in applications requiring storage of encyclopedic data which must be searched at very high speed. Large capacity stores such as library and other catalog search files, language translators and medical diagnostic tables, as well as small stores such as code converters, computer program memories and other real-time search files, may be of this type.

A magnetic realization of a fixed, content-addressed memory was discussed by Goldberg and Green[3] in May 1961. An interrogation routine to resolve multiple responses in this type of file was described by Frei and Goldberg[4] in December 1961.

The work on which this paper is based began, first, with the realization that a symmetrical diode matrix exhibits the basic retrieval properties of an associative memory and, second, with the development of techniques to fabricate diode arrays by vacuum evaporation of organic films.[5] One result of this investigation already published,[6] was a very efficient algorithm for retrieval of multiple "matches" from a file of this type.

SYMMETRICAL DIODE MATRIX

Consider the diode array shown in Fig. 1. It is composed of w word lines (rows) and b pairs of bit lines (column pairs) and therefore stores w words, each b bits in length. At every row and column-pair intersection is a diode, one of whose terminals is connected to the word line. A "0" is stored by connecting the other terminal of the diode to the right column of the pair. Connecting the diode to the left column indicates

[1] An extensive bibliography on fixed memories can be found in the survey paper by D. M. Taub, "A Short Review of Read-Only Memories," Proc. IEE (British), vol. 110, No. 1, pp. 157–166, January 1963.

[2] A comprehensive bibliography on associative memories can be found in the survey article by A. Corneretto, "Associative Memories," Electronic Design, Vol. 11, No. 3, pp. 40–55, February 1, 1963. A set of three papers on associative memories is published in Proc. Pacific Computer Conference, Cal. Tech., Pasadena, Calif., pp. 96–130, March 15–16, 1963.

[3] J. Goldberg and M. W. Green, "Large Files for Information Retrieval Based on Simultaneous Interrogation of All Items," Proc. Symposium on Large Capacity Memory Techniques for Computing Systems (book), edited by M. C. Yovits, published by Macmillan Co., N. Y., pp. 63–77, May 23–25, 1961.

[4] E. H. Frei and J. Goldberg, "A Method for Resolving Multiple Responses in a Parallel Search File," IRE Trans. on Electronic Computers, Vol. EC–10, No. 4, pp. 718–722, December 1961.

[5] A. Sussman, "Rectification in Evaporated Thin Films of Copper Phthalocyanine," Joint Symposium in Thin Films, Electrochemical Society, Pittsburgh, Pa., April 14–18, 1963.

[6] M. H. Lewin, "Retrieval of Ordered Lists From a Content-Addressed Memory," RCA Review, Vol. XXIII, No. 2, pp. 215–229, June 1962.

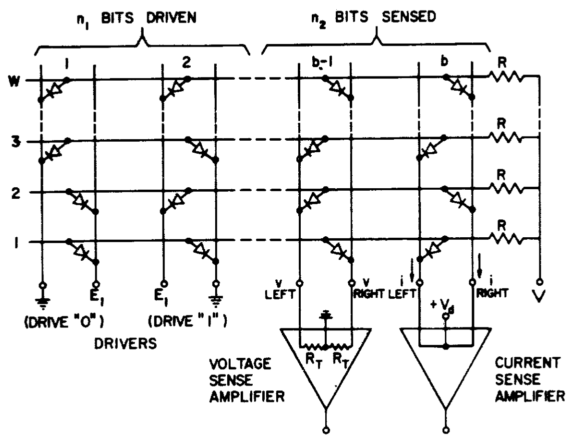


Figure 1. Diode Associative Array.

a "1" stored. Thus, the pattern of connections determines the information stored in the array. Every row is returned, through a resistor R, to a common voltage source V. For sufficiently large R and V, it sees a relatively constant current source looking into R. Each of the columns is terminated with either a driver or a sense amplifier. A given column-pair (bit) is driven with a "0" if the left column is grounded while the right column is connected to a voltage source E_1. Alternatively, one drives a "1" by reversing these conditions. Thus, pairs of bit lines are always driven with complementary signals. The sense amplifiers may be either voltage or current amplifiers, both of which are illustrated in Fig. 1. R_T may be considered as the input impedance of a voltage amplifier. V_d, the voltage to which the input of a current amplifier is returned, is the maximum forward drop across a memory diode conducting a current V/R. A positive sensed voltage v or positive sensed current i indicates a "1" sense signal. No sensed voltage or current denotes a "0". To keep the discussion general, assume each sense signal is amplified by a separate sense amplier (two per column-pair). Under some conditions a single sense amplifier per sensed digit may be adequate. A difference amplifier for each pair of sensed columns can also be used.

One can divide the matrix into two parts: n_1 bits driven (inputs) and n_2 bits sensed (outputs), where $n_1 + n_2 = b$. In Fig. 1 the left portion is driven while the right portion is sensed. (Note that every row is the output terminal of an n_1-input diode AND gate.) Assume, for a given set of inputs, only one row develops a relatively positive ("1") voltage (i.e., it is selected). All other rows are clamped to ground (inhibited) and each of their currents V/R is steered to ground through at least one conducting diode connected to a grounded column wire. Those sensed columns which are coupled via diodes to the selected row will also be driven positive. Alternatively, one can say that the V/R current for the selected word is steered to split among those of its diodes which are connected to sensed columns. (Note that each sensed column is the output terminal of a diode OR gate. The number of inputs to the gate whose output terminal is the left (right) column is equal to the number of words which store a "1" ("0") in that bit position.) Thus, the pattern of sensed voltages or currents will correspond to the pattern of diode connections between the selected row and sensed columns. Such an arrangement can be used as a decoder-encoder combination or as a fixed memory. In the first case, one is translating from an n_1-bit code to an n_2-bit code. In the second case, the n_1 bits are the address of an n_2-bit word stored. More generally, this array can be used as a content-addressed memory as is explained below.

The important property of such a matrix is the symmetry or reversability involved. For example, one can reverse input and output roles (i.e., drive the bits on the right side and sense those on the left side) to obtain an n_2-bit to n_1-bit decoder-encoder. In fact, any column-pair can be either driven or sensed—that is, serve as an input or an output. Thus, in the general case, any arbitrary number of bits, scattered in any manner among the b bit positions, can be driven while the remaining bits are sensed. The fact that this yields a content-addressable memory can be made clear with a simple example. Suppose one asks for retrieval of all words (assume there is only one, to begin with) which answer the following specification: "0" in the 1st place, "1" in the 4th place, "1" in the 8th place, etc. All of the column-pairs in the positions designated above are driven with polarities corresponding to the specified bits (tag bits or descriptors). All unspecified digit line pairs (i.e., the 2nd, 3rd, 5th, etc.) are sensed. Thus, sensing a pair of columns corresponds to a "don't care" or "Ø" specification for that bit. (The interrogation word for this case is ØØØ1ØØØ1ØØ . . .) Clearly, for the most general associative memory, each pair of bit lines must be terminated in a combination driver-sense amplifier which can be switched between "drive" and "sense" in accordance with the specification for that bit in the interrogation word. Circuits developed for this purpose will be described later.

When the drivers and sense amplifiers are set up corresponding to the interrogation word given, the selected word line assumes a relatively positive voltage while all others remain nearer to ground potential. The pattern of sensed voltages or currents corresponds to the pattern of diode connections for the selected word, so that all the other bits in the word answering the description given are read out simultaneously.

MULTIPLE-MATCH READOUT

More generally, more than one word is selected by a given specification and all must be read out in some ordered fashion. This problem is common to all physical realizations of associative memories. There are, broadly, two solutions to the problem. One involves incorporating in the memory array, appropriate circuits to allow for sequential activation and readout of selected words. The other involves manipulation of the interrogation word, outside of the array, in such a manner that all selected words are isolated and read out in sequence. This latter method has proven very well suited for use with the diode matrix. An algorithm has been developed[6] which requires no modifications of the basic memory array and yet retrieves all selected words in less than two memory cycles per word. This is true, independent of the total number of words stored in the file and independent of the number of bits per word.

The interrogation algorithm relies on the fact that *two* columns per bit are available for sensing. If the memory is driven so that only unique selections occur, only one column per bit need be sensed, since the output signals of any column-pair will always be complementary (i.e., 0, 1 for a "0" stored; 1, 0 for a "1" stored). The first use of sensing both columns of a bit is simply to detect when *no* word stored answers the description given. If there are no words selected, all rows are clamped to ground. Thus, a 0, 0 detected at *any* sensed column-pair gives this indication immediately. If a multiple-selection is made, as is usually the case, more than one row assumes a relatively positive voltage. If all words selected (isolated) have the same bit in a given position being sensed, the sense signals for that bit are similar to those detected for a unique selection of one of these words. One can say that the sense signals are "reinforced" or stronger in the multiple-selection case. More often, however, some of the words selected have a "1" stored in a particular sensed position, while the others have a "0" stored there. In this case, *both* columns of that bit are driven positive, or carry a positive sense current (i.e., 1, 1 detected), since each is coupled through at least one diode to a selected row. One can say that an "x" has been sensed in this case.

The ability to detect the above conditions, coupled with the fact that any bit can be either driven or sensed, allows one to efficiently retrieve all words selected. Combination driver-sense amplifiers, controlled by external logic inputs, are used. The interrogation routine is based on the fact that, as one converts sensed-"x" bits to driven bits, smaller sets of words are selected. In this manner, we have, with appropriate external logic, made successive interrogations converting sense amplifiers (sensing "x") to drivers and drivers to sense amplifiers so that all words originally selected are isolated and read out individually. The interrogation sequence is generated based on a set of rules by which the interrogation pattern for a given cycle depends on the pattern and the sensed results of the previous cycle. One effectively generates a "decision tree" in this manner.

This algorithm is described in detail in the paper[6] referred to earlier. Included in the paper is a detailed flow chart for generating the interrogation sequence, a comparison of this routine with other published work dealing with the same problem, a complete per-digit logic design for mechanizing the routine and a proof that retrieval of m words always takes exactly 2m—1 memory cycles, independent of the number of words in the memory or the number of bits per word. Since the paper deals primarily with the logic of the routine, applicable to any physical realization which allows "column-pair sensing" and simultaneous activation of all selected word lines, no specific mention is made of fixed memories or diode matrices. (A cryogenic implementation is described only as an example.)

One point to note is that, while some physical realizations would require a word-driver per word to achieve simultaneous activation of all selected (or "matched") word lines, the diode array described above *automatically* furnishes this without the need of word-drivers. (I.e., every selected row develops a positive voltage which *directly* couples to sensed columns via its diode pattern.) This is a fundamental reason why the algorithm is particularly well-suited for a fixed memory configuration of the type discussed above.

EVAPORATED DIODE ARRAYS

A large diode matrix serving as a fixed, associative memory is economically feasible only if sizeable arrays of diodes can be fabricated at sufficiently low cost. A new technique for constructing integrated arrays of thin-film diodes has recently been described.[5] It embodies vacuum evaporation of an organic semiconductor, copper phthalocyanine. Diodes fabricated by this technique were used experimentally to implement the associative memory concept considered here.

A multiple organic diode card is shown in Fig. 2. Each 4″ × 4″ × .020″ board holds 128 diodes which are distributed around the outline of the dark phthalocyanine inner square. All diodes have a common cathode, connection to which is made at any corner of the card. (Diode polarities for all of the experiments described are the reverse of those given in Fig. 1.) The card shown can be used to store one 128-bit word, with the common cathode conductor being a row wire of the matrix. A set of n such cards, completely interconnected, comprises an n-word, 128-bit-per-word, memory. Two etched wires per diode form the anode connection, "fanning out" to the card edge. These allow connection of a given diode anode to both of its associated column wires. Information can be written in

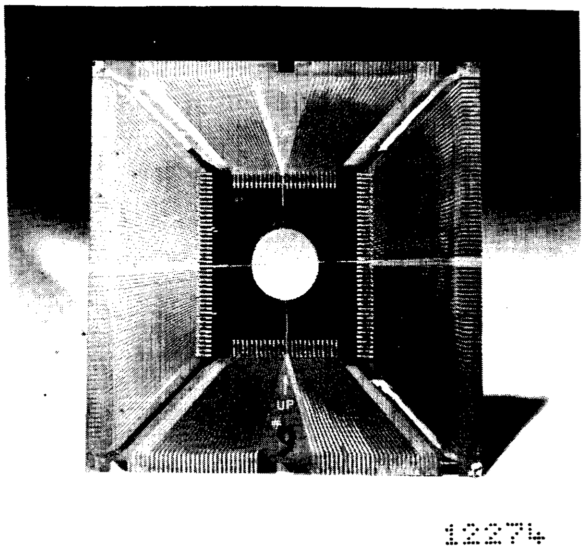


Figure 2. Evaporated Organic Diode Card.

by breaking the appropriate anode connections in any of a number of ways such as, for example, by punching a series of small holes in the card. The card pattern shown was chosen to allow a relatively small area for the evaporation of the diodes, in order to insure the desired uniformity of characteristics.

The static characteristic of a typical evaporated thin-film diode is shown in Fig. 3. With a diode area of 3.5 mm^2, the voltage drop is approximately 1.5 volts at a forward current of 2 ma. Its rectification ratio is approximately 10^5.

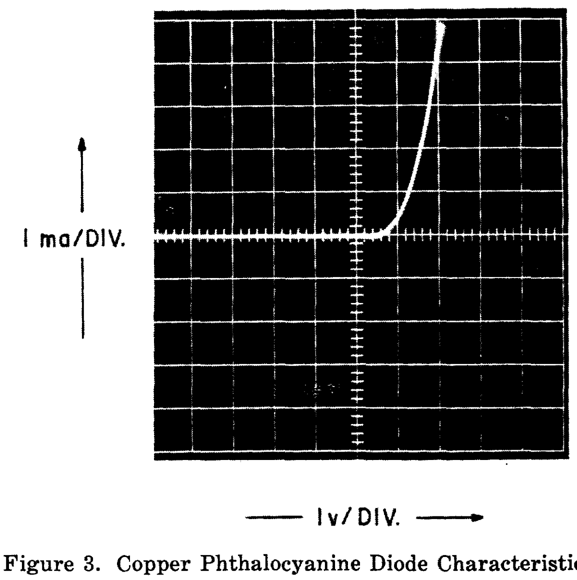


Figure 3. Copper Phthalocyanine Diode Characteristic.

Each diode consists of a three-layer sandwich, the middle layer being copper phthalocyanine and the outer layers being metal anode and cathode electrodes. Since the diode is such a thin element, its equivalent parallel plate or "case" capacity is large, being typically, for the area given above, approximately 50 pf.

EXPERIMENTS

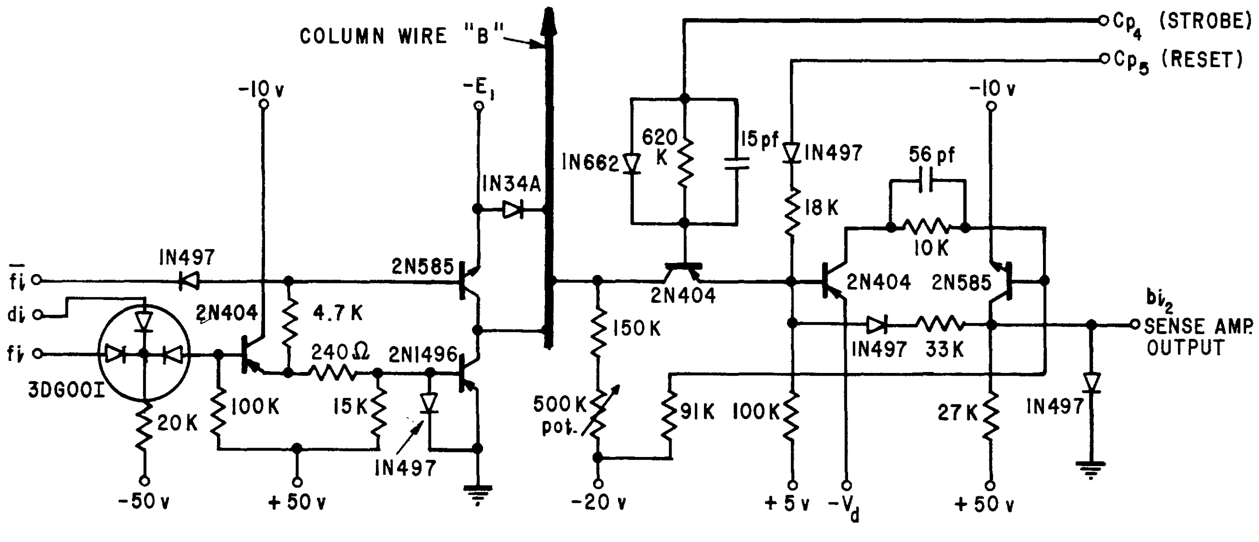
a) *Peripheral Electronics*

The circuit design of the peripheral electronics system which was constructed to exercise and test various embodiments of the diode associative memory was largely conventional (primarily using RCA 2N404 transistors). The apparatus consists of all the logic and the combination driver-sense amplifiers for implementing the search routine[6] for a memory word length of 10 bits. Display of retrieved words, memory cycle count and driven or sensed states, for each bit, are included.

The requirement of a combination driver-sense amplifier controlled by external logic inputs, resulted in a somewhat novel design. A schematic diagram of the circuit is indicated in Fig. 4. When in the "drive" state, it can furnish the memory with up to 0.5 amps at ground potential. This would allow the circuit to drive a 1000 word (100 bit per word) diode associative array. It is also capable of detecting 10 μa of signal current when in the "sense" state and strobed (CP_4 is the strobe pulse). The circuit will switch between "drive" and "sense", as demanded by the input signal f_i, in 0.5 microseconds. The function table shown in the figure relates the circuit state (drive or sense) to the logic input states. Two such circuits are required per column pair.

b) *500-Bit Diode Matrix*

A small, conventional-diode array, constructed as a 50 word, 10 bit-per-word manually-alterable memory, served as initial experimental



DRIVER-SENSE AMPLIFIER FUNCTION TABLE, COLUMN "B"

fi	f̄i	di	FUNCTION
0 v	-10 v	0 v	SENSE DURING STROBE PULSE DRIVER PRESENTS A HIGH IMPEDANCE
0 v	-10 v	-10 v	SENSE DURING STROBE PULSE DRIVER PRESENTS A HIGH IMPEDANCE
-10 v	0 v	0 v	DRIVE "0"
-10 v	0 v	-10 v	DRIVE "1"

NOTE :- COLUMN "A" DRIVER-SENSE AMPLIFIER IS IDENTICAL BUT WITH d̄i REPLACING di. THIS PROVIDES THE REQUIRED COMPLEMENTARY DRIVE.

Figure 4. Driver-Sense Amplifier Circuit.

verification of the basic idea of a diode associative memory and of the electronic mechanization of the interrogation routine. Various patterns of information were written into the array by manually changing the pattern of diode connections.

A number of tests were made with this matrix. These include the lexicographic ordering of the entire memory contents and a large number of searches using a wide variety of input descriptor patterns. (Descriptors are inserted using a bank of toggle switches.) For each test series, the interrogation routine can be manually stopped along and monitored or can proceed automatically at a maximum rate of 100 kc (10 microsecond cycle time). The total number of cycles required for each retrieval was monitored by a counter and could be compared with the predicted count.

c) *Organic Diode Array*

Fig. 5 is a photograph of a stack of organic diode cards, each of the type shown in Fig. 2, which was successfully operated as a small test memory. Although each board contained 128 diodes, only 10 bits on each card were used for the tests because of the limited amount of peripheral electronics which was then available.

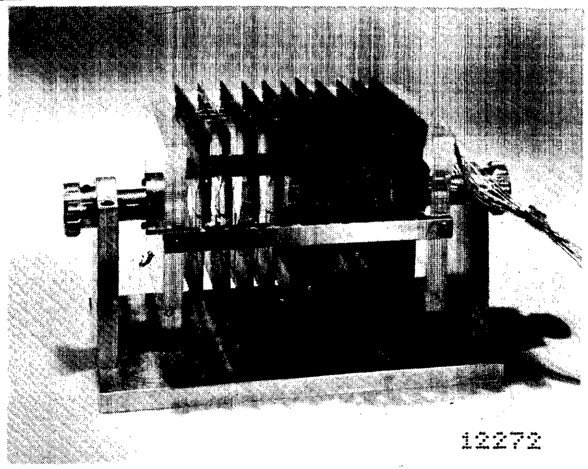


Figure 5. Organic Diode Test Memory Stack.

The diode cards are mounted in a metal frame with specially-designed "finger" connectors used for each card. Column wires running down the outside of the card stack are soldered to the finger connectors and provide the required card-to-card interconnection. The information stored is determined by the pattern of connections to these column wires. This allowed one to change the information stored without removing the cards and was convenient for the tests made.

Various search and retrieval tests were performed with the organic diode memory. In general, the experiments duplicated those performed with the conventional diode matrix. They demonstrated the feasibility of using thin-film organic diode arrays as digital elements.

CONCLUSIONS

The purpose of this paper has been to explain the associative properties of a symmetrical diode matrix, including the applicability of a very efficient interrogation algorithm to resolve multiple-matches, and to describe experiments which verify these concepts and which, in addition, utilize arrays of new thin-film diodes. Clearly, the physical realization of this type of diode matrix can be in many forms and may well involve other diode array fabrication technologies, including those now developing in the semiconductor industry.

ACKNOWLEDGEMENTS

The authors wish to thank A. Sussman, who did the basic experiments which resulted in the evaporated, copper phthalocyanine diodes and who designed and fabricated the diode cards. We are also indebted to O. Kornehed for assistance in peripheral circuit design, J. Schumacher, for measurements on the organic diode parameters, J. Guarracini, for the design of the "finger" connectors and H. Schnitzler, who constructed the test equipment and assisted in many of the measurements.

GENERALIZED MULTIPROCESSING AND MULTIPROGRAMMING SYSTEMS

—Status Report—

A. J. Critchlow
IBM Corporation
San Jose, California

1.1 DEFINITIONS

In this paper, the following definitions have been followed:

1. *Multiprogramming*—the *time-sharing of a processor by many programs operating sequentially.* Many programs are available and in memory but only one program is actually being executed at a given time. Control of object programs is provided by a supervisory control program. Thruput is highest when many programs can be interleaved to use hardware most efficiently. In general, the time required to complete a selected program will be increased over single program operation.
2. *Multiprocessing—independent and simultaneous processing accomplished by the use of several duplicate hardware units.* Specifically, duplicate logical and arithmetic units are assumed, although systems with separate input-output channels can also be said to be multiprocessors. Note that "processors" do not include storage units while "computers" do. (Table 1.2.2)
3. *Scheduling*—is the determination of the sequence in which job programs will use the available facilities. Scheduling assignments are based on the availability of all required facilities, the priority of the job program and the relative priorities of other programs. Scheduling algorithms aim to optimize performance of the system with respect to chosen goals.
4. *Allocation*—is the assignment of particular facilities: core memory, tapes, disk files to a job program.
5. *Interrupt and Trapping* are considered synonymous. Both mean the ability, provided by hardware, to monitor particular conditions in the system during execution of all other operations and to provide an alarm signal which can interrupt a processor to obtain required action. Program interrupts or Intentional interrupts are really branching operations which sometimes use the alarm signal hardware.

1.2 BACKGROUND

1.2.1 *Development of Multiprogramming*

Multiprogramming is expected to be more efficient than single-program operation because facilities are used which would be idle otherwise. It is necessary that the control cost of multiprogramming be less than the increased output of useful work if a net gain in efficiency is to be achieved.

The first approach to multiprogramming was to select or match two or more programs so that better utilization of facilities was obtained. Scientific programs, in general, provide a heavy load on the processor and a light load on peripheral equipment. Business data processing tends to load peripherals in order to produce the sorted data and output of printed reports required. Combining these two types of operation uses facilities more effectively. Codd (1) reports timing improvements of 2 to 1 when multiprogramming mixed program sets.

An added complexity is introduced, however, because both programs may need the same facility simultaneously, so one of them must wait. In more complex operations with many programs and perhaps more than one processor, the sequencing of operations becomes quite difficult.

At first the programs to be run together were assembled onto one magnetic tape with sequencing information included on the tape so the two programs were, in effect, just one large program. Running programs this way is efficient if all the programs are production programs which can be run on a regular schedule. When one program must be altered or deleted, it is necessary to reassemble the program tape at a considerable time cost.

When control of multiprogramming operation is turned over to an executive program and there are suitable hardware provisions for interrupt, memory protection, priority control, etc., it is possible to write each program as though it alone is being run. The multiprogramming sequencing, queuing and input-operation task is handled by the Executive program. Efficient operation requires that many programs be available ready to run so that the Scheduler or Sequencer program will have several possible choices to maximize operational efficiency. (Table 1.2.1)

An example of the dynamic scheduling of many programs to run together on the same system is worked out in section 4.2.2.

Communication between programs is necessary so that branching to subroutines can be accomplished. One solution is to have a "common" area of memory for subroutines used by several programs. A more flexible method utilizes a "universal" symbol which is recognized by the supervisory program. The supervisory program maintains a table of addresses for subroutines and supplies the required address when signalled by use of the "universal" symbol request.

Table 1.2.2—Classification of Functional Types

System	*Data Processor*	*Instruction Processor*	*Input-Output Processor*	*Switching Central*	*Storage Processor*	*Ref.*
CDC 3600	Computation Module	Computation Module (overlapped memory operation)	Housekeeping Module or Data Channel	Multiple Gates & Registers on Storage Module	Storage Mod. (8 prs of 16,384 wds. ea. (access overlapped)	3
Burroughs D-825	Computer Module	Computer Module	Input/Output Control Modules & Automatic I/O Exchange Crossbar Switch (64 devices)	Crosspoint Switch Matrix (4 × 16) Bus Allocator (priority basis)	Memory Module (overlapped operation) (16 of 4096 wds. ea.)	4
Pilot-Multiple Computer System	Primary Computer	Secondary Computer	Format Computer (I/O Trunk Control)	Communicate thru Primary Storage	Primary Storage, Secondary Storage, 3rd Storage	29
Gamma 60	(a) Arithmetic Unit (b) General Comparator (c) Logical Unit	Program Distributor Data Distributor (parts of the central program & Coord-Unit)	Transcoder	Data Distributing Channel & Data Collection Channel Central Program & Coordination Unit	Central Store	7

1.2.2 *Growth of Multiprocessing* (Table 1.2.2)

By definition, the Princeton machine designed by Burks, Goldstine and von Neumann (2) in 1946, will be called a "conventional processor" or uni-processor. This was a parallel machine, with a hierarchy of memories which could be accessed sequentially.

In the IBM 701, the input-output equipment was controlled directly by the processor. All timing of tape gap times, card feed delays, etc., was controlled by the computer.

Data Channels (I/O Channels)

Data channels were a considerable improvement. As described in section 3.2, they made possible the simultaneous operation of peripheral equipment and the central processor.

Separate I/O Processors

Next on the trail to multiprocessing is the use of a completely separate Input-Output Processor with its own memory. Noteworthy among those in daily use is the IBM 1401 which is used with a high percentage of the 7090–94 installations. Many of these are used as off-line computers with the only means of communication a reel of magnetic tape. Others are directly cable-connected and provide editing, data conversion, peripheral control and communication control functions.

Multiple Computers

Multiprocessing has come to fruition with such systems as the multiple computer CDC-3600 (3) and D–825 systems (4). These systems were designed as multiprocessors and have the flexible coupling and control provisions necessary (sections 3.0, 4.0, 5.0).

Possible Future Steps

The next step may be in either of two directions or a combination of the two. Networks of processors all controlled by the same control unit have been proposed and partially designed. Solomon (5) and the Holland (6) iterative network are examples. They appear to have advantages in large matrix or relaxation problems where many computations can be carried on in parallel. As many as 2000 parallel processors have been proposed.

Another possibility is an extension of the modular unit approach of the Gamma 60 (7) to a multiprocessor system in which specialized Add-Compare Units, Multiply Divide Units, Edit Units, Logical Operation Units, Shift Units, etc., would efficiently perform one service. Problems of loading and scheduling are critical in the success of such a system. A very large number of problems is required to produce a good statistical mix so units can be efficiently used.

2.0 GOALS OF MULTIPROGRAMMING AND MULTIPROCESSING

There are two competing trends in modern data processing, the trend toward large, complex, centralized systems and the opposite trend toward small, simpler decentralized systems. Advocates of the centralized systems point to the growing need for communication between computers and the resulting ability to gather large quantities of data at one place. Then, they argue, the most efficient, most reliable, most flexible way to handle this large mass of data is by multiprogramming and multiprocessing (8, 9, 10). Decentralization advocates point to the convenience of small computers and argue that a simple computer can do a simple task more economically. Furthermore, many businesses like to control their own data and will pay a small increased cost for this privilege if necessary (11).

Multiprocessing systems emphasize the characteristics of reliability, efficiency, flexibility and capability to differing extent depending on the application.

A spare processor is used to provide increased reliability in military command and control applications and in the SABER commercial airline reservation system. The additional processor was used as a standby only in case of failure. More recent systems obtain increased efficiency and capability by coupling processors through disk files (12, 13) and also thru switching centrals (14).

One important recent activity is the development of systems with multiple remote terminals, each "time-sharing" the centralized system (10, 15). These systems assume the existence of multiprogramming so that each terminal may operate as though the others do not exist and it alone controls the computer.

In the following paragraphs some of the advantages of the large multiprocessing and multiprogramming system are described and evaluated. Note, however, that multiprogramming ability on small systems is also being actively considered (16).

2.1 INCREASED EFFICIENCY

Fuller use of equipment can be brought about by "time-sharing" and "space-sharing." Processors, switching equipment, input-output controls and memory address registers may be time-shared. Core memories, disk files, and to some extent, magnetic tapes and printers may be space-shared.

2.2 INCREASED RELIABILITY

Multiprocessing systems provide increased reliability by: sharing of duplicate equipment, automatic switchover and recovery facilities, built-in error detection and correction, prevention of error by automatic supervisory control and performance monitoring, the use of diagnostic programs to catch marginal conditions, and improved maintenance facilities on the computer site. Many of these capabilities are available in uniprocessing systems but receive added emphasis in multiprocessing systems. In general, the large volume of operation on multiprocessing systems makes increased reliability necessary but also provides economic feasibility by sharing costs over many tasks.

2.2.1 *Sharing of Duplicate Equipment*

An efficient centralized system can use duplicates of each item of equipment to share the total operational load. Good design consists of balancing the system so essential peak activities can be handled even if some part of the system fails. In normal operation the additional capacity can be fully absorbed doing routine work. If necessary, additional work load can be brought in on communication lines to keep the system fully loaded.

2.2.2 *Automatic Supervisory Control*

Many programming errors can be prevented or made harmless by a good supervisory program. A major source of programming errors is I/O handling. As described in section 4.0, the programmer's task is simplified on I/O since the details are handled by the supervisory program. In addition, program loops, memory address errors, and other program errors are prevented from tying up the system.

2.2.3 *Automatic Switchover and Recovery Facilities*

A price paid for multiprocessing and multiprogramming is the complexity of the system. The Executive, Scheduler and other control programs are difficult to write. When an error occurs, it may affect several programs. (It is possible for a disk head to drop down and mangle the data on many disk tracks, for example.)

It is essential to provide backup and recovery capability in the system. In case of subsystem failure, it is relatively easy to provide for switching in another subsystem.

2.3 INCREASED CAPABILITY

Some tasks require high speed processors, large memory capacity, many tape units, large disk files or multiple path communications. These tasks cannot be done effectively on small systems. Sorting of large files, design automation programs, and linear programming of inventory problems are three types of business problems where large memory (16 to 32K) and many I/O units are required.

Ward (18) argues that faster computation is more important than better organization or parallel computation. He mentions several problems requiring speed increases 100 times as great as present computers such as: ballistic missile and satellite launch, neutron diffusion problem in reactors (50^3 grid points), Monte Carlo problems and the weather research problem (10^5 grid points). It appears also that large, fast memories would be required for these applications.

Faster computation or increased thruput capacity is possible on some problems by using multiple processors to run the same program. M. Conway's paper (this issue) illustrates this capability.

Multiprogramming and multiprocessing make economically feasible three new capabilities which are expected to greatly extend the usefulness of computers.

2.3.1 *On-Line Debugging*

Since many programs may be in the system at one time, it is possible for programmers to enjoy the luxury of console debugging without slowing the processor appreciably. Instead of time-consuming memory dumps or traces, the programmer can guide his program from error to error, correcting as he goes. Program debugging has been reported to take up to 32% of processor time, so this capability is of considerable value.

2.3.2 *Man-Machine Interaction*

Many engineers who use computers will welcome the ability to get only the data they need without the necessity for requesting in advance that all possible permutations of the data be calculated. Guidance from a knowledgeable scientist or engineer is made possible by a time-sharing console. Singularities or unreal trends in the course of computation can be halted quickly so that errors do not result in a pile of useless paper.

More significant is the use of control and display consoles to allow the computer to assist in the design process. Calculation, data storage and display reduce the routine work of design so that greater creativity can occur (19).

2.3.3 *Remote Operation and Communication*

The high reliability and great flexibility of a multicomputer system means that it can perform useful services to a wide group of users on a time-shared subscriber basis.

Small companies or even individuals may have a typewriter-like keyboard-printer available to handle all business transactions.

Airline reservations systems are well known. Not so well known are the integrated business systems which provide a network of communication lines to control and record sales and inventory transactions. Direct communication from the sales office to the warehouse provides for ordering, billing, inventory control and warehouse picking operations, all under computer control. The delay due to mailing of orders, receipts and bills is avoided and accurate records are kept of all transactions.

2.4 SIMPLIFIED OBJECT PROGRAMMING

Control of a multiprogramming, multiprocessing system requires a complex supervisory program (section 4.2), which is difficult and expensive to prepare. (Similar programs are required for uniprocessor systems, but are much simpler.)

In return for this complexity, which is largely assumed by the system manufacturer, the programmer's tasks become simpler in the following ways:

1. Input-output control is handled by the supervisory programs so all problems of timing, interaction, assignment and error control are eliminated from the object programmer's responsibility.
2. Addressing can be symbolic for memory and peripheral equipment.
3. If "page turning" capability is available, the programmer can write programs as though a large core memory is available rather than being restricted to small memories.
4. A large library of specialized routines can be called upon by the programmer to do specific tasks. These subroutines can be written by specialists so they perform efficiently. Calling routines for these programs are simplified because of assistance from supervisory programs.
5. Increased specialization of programmers becomes possible so that each programmer need not know all the formidable array of techniques and methods now available. Simplifications of the programmer's tasks are increasingly important as more computers are installed and the need for programmers increases.

3.0 SYSTEM CONTROL—REQUIREMENTS AND EVALUATION OF TECHNIQUES

3.1 CENTRAL SWITCHING OR MEMORY ACCESS CONTROL (Table 3.1)

Requests for memory access can come from two to five processors, Input-Output exchanges,

Table 3.1 Memory Access Control

Technique Example	*Hardware Cost*	*Example*
1. *Crosspoint Switch* Switching interlock is provided by a crosspoint switch matrix and a bus allocator to resolve time conflicts. Queued in priority order.	Crosspoint matrix (4 computer modules to 16 possible memories). Full crosspoint would require an estimated minimum of 300,000 switch points.	D-825
2. *Multiple Bus Connected* Five-way switch built into storage module. (Processors can transmit address to storage module and request service on a first come, first served basis.)	5 sets of address registers, gates and line drivers.	CDC-3600
3. *Time-Shared Bus* Bus control unit receives memory requests from Lookahead unit. Handles requests in priority order based on availability. Bus is time-shared on 0.2 μsec. cycle. Control decisions overlap address transmission to memory units.	Lookahead unit has 4 sets of address and operand registers and 5 control counters.	STRETCH IBM 7030

Data Channels, and in some cases, directly from peripheral equipment. Many of these requests are urgent and must be handled on a priority basis.

3.1.1 *Crossbar Switch*

The crossbar switch or cross point matrix provides multiple-wire paths from M requesting modules to N accepting modules. Each path may be on the order of 50 to 100 lines wide in order to carry full memory words (36 to 72 lines), memory addresses (12-16 lines) and control signals (6 to 20 lines). Sometimes cables are unidirectional so that another set of 50 to 100 lines is required in the opposite direction.

Crossbar switches were first developed for telephone switching and were electromechanical. The crosspoint matrix switches used in computers have been transfluxor magnetic cores (RW-400) or diode AND gates. In the D-825 (4) (Figure 3.1.1), provisions are made for modular addition of switching matrices and associated controls so that a maximum of 4 computer modules can access 16 possible memories. A minimum of 300,000 switching points would be required for a complete system so that the cost approaches or exceeds the cost of a large computer module.

Tremendous flexibility is obtained in a crosspoint switch since any processor can connect to any memory in a fraction of a microsecond. Also, there are numerous ways to provide a function in case of failure in any part of the system. Duplication of the crosspoint matrix is required, however, in a system requiring maximum reliability.

3.1.2 *Multiple-Bus Connected*

A lower cost system than the crosspoint matrix is provided by use of separate busses connecting a processor (or input-output channel) to one or more specific memories (Figure 3.1.2). The saving is due to the reduction in the number of switch points. Each computational module may have a direct connection to private storage in addition to sharing common storage. This technique is less flexible than the crosspoint matrix but may be completely adequate in a system designed for a specific range of applications. If connections are easily changed physically, it is much less expensive to set up new paths by plugging rather than by switching.

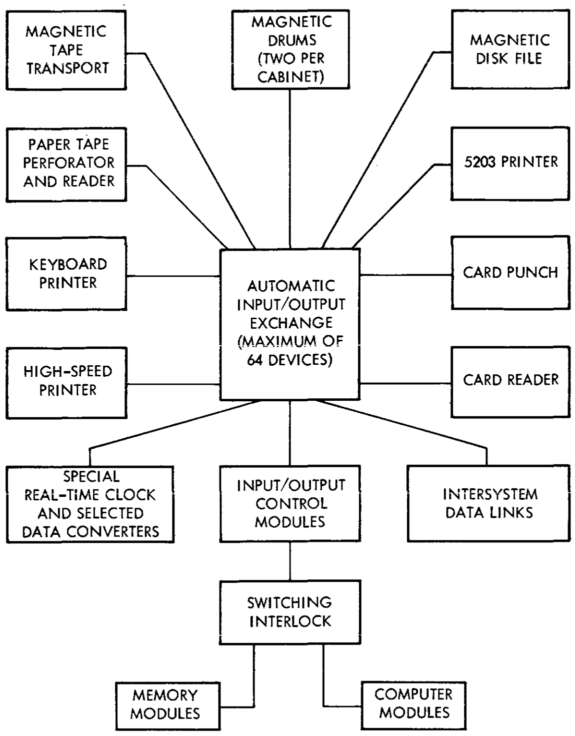

MAGNETIC TAPE TRANSPORT
MAGNETIC DRUMS (TWO PER CABINET)
MAGNETIC DISK FILE
PAPER TAPE PERFORATOR AND READER
5203 PRINTER
KEYBOARD PRINTER
AUTOMATIC INPUT/OUTPUT EXCHANGE (MAXIMUM OF 64 DEVICES)
CARD PUNCH
HIGH-SPEED PRINTER
CARD READER
SPECIAL REAL-TIME CLOCK AND SELECTED DATA CONVERTERS
INPUT/OUTPUT CONTROL MODULES
INTERSYSTEM DATA LINKS
SWITCHING INTERLOCK
MEMORY MODULES
COMPUTER MODULES

Figure 3.1.1. D825—A Multiple-Computer System for Command and Control.

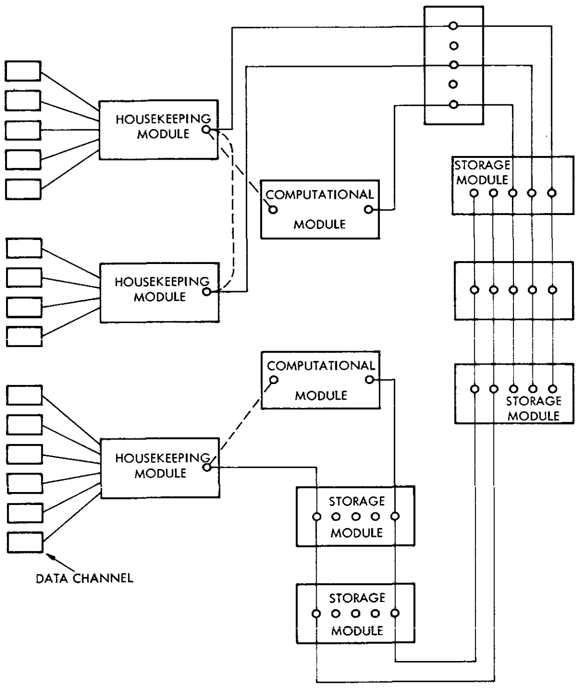


Figure 3.1.2. Two-computer System with Private and Common Storage (CDC–3600).

3.1.3 *Time-Shared Bus*

The lowest cost switching system, Figure 3.1.3, takes advantage of the availability of memory registers in each processor and each memory module to allow the bus system to be time-shared. Instead of connecting a processor and memory continuously, they are connected for only the time required to transfer information. This technique is especially useful if memory accesses can be pre-planned such as in sequential instruction fetches and data fetches. More than one channel can be used if the number of accesses required becomes large enough to slow down the total access time. Multiple bus channel control, priority switching requirements and the need for two-way transmission add to control complexity. A system of this type was used on the IBM 7030 (STRETCH).

3.2 I/O SWITCHING AND CONTROL

In the early days of computers, the computer operated peripheral equipment directly. If a tape was to be read or written, all other activity was stopped while the computer controlled the transfer of information between the tape and core memory. Memory addresses were prepared, timing of tape gaps was calculated and the remainder of the system sat idle.

3.2.1 *Input-Output Channels* (Data Channels)

Input-output channels were a great improvement. Each channel had direct access to mem-

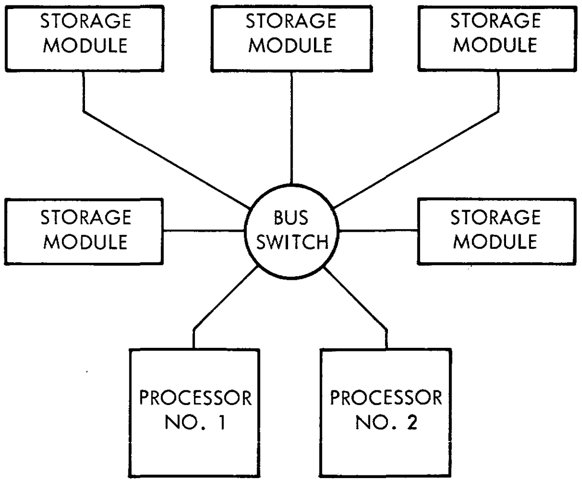


Figure 3.1.3. Time-Shared Bus Assignment.

ory, with its own address register and the ability to keep a count of the number of records transferred. Even these channels sometimes used the address preparation and arithmetic capabilities of the main computer. Also, since they shared the main memory, an interrupt system was required to insure priority of access to memory for peripheral information.

This simple interruption to enter data into memory is quite efficient since the I/O channel "steals" only a memory cycle as needed where the complex program interrupt requires storage and retrieval of arithmetic registers, etc.

3.2.2 *Asynchronous Input-Output Requirements*

Multiprogrammed and multiprocessor systems must operate in an uncontrollable environment, accepting information from many sources simultaneously, processing it and dispatching the processed information to many points.

Earlier systems attempted to provide synchronous switching systems to cope with these problems but had no way to handle the frequent, probabilistic stacking up of control or information requests. It was found necessary to provide for queuing of requests and buffering of information flow.

3.2.3 *Control Word Philosophy*

The STRETCH exchange has 256 words of core storage to provide an essential control function in a complex system (20). Instead of providing hardware registers to store addresses and counts for the control of peripheral channels, the control words are stored in a fast (one microsecond access) core memory and one set of hardware registers are time-shared in a rapid, asynchronous sequence. When a memory access request is made, the required control words are pulled from memory to the control registers and used to set up the necessary switching paths. These control words are then updated by adding one to the address, deducting one from the word count, modifying status conditions, and replaced in memory.

3.2.4. *Queuing of I/O Requests*

Systems loading is controllable to some extent by refusing to start new tasks until previous tasks are completed.

Three types of queues are maintained in a multiprocessing, multiprogrammed system:

1. New tasks not yet started.
2. Tasks partially completed, awaiting completion of a specific peripheral operation.
3. Tasks being run on one of the system processors.

In addition, there may be "standby" tasks such as diagnostics, program check runs, or billing runs which can be pulled in whenever processor loading permits.

Queuing is controlled by an operating system program called the Peripheral Control Program, Input-Output Supervisor or a similar title (section 4.2.4). These programs maintain peripheral control tables containing essential information about each program and each piece of equipment.

3.2.5 *I/O Processors*

The STRETCH exchange was really a small separate input-output processor with memory and limited instruction capability. Many variations on this approach have been tried.

In the CDC-3600, a separate housekeeping module is provided which handles all input-output functions including a number of data channels. The CDC-3600 can also use a CDC-160 as a direct on-line processor handling peripheral equipment.

An I/O Control Module on the Burroughs D-825 is connected to an automatic I/O exchange to provide control signals, parity checks, timing interface and data conversion. It contains a separate instruction register, decoding circuitry, data register and a data manipulation register.

It is also possible to come full circle back to using the main computer as an I/O processor. In a multiprogrammed system with powerful interrupt and sufficiently rapid storage and retrieval of status information, time-sharing of the central processor becomes feasible. When all registers are in thin-film memory, for example, program interruption can be accomplished by merely changing the program counter to a new address so that no time penalty is paid for an interrupt.

3.3 PRIORITY AND INTERRUPT CONTROL

Multiprogramming and multiprocessing cannot be done effectively without the ability to establish priority between programs and to interrupt operations when events of higher priority demand attention.

Older computers ran one program until it was completed, performing tests for only those occurrences which the programmer could anticipate. Since testing was costly in processor time it was done initially only to detect such items as overflows or underflows in arithmetic or to determine whether a peripheral had completed an assigned task.

Later computers had the ability to "trap" and react to an unusual occurrence by passing program control to a specified location in memory. This testing was done by separate hardware in parallel with processor operation and did not delay the object program being run unless the trapping operation actually occurred. After trapping, the program was required to search a register to find out the cause of interference and then jump to a new program to take action.

This slow procedure was adequate for uniprogramming systems. In a complex system, searching of a number of interrupts is too time-consuming, so a better way was sought. Present systems provide multi-level interrupt ability so that an interrupt causes direct transfer of control to the location of the program which is to handle the interrupt.

Several types of interrupt may be provided. Some types are:

1. I/O interrupts; at the completion of an assigned task by a peripheral, arrival of an I/O request or perhaps from an operator at a console.
2. Program interrupts may occur due to arithmetic overflow, the periodic signal from an elapsed time clock or an interrupt instruction in the program itself.
3. Malfunction interrupts are those from an I/O malfunction such as a broken tape, card jam, or parity error, or major equipment malfunctions such as memory parity, error or failure of a subsystem to respond when interrogated.

Each interrupt must be accepted and eventually handled. If too many control interrupts come in and some are lost there is loss of input or output information. To prevent this, interrupts must be handled rapidly with the highest priority items handled first. Queues of interrupts are built up on each requested facility under the control of a supervisory program in the operating system.

The requirements of service are taken into account in assigning priority levels. Highest priority must go to serious malfunctions such as power failure, next to error causing malfunctions such as memory parity error, then to peripherals which must be serviced within a limited time, and finally to requests from the processor itself.

Processors can always wait for memory access since no information is lost. However, no designer feels happy about forcing a high speed processor to be idle.

During an I/O interrupt where a separate data channel to memory is available, the processor is not affected except in being denied access to memory.

During a processor interrupt it is necessary to perform several operations in a short time. These functions, accomplished partially by hardware and partially by program are:

1. Prevent additional interrupts at this priority level and lower priority levels.
2. Store present contents of all registers affected by this interrupt class. (This can be done with a SAVE instruction which transfers register contents to memory.)
3. Determine interrupting channel and device. (Automatic in a well designed system.)
4. Determine cause of interrupt. (Encoded on interrupt lines in some systems, program scan required in others.)
5. Determine required action. (May be prestored in memory location addressed by interrupt.)

6. Determine urgency of interrupt, assign priority and set task in a processing queue. (Although interrupt is high priority, action may be data dependent so that delay is tolerable.)
7. Perform action required by interrupt.
8. Test for additional interrupts at this and lower priorities and handle if they exist. (Higher priority interrupts would have caused storage of information and queuing of this interrupt request.)
9. Restore register contents and continue interrupted program.

Interrupt routines are part of the Executive program. Unless the proper hardware capabilities are available, the Executive program becomes complicated and unwieldy.

In a well designed system with several processors, the total supervisory control system can be as low as 5000 words and still perform all essential functions, since hardware performs many of the time and memory consuming operations. Such systems make multiprogramming and multiprocessing feasible (9).

4.0 SYSTEM DESIGN AND OPERATION

In order to meet the goals described in section 2.0 an integration of hardware, software and application knowledge is essential in order to analyze the trade-offs and compromises to be made.

4.1 SYSTEM PLANNING AND SIMULATION

In system planning, a set of success criteria is required. For some types of scientific work the utmost in speed and capacity may be the goal.

Performance/Cost Ratio

For the multiprocessing system the goals are efficiency, reliability, and capability. The final measure is the ratio of performance to cost.

As an example of a method of calculation, assume the multiprogramming load of section 4.2.2 is typical. In addition, use some *estimated* figures for the ratios of input-output processor cost, tape system costs and central switching costs so that a set of cost figures are obtained as follows. (In the dual system a central exchange has been added to connect processors to memory.)

U = Uni-Processor System	
1. Processor (1)	2.04
2. Memory (1)	1.86
3. Tape Control (1) (Handle 32 tapes)	1.0
4. Tape Units (12) (Each 0.2)	2.4
5. ------------------------------	
Cost	7.30

D = Dual Processor System	
Processor (2)	4.08
Memory (1)	1.86
Tape Control (1)	1.0
Tape Units (20)	4.0
Central Exchange (1)	0.5
(Each .5 × 10^6 X)	11.44

$$\text{Cost ratio} = \frac{D}{U} = \frac{11.44}{7.30} = 1.57$$

Performance can be calculated on the basis of equipment usage of the two systems at the costs estimated, noting that the multiprocessor system is doing, in addition, programs Number 2 and 3 (of 4.2.2).

Uniprocessor System (Programs #1, #5, #6)	
1. Processor 90% × 2.04 =	1.84
2. Memory $\frac{18}{32}$ × 1.86 =	1.05
3. Tape Control $\frac{12}{32}$ × 1.0 =	.38
4. Tape Units 100% × 2.4 =	2.40
Performance	5.67

$$\text{Uni-System Cost/Performance} = \frac{5.67}{7.30} = 77.8\%$$

Dual-Processor System (Programs #1, #5, #6, #2, #3)	
1. Processors $\frac{160}{200}$ × 4.08 =	3.27

2. Memory $\frac{28}{32} \times 1.86 =$ 1.63

3. Tape Control $\frac{20}{32} \times 1.0 =$ .63

4. Tape Units 100% $\times$ 4.0 = 4.00

5. Central Exchange $\frac{160}{200} \times .5 =$ .40

Performance 9.93

Performance Ratio $\frac{D}{U} = 1.11$

Dual System Cost/Performance =

$$\frac{9.93}{11.44} = 86.5\%$$

Even though an expensive processor and central exchange was added, an increase in performance was obtained in the dual system. It does not seem worthwhile, however, to add enough additional equipment to handle Program #4, although with a 16K memory instead of a 32K memory it may be close.

While hand calculations of this kind are instructive it is necessary to consider many more factors in more complex ways in a real system. Analytic methods have not been satisfactory so simulation methods have been used extensively.

Simulation

Smith (21) describes the use of the General Purpose Systems Simulator (GPSS) developed by Gordon (22) which was used to analyze a multiprocessing, multiprogramming system.

Some results of simulation described by Smith (21) show 11% increase in thruput of four 7090's connected in a multiprocessor configuration compared to four separate 7090's.

Simulation can be applied to any part of the system depending on need. It must be used with caution, however, since the results are only as good as the model.

4.2 OPERATING SYSTEM OR SUPERVISORY CONTROL (Multiprogrammed System)

Control of a large flexible system is provided by a Supervisory Control program or Operating System which resides permanently in a protected area of memory. Initially, magnetic core memory may be used but there is a trend toward fixed memory (or "read only" memory). The hierarchy of control is shown in Figure 4.2.1, while detailed sequencing of control is shown in Figure 1.2.1.

4.2.1 *Executive*

After initial loading of programs by the Loader routine, control of the system is turned over to the Executive which assigns tasks to other routines and monitors system performance. Errors of all types cause program interruption to the error routines controlled by the Executive. In addition, the Executive handles interrupts of all kinds.

Each object program requests attention from the Executive and is assigned a number and a priority based on its intrinsic priority or required completion time. It is then turned over to the Scheduler for assignment of facilities.

4.2.2 *Scheduler*

The Scheduler maintains a list of system facilities and the programs currently queued on those facilities. Each incoming program is provided with a header which contains such information as: Memory space required, number of tapes required, output requirements, estimated running time and estimated completion time.

The Scheduler examines the program requirements, checks availability of peripherals and memory and determines whether the program should start immediately or be queued awaiting some facility. Queued programs are rechecked whenever a previous program com-

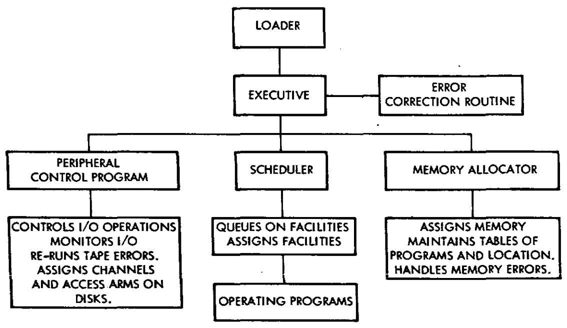

LOADER
EXECUTIVE
ERROR CORRECTION ROUTINE
PERIPHERAL CONTROL PROGRAM
SCHEDULER
MEMORY ALLOCATOR
CONTROLS I/O OPERATIONS MONITORS I/O RE-RUNS TAPE ERRORS. ASSIGNS CHANNELS AND ACCESS ARMS ON DISKS.
QUEUES ON FACILITIES ASSIGNS FACILITIES
OPERATING PROGRAMS
ASSIGNS MEMORY MAINTAINS TABLES OF PROGRAMS AND LOCATION. HANDLES MEMORY ERRORS.

Figure 4.2.1. Hierachy of Control.

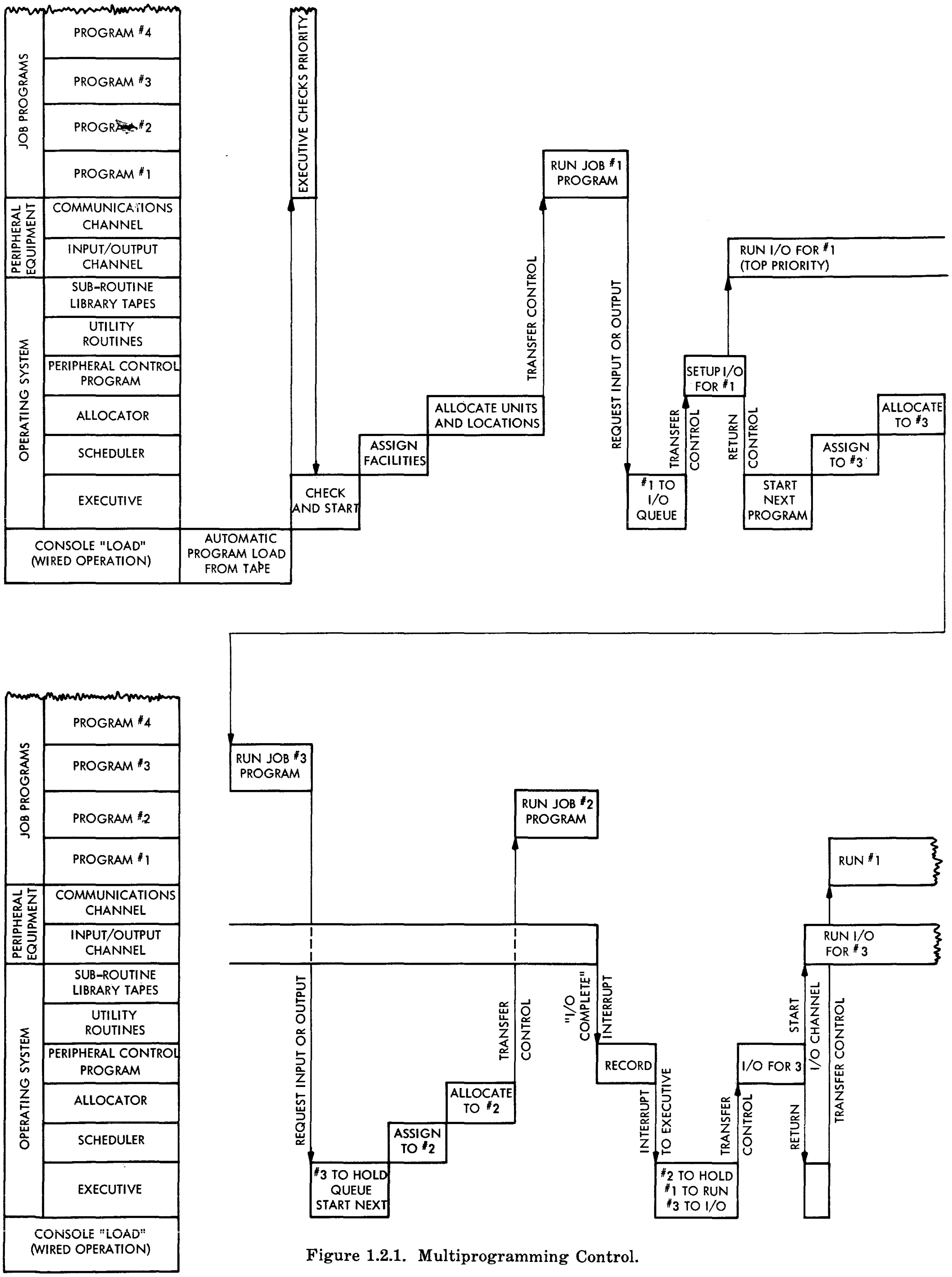


Figure 1.2.1. Multiprogramming Control.

pletes an operation and a facility (memory, peripheral, or processor) becomes available. An important part of the systems programmer's design task is to minimize the Scheduler program's operating time, while maximizing system efficiency.

When the Scheduler determines that a program can be run and has assigned facilities, it sets up an account for billing the program. Actual running time on various facilities is recorded as determined by readings from a Real-Time Clock. It then turns the program over to the Allocator for assignment of actual units and memory locations. (See Figure 1.2.1.) This operation can be followed in the associated tables (Table 4.2.2).

The Scheduler gathers information from headers of all programs currently in memory as assigned by the Executive. It compiles the program requirements table from this data and maintains a status report. After each program assignment or interrupt which changes status the assignment symbol is updated.

Table 4.2.2(a) *Program Requirements Table*

Schedular Action			*Program Parameters*			
Step	*Assigned*	*Priority*	*Prog.*	*Memory*	*Processor*	*Tapes*
2	1	1	#1	4K	40%	2
—	0	2	#2	8K	60%	6
—	0	3	#3	2K	10%	2
—	0	4	#4	16K	50%	4
3	1	3	#5	4K	20%	4
1	1	1	#6	6K	30%	8

Initially, the Scheduler scans the Program Requirements Table to determine the number of programs of highest priority, Priority 1. Initially, it finds two P1 programs, #1 and #6, so it must make a further comparison.

Priority Program requiring most tapes is assigned first because tapes are least flexible resource, largest memory is assigned next because small memory blocks are easier to obtain than large blocks if otherwise equal the program with lowest number is selected. General rule is to do most difficult task first. Therefore,

Table 4.2.2 (b, c, d) *Facility Load and Allocation Table*

(b) Memory—Total 32K Assignment Table | (c) Processor Load Table—% | (d) Tape Assignment Table

Prog.	*Amount*	*Location*	*Prog.*	*Amount*	*Program*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*
Oper.	4K	0000-4096															
1	4K	10240-14336	1	40	1							0	0				
2	—	—	2		2												
3	—	—	3		3												
4	—	—	4		4												
5	4K	14336-18432	5	20	5									⊕	⊕	⊕	⊕
6	6K	4097-10240	6	30	6	x	x	x	x	x	x						
Avail.	*		Avail.	*													

Step Sequence—Scheduler

After Step #1	22K	Available	Avail.	70	– – – – – –
After Step #2	18K	"	"	30	– – – –
After Step #3	14K	"	"	10	None

* The availability number is changed by the Scheduler program. It is shown here as a series of steps for expository purposes.

Prog. #6 is assigned tapes 1 through 6, 6K of memory and 30% of processor time.

Other Scheduling Algorithms

1. Corbato (10) assigns each user program as it enters the system to a multi-level priority queue.
2. A simple scheduling rule is to run each of N programs for a fraction 1/N of the total available time and in a fixed sequential order. This has the virtue of reducing supervisory program requirements to handling I/O only. Also, it can be quite efficient since "lookahead" is possible so that the next program to be run can be brought in while the previous program is being run. This lookahead requires a slight increase in supervisory time, additional memory capacity, and the availability of separate input-output channels.

4.2.3 *Allocator*

Allocation of memory space, overlay of programs and calculation of assignment algorithms is handled by the memory allocator. In a system like ATLAS, it automatically stores and retrieves "pages" of data from drums. It also provides the base addresses to provide for memory relocation, so that the programmer may write all programs in symbolic form. After completion of memory allocation the object program is assigned to a processor and starts its run. (See Communications of the ACM, Storage Allocation Issue, October, 1961.)

4.2.4 *Peripheral Control Program (PCP)*

When an object program requests a peripheral device it does so symbolically in some systems. Actual assignment of peripherals and maintenance of assignment tables is done by the Peripheral Control Program. An object programmer may call for Tape 1 or Tape "C" as he wishes, the Peripheral Control Program (PCP) will assign an available tape, say actual Tape 12, and maintain a table of such assignments.

All actual Input-Output is controlled by the PCP so that the programmer need not know even the speed of the peripheral equipment. Simple commands are sufficient although optimizing of operation is always assisted by programmer understanding.

The Peripheral Control Program also prints out instructions to the operator, usually on a console typewriter, so that he can mount tapes, feed cards, change printing forms, or perform other accessory and essential tasks.

4.3 OBJECT PROGRAMS

Object programs for use in multiprocessing systems need not be affected unless there is a need for greater speed than one processor can provide. Higher speed can be obtained by processing different parts of the same program simultaneously on different processors. This is only feasible when programs can be segmented into blocks of reasonable size.

Segmentation

Segmenting of programs can occur when there is no sequential relationship between two parts of a program so that results obtained in one part of the program are not required in the part which is to be processed simultaneously. Indicators are required in the program to identify the "Forks," where the program may be separated and processed in separate segments, and "Joins" before the segments where the program must be processed sequentially.

"Forks" and "Joins"

Insertion of "Forks" and "Joins" is the programmer's responsibility. Control of processing of the segments is done by the Executive program.

An intentional interrupt or branch to the Executive program, which includes the address of a control routine, provides a satisfactory way to implement the Forking and Joining of segments.

Care must be exercised in programming to insure that segments exceed a minimum length so that the Executive program time required per segment is small compared to segment length. Interrupt control, register saving, priority checking and facility assignment are required for each segment started. These can easily require 10 or more memory cycles so that segments less than 100 memory cycles in length are apt to prove uneconomical.

Hardware facilities for performing all of these functions can be provided if speed is essential and cost is secondary.

5.0 HARDWARE AIDS TO MULTIPROGRAMMING AND MULTIPROCESSING

Some essential abilities are required in system hardware to make multiprogramming and multiprocessing reliable and efficient. Many programs are being run simultaneously and are time-sharing the capabilities of several processors, memories, input-output channels and peripheral equipment.

5.1 MEMORY PROTECTION

When many programs are operating in the system at the same time and more than one processor is accessing each of several memory banks, it is essential to have an effective means to prevent memory addressing errors. Some of the ways which have been used are:

1. Upper and Lower Limit Registers

   The IBM STRETCH used limit registers which were set by the supervisory program to bound the memory area allotted to an object program. Hardware comparison of each address to these registers was automatically and simultaneously made for each program step requiring memory access.

   The RCA 601 used the same technique but provided also the ability to add the lower limit register to the address provided in the object program. This made possible the relocation of programs in memory at run time.

2. Mask Registers

   Another technique is the use of a "mask" register which contains a bit for each memory block of given size. This technique requires less hardware since only one bit is needed per memory block. Also, the program control is somewhat simpler. In operation the instruction address is decoded to select a particular line which is compared against the corresponding bit in the mask register. If this bit has been set to "1" the program may use that block of memory, otherwise it may not. Several blocks of memory may be allocated to a program by proper assignment of the mask bits. The memory allocator program must maintain a list of memory blocks and the program to which assigned.

3. Hardware Lockout

   In the ATLAS computer (8), the object programs operate in a different mode than the supervisory programs, so cannot generate addresses which would address a restricted area of memory. If, due to error, a bit is generated which would cause the supervisory memory to be addressed an interrupt occurs to an error routine. This insures excellent protection for the vital supervisory programs.

   Other hardware schemes can be envisaged which are variants of the above and are of value in particular situations.

4. Fixed Memory (or "Read Only" Memory)

   As a further protection against error and to speed up operations, there are some fixed memories in use.

   These are usually deposited capacitors or inductive arrays (23) which are set at the factory. They can be altered by hand punching or wiring in the field but cannot be altered by the computer. Compactness and low power requirements enable high switching speeds to be attained. When used for supervisory programs they provide the best possible protection. Alterable memory must still be used, of course, for tables and lists maintained by the supervisory programs.

5. Program Protection

   It is possible to provide a check routine in the program which compares each memory address to an upper and lower bound, or otherwise checks the validity of the address. The operational delay is intolerable in most situations. Sometimes this type of check is used only for undebugged programs where the delay can be accepted.

5.2 MEMORY RELOCATION

Some type of address conversion is desirable so that programs may be written independently of each other without regard to the space they will occupy in memory.

Actual location of data or programs in a desired address can be done: When the program is written, at assembly time when the program is assembled from the separate routines into a complete program, at run time, when the program is initially read into the memory, and at execution time. (This requires repeated relocation of the same program data and instructions.)

Another kind of problem exists when it is desired to run assembled programs on a multiprocessing system where the processors share several banks of memory. In this case neither the programmer nor the assembly program is able to determine in advance where in memory this object program will be stored. These cases are handled by providing in the multiprocessing system an executive program which contains a memory allocation subroutine. The memory allocation subroutine (see 4.2.3) maintains a list of the programs in the processor complex and assigns memories to them from its reservoir of available memory.

5.2.1 *Base Registers*

Hardware registers have been used in some machines to simplify the memory relocation problem. For example, all routines can be written relative to a base of zero and then at assembly time the assembler or allocator can assign to a register the starting location (base address) in memory. As each command is assembled, the base register is added to the command address to provide the actual memory location at which this address is to be stored. This sounds simple, but it is complicated by the necessity for providing for different types of commands.

The address portion of a command may refer to at least 7 types of addresses as described in References 24 and 25. These address types must be considered individually when memory relocation occurs since some are not affected, others may require addition to base addresses, while some may require complex analysis.

In addition to providing base registers, a suitable processor also provides hardware means to detect the presence of different types of addresses and handle them accordingly.

One of the simpler techniques is the use of a "relocation bit" in the instruction which signals when addition of the base address is required. Any instructions without this bit are analyzed to determine what other changes are required, if any.

It would be most efficient if each program were fitted next to the preceding one so that no one intervening memory space existed, however, this is difficult for the program to do without the use of separate registers for each program, which can become quite expensive. The usual solution is to assign memory in blocks in some regularized way. This requires that a table be maintained in memory of all programs and their memory assignment.

5.2.2 *Page Turning*

The ATLAS system has provided hardware for an interesting technique known as "page turning" (26). A "page" is equivalent to 512 memory words and may be stored on drums, disks or in core storage.

There are 32 blocks of memory pages of 512 words each in the ATLAS memory. A "page address register" is associated with each page and contains the most significant bits of the memory address of the page. Hardware is provided to compare a requested address with all the page address registers in parallel. If a page is in core memory it is automatically selected, if not a "non-equivalence" interruption is made to the supervisory program which then goes to the appropriate drum to pick up the block of memory required.

The advantages of this technique are:

1. Absolute memory protection is provided.
2. Object programs may be written from zero as a base with program relocation automatically handled.
3. Programs may be written which exceed core memory capacity, up to drum capacity, without segmenting.
4. Memory allocation is simplified for the executive or supervisory program.

The cost is high; 32 registers are required, each capable of addressing 32 memory blocks. However, with the decreasing cost of registers this technique probably will be adopted further.

5.3 MEMORY HIERARCHY

Memory size and speed are directly related, as discussed by Rajchman (17). Small memories 500 to 1000 words are commercially available with cycle times less than 300 nanoseconds. Equally well designed 32,000 word memories have cycle times on the order of 1.5 microseconds.

Many systems now are using the faster memories to store index registers, base registers and sometimes the whole complement of processor registers.

5.4 MODE CONTROL

The operating system or executive control programs must have access to the full capability of the machine. Other programs and operations need not. As a result, newer systems are incorporating multi-level "mode control" to insure isolation of operating functions by hardware means.

The simplest type of mode control is the provision of a bit in the instruction which identifies the instruction as an "object" program instruction (LOAD, ADD, STORE, etc.) or a "supervisory" instruction (READ TAPE, ALERT DATA CHANNEL, CHANGE PRIORITY, etc.).

Mode control protects the system from object program errors. It also insures that processor time will be available periodically to handle high priority interrupts.

Operating system programs are normally maintained in a separate portion of memory which is protected by limit registers, hardware address control or other means.

In the "operating system" mode or "disabled" mode, object programs are not permitted to run. Accidental transfers of control cause interrupt to the operating system. Even in "normal" or "enabled" mode, the attempt of an object program to use a "supervisory" instruction causes an interrupt.

STRETCH (20) and ATLAS (8) have the most extensive mode control of present systems.

5.5 PRIORITY AND INTERRUPT HARDWARE

Priority control can be obtained by cascading circuits so that actuation of a string of logic any one breaks the path to those farther down the string. This technique provides for priority in physical order. The priority of a channel or data processor can only be changed by physically unplugging a cable and plugging it in at a different point.

Flexible control of priority is possible by use of priority registers in each unit. A processor requesting memory access would present to the memory access control a set of lines from its priority register, previously set by the supervisory program. The memory control would scan all sets of lines and allow memory access to the processor with the highest priority. This flexible control is expensive because of the additional registers, cabling, scan and control circuits required. In most business data processing systems it would not be justified.

Interrupt hardware can provide one level of interrupt or multi-level interrupt on interrupt as is provided in the STRETCH system (20).

ATLAS (8) has an Interrupt Flip-Flop which is triggered when any L.A.M. (look at me) signal occurs. No action is required if the Interrupt Flip-Flop is not set. However, when a L.A.M. has occurred, the next instruction in process is delayed and the L.A.M.'s are examined in groups by an interrupt program. An Interrupt Control Register (24 bits) is provided and used to read successively out of V-registers associated with the particular cause of interruption.

6.0 CONCLUSIONS—PROBLEMS AND GROWTH POTENTIAL OF MULTI-SYSTEM SYSTEMS

The growth of multiprogramming and multiprocessing has been traced from the first stored program machine (the Princeton or von Neumann machine) to the true multiprocessors of today controlled by a powerful and complex operating system program.

Major problems to be solved are: to provide backup and recovery capability in complex systems so that errors of any type do not cause catastrophic failure of the system, devise control programs and hardware to allow many programs to run concurrently and efficiently, reduce cost so that large centralized systems can continue to be competitive with small decentralized systems.

It is probable that large, centralized systems and small, decentralized systems will co-exist. When they become connected by communication lines, which appears inevitable, many new kinds of data processing and control become available. Like Robert Young's old railroad slogan "A pig can cross the U.S. without changing cars—why can't you?" the slogan for computing may be "Why mail your order when your computer will do it for you?" It seems clear that by 1970 the tremendous mass of paper work moving around the U.S. will be replaced by direct computer to computer communications of orders, bills, invoices, catalogs, quotations, etc., immediate handling of routine decisions and tremendous increase in the efficiency of business and industry.

In the same way, many scientific problems will be solved by time-sharing of the capabilities of large centralized computers.

7.0 REFERENCES

1. E. F. CODD, Multiprogramming Stretch: A Report on Trials—p. 574 Proc. of IFIP Congress 1962, Munich, Aug. 27 to Sept. 1, North Holland Publishing Co., Amsterdam.
2. A. W. BURKS, H. H. GOLDSTINE, and J. VON NEUMANN, Preliminary Discussion of the Logical Design of an Electronic Computing Instrument (reprinted) p. 24, Datamation, Sept. 1962.
3. C. T. CASALE, Planning the 3600, p. 73, Proceedings EJCC, December 1962. See also CDC-3600, Datamation, May 1962, p. 37–40.
4. J. P. ANDERSON, S. A. HOFFMAN, J. SHIFMAN, and R. J. WILLIAMS, p. 86, Proceedings FJCC, December 1962—A Multiple Computer System for Command and Control. See also D-825 Manual—Burroughs Corporation.
5. D. L. SLOTNICK, W. C. BORCK, and R. C. MCREYNOLDS, The Solomon Computer, Proc. FJCC, p. 97, v. 22, 1962 (AFIPS).
6. J. H. HOLLAND, On Iterative Circuit Computers Constructed of Microelectronic Components and Systems, p. 259, Proc. WJCC, May 1960.
7. P. DREYFUS, Programming Design Features of the Gamma 60 Computer, Proc. EJCC, December 1958.
8. T. KILBURN and R. B. PAYNE, The ATLAS Supervisor, p. 279, vol. 20, Proceedings of EJCC, 1961, Washington, D.C. (AFIPS).
9. W. F. BAUER, WHY Multi-Computers, Datamation Magazine, September 1962.
10. F. J. CORBATO, M. MERWIN-DAGGETT, and R. C. DALEY, An Experimental Time-Sharing System, p. 335, Proc. SJCC 1962 (AFIPS) (see also reference 19).
11. H. KOLSKY, Centralization vs. Decentralization, Tenth Annual Symposium on Computers and Data Processing, June 26–27, 1963.
12. J. D. EDWARDS, An Automatic Data Acquisition and Inquiry System Using Disk Files, (Lockheed Missiles and Space Co.) Disk File Symposium, March 6–7, 1963 (Informatics, Inc., Culver City, California).
13. L. W. MCCLUNG, A Disc-Oriented IBM 7094 System, Paper #3, Disk File Symposium, March 6–7, 1963, Hollywood, Calif. (sponsored by Informatics, Inc.).
14. The RW-400—A New Polymorphic Data System, p. 8–14, Datamation, v. 6, No. 1, Jan./Feb. 1960.
15. A. J. PERLIS, A Disc File Oriented Time Sharing System, Disk File Symposium, March 1963, (sponsored by Informatics, Inc., Culver City, California).
16. J. D. PENNY and T. PEARCEY, Use of Multiprogramming in the Design of a Low Cost Digital Computer, Comm. ACM, p. 473, v. 5, No. 9, September 1962.
17. JAN A. RAJCHMAN, A Survey of Computer Memories, p. 26, Datamation, December 1962.
18. J. A. WARD, The Need for Faster Computing, p. 1, Proc. Pacific Computer Conference, March 1963, IEEE (T-147).

19.

A) S. A. COONS, An Outline of the Requirements for a Computer-Aided Design System, p. 299, Computer Aided Design—1963 SJCC.

B) D. T. ROSS and J. E. RODRIGUEZ, Theoretical Foundations for the Computer-Aided Design System, Computer Aided Design, 1963 SJCC, p. 305.

C) R. STOTZ, Man-Machine Console Facilities for Computer-Aided Design, p. 323, Computer Aided Design, 1963 SJCC.

D) I. E. SUTHERLAND, Sketchpad: A Man-Machine Graphical Communication System, p. 329, Computer Aided Design, 1963 SJCC.

E) T. E. JOHNSON, Sketchpad III: A Computer Program for Drawing in Three Dimensions, p. 347, Proc. of 1963 SJCC, Detroit, Mich., May 1963.

20. W. BUCHHOLZ (editor), Planning a Computer System—Project Stretch, McGraw-Hill Book Co., Inc., N.Y. 1962. (see also IBM 7030 (STRETCH) Manual)

21. E. C. SMITH, JR., Simulation in Systems Engineering, p. 33, IBM Systems Journal, vol. 1, September 1962.

22. G. F. GORDON, A General Purpose Systems Simulator, p. 18, IBM Systems Journal, Sept. 1962. See also p. 87, Proc. of EJCC, December 1961.

23. TAKASHI ISHIDATA, SEIICHI YOSHIZAWA, and KYOZO NAGAMORI, Eddycard Memory—A Semi-Permanent Storage, p. 194, vol. 20, EJCC, 1961.

24. G. M. AMDAHL, New Concepts in Computing Systems Design, Proc. IRE, vol. 50, no. 5, May 1962 (Memory Protection).

25. F. S. BECKMAN, F. BROOKS, JR., and W. J. LAWLESS, JR., Developments in the Logical Organization of Computer Arithmetic and Control Units, Proc. IRE, vol. 49, no. 1, January 1961.

26. T. KILBURN, D. B. G. EDWARDS, M. J. LANIGAN, and F. H. SUMNER, One Level Storage System, p. 223, vol. EC-11, #2, April 1962, IRE Transactions on Electronic Computers.

Manufacturer's descriptive literature on the following systems was also consulted: Gamma 60, Burroughs D-825, CDC-3600, IBM 7090, Burroughs B-5000.

BIBLIOGRAPHY

27. E. F. CODD, Multiprogramming Scheduling, Comm. ACM, vol. 3, June 1960.

28. W. J. LAWLESS, Developments in Computer Logical Organization, Advances in Electronics and Electron Physics, vol. 100; Academic Press, Inc., New York, 1959.

29. A. L. LEINER, W. A. NOTZ, J. L. SMITH, and W. W. YOUDEN, Pilot Multiple Computer System (Manual), National Bureau of Standards Report 6688. See also Journal of ACM, vol. 6, no. 3, July 1959.

30. H. A. KEIT, Polymorphic Principle in Data Processing, 1960 IRE Wescon Conv. Record, pt. 4, pp. 24–28.

31. J. M. FRANKOVICH and H. P. PETERSON, A Functional Description of the Lincoln TX-2 Computer, p. 146, 1957 Western Computer Proceedings.

32. J. P. ECKERT, J. P. CHU, A. B. TONIL, and W. F. SCHMITT, Design of Univac—LARC System I, Proc. EJCC, Dec. 1959.

33. W. LONERGAN and P. KING, Design of the B5000 System, Datamation, vol. 7, no. 5, May 1961.

34. N. LANDIS, A. MANOS, and L. R. TURNER, Initial Experience with an Operating Multiprogramming System, Comm. ACM, vol. 5, May 1962.

35. F. P. BROOKS, JR., A Program Controlled Program Interrupt System, Proc. EJCC, December 1957.

36. J. W. WEIL, A Heuristic for Page Turning in a Multiprogrammed Computer, p. 480, Comm. ACM, v. 5, no. 9, September 1962.

37. M. J. MARCATTY, F. M. LONGSTAFF, and A. P. WILLIAMS, Time Sharing on the Ferranti-Packard FP 6000 Computer System, p. 29, vol. 23, 1963 SJCC (AFIPS).

38. R. J. MAHER, Principles of Storage Allocation in a Multiprocessor Multiprogrammed System, Comm. of ACM, vol. 4, Oct. 1961, p. 421-22.

39. N. STERNAD, Programming Considerations for the 7750, p. 76, IBM Systems Journal, vol. 2, March 1963.

40. F. R. BALDWIN, W. B. GIBSON, and C. B. POLAND, A Multiprocessing Approach to a Large Computer System, p. 64, IBM Systems Journal, vol. 1, September 1962.

41. E. S. SCHWARTZ, Automatic Sequencing Procedure with Application to Parallel Programming, Journal of ACM, v. 8, pp. 513–537, Oct. 1961.

42.

A) S. I. GASS, et al., Project Mercury Real-Time Computational and Data Flow System, p. 33, Proc. EJCC, December 1961 (AFIPS).

B) M. B. SCOTT and R. HOFFMAN, The Mercury Programming System, p. 47, Proc. EJCC, December 1961.

43. A Survey of Airline Reservation Systems, p. 53, Datamation, June 1962.

44. F. H. SUMNER, G. HALEY, and E. C. Y. CHEN, The Central Control Unit of the ATLAS Computer, p. 657, Proc. of IFIP Congress, 1962.

ACKNOWLEDGEMENT

Comments and assistance from several reviewers is gratefully acknowledged in reducing the large mass of published information on this subject to manageable form. Thanks are especially due to G. Hollander for his pertinent comments and frequent review.

ORGANIZING AND PROGRAMMING A SHIPBOARD REAL-TIME COMPUTER SYSTEM

Dr. George G. Chapin
UNIVAC Division of Sperry Rand Corporation
Saint Paul, Minnesota

INTRODUCTION

In recent years, Naval warfare has been altered radically with the introduction of new, ultra-complex, and tremendously effective combat weapons and associated support systems. A requirement of nearly all of these systems is the capability to process information. A variety of computing systems have evolved to meet this requirement. Initially the computers were special purpose; that is, designed to solve the data processing problem associated with a specific and, by today's standards, simple system. As the capability of weapons and support systems has grown, the data processing requirements have also increased in scope and complexity, providing an ever-increasing challenge to the computer industry. The trend has been to satisfy these requirements by designing larger, integrated systems around general purpose, stored-program digital computers. This approach has eliminated many of the problems encountered with the use of small special-purpose systems but has created another type of problem—the programming of the general purpose computers. This paper will discuss some aspects of this programming problem, using the Naval Tactical Data System development as a prime example.

DESCRIPTION OF THE NAVAL TACTICAL DATA SYSTEM

The Naval Tactical Data System (NTDS), operating in real time aboard ships, is a multi-computer combat direction system which processes, correlates, and evaluates tactical data. Real-time inputs are inserted into the system from sensors and from other systems. Real-time outputs are provided to other systems internal and external to the ships.

Operationally, this computer-centered control system *coordinates* the collection of data from the sources aboard ship (radar, sonar, navigation inputs, etc.) and from external sources via communications links; *correlates* the data to obtain a clear picture of the tactical situation; *processes* the data required for the decision-making function; and *communicates* the decisions for action to the weapons systems selected. This rapid exchange of tactical information between ships and units of the fleet permits a high degree of task force integration and coordination while enhancing the ability of each ship to perform its separate mission. By solving problems at computer speeds rather than by voice and/or manual means, NTDS has increased combat effectiveness by:

- providing increased individual ship capability to automatically process and evaluate tactical data which the ship itself collects;
- automatically giving each ship immediate access to evaluated tactical information held by other ships in the formation;
- permitting various levels of command to appraise more realistically the current tactical situation in its entirety (by pro-

viding common access to the total tactical information accumulated by ships in formation);
- increasing effectiveness in executing orders.

Two essential elements of an NTDS complex are men and machines. The object of this relationship is to remove from the operator, to the maximum practicable extent, tiring and repetitive operations in order to concentrate his effort in areas requiring decisions based on judgment and experience. Most operators are positioned at display consoles in the Combat Information Center (CIC) of the ship, where they are in direct communication with the computers. In addition to the displays and general-purpose stored-program computers, the system includes a number of other types of equipment, such as serial and parallel digital communications links, radar switchboards, radar video processors, manual keyset entry devices, magnetic tape systems, etc. A typical but oversimplified system configuration is shown in Figure 1.

A third essential element of an NTDS complex is the computer programs. These programs are to the equipment part of NTDS as the brain and nervous system are to the human body. They manage the operation of the equipment complex by controlling information flow and processing procedures, and by automatically executing many minor decisions.

BASIC SYSTEM DESIGN PRINCIPLES

A number of basic ground rules guided the

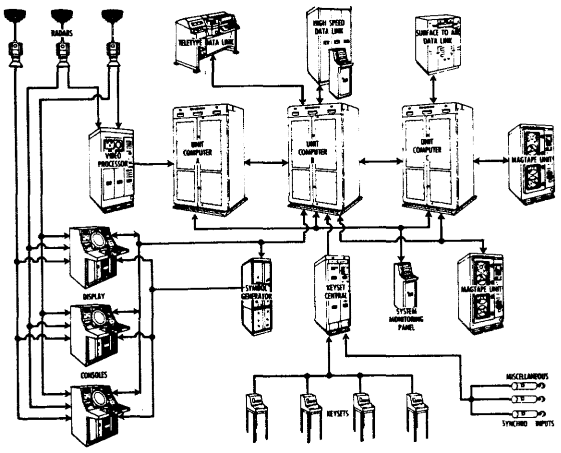


Figure 1. Typical System Configuration

development of NTDS. Some of the more important were:

- The number of equipment types was to be minimized—to reduce logistics, training, and maintenance problems.
- Ships with different missions, different sensors and weapons systems, and different data processing requirements were to be outfitted with the same NTDS equipment types.
- Additional and changing requirements were to be imposed on the system—including new and undefined sensors, weapons systems, and tactics.
- System reliability required for twenty-four hour-per-day operation was not to be achieved by "duplexing" all equipment (because of cost and space limitations)—in fact, all equipment was to be operating during periods of maximum system capability.
- Failure of a system element was not to result in a complete loss of capability; rather, the system was to operate in a degraded mode if necessary.

In order to satisfy these ground rules, a number of system design principles were established for NTDS at the start of development. Some important principles related to data processing were that:

- A standard computer should be used for all installations.
- Capacity requirements should be met by using multiple units of the standard computer, hence the term "unit computer".
- The unit computer should be general purpose, but with emphasis on input/output and data manipulation ("bit-fiddling" as compared with arithmetic operations).
- At least two unit computers should be used for any installation in order to satisfy the high reliability and twenty-four hour-per-day requirements. Full operational capability of any installation should require all computers, with fewer than the maximum number resulting in degraded system performance.
- Changes in the system because of additional or changing requirements, either

during development or after becoming operational, should be met by reprogramming and, if necessary, by addition of one or more unit computers.

The application of these principles made it necessary to design NTDS with no stand-by equipment. Equipment configurations were established for different installations involving two to four computers and variable numbers of displays and other equipment types. A set of operating modes was defined which provided the ability to handle the most important functions (at reduced capability for some modes) in all but the most extreme conditions of damage and malfunction. A secondary advantage of the design is that under low load conditions parts of the system can be used for other purposes such as simulation, training, and shipboard inventory control.

In order to emphasize the size of some of the systems now in operation, NTDS requires between forty and eighty equipment components per ship, depending upon type of ship. Most of these equipment components communicate directly with the two or more computers which are required for each ship, depending upon the type. As to reliability of the system, over the initial eight-month test period, seven shipboard computers averaged a mean-time-between-failure of fifteen hundred hours. At no time during the test was a system out of commission because of any equipment malfunction.

THE UNIT COMPUTER

The heart of the NTDS is the AN/USQ-20 Unit Computer (UNIVAC®1206); a general-purpose, stored-program, solid-state machine capable of processing very rapidly large quantities of complex data. The computer, shown in Figure 2, features the following major characteristics:

- Internal high-speed storage with a cycle time of eight microseconds, an access time of 2.8 microseconds, and a capacity of 32,768 words;
- Internally stored program;
- Thirty-bit instruction word length; 30-bit or 15-bit data word length;

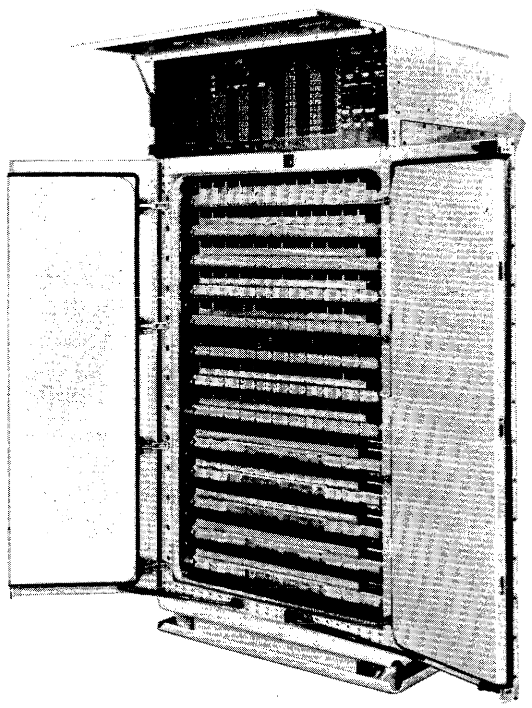


Figure 2. The UNIVAC AN/USQ-20 Unit Computer (with Doors Open)

- Single-address instructions with provision for address modification via seven index registers;
- Repertoire of 62 instructions, most of which provide for conditional program branches;
- Average instruction execution time of 13 microseconds;
- Programmed checking of data parity;
- Parallel, ones' complement, subtractive arithmetic;
- Internal millisecond real-time clock, with modulus of six days;
- Twelve input and twelve output channels for rapid data exchanges with external equipment without program attention;
- Two special input and two special output channels for direct intercomputer data transfer;
- Sixteen-word wired auxiliary memory for automatic recovery in event of program failure and for automatic initial loading of programs (bootstrapping);

- Entirely contained (including memory and input-output) in a cabinet less than fifty cubic feet in volume;
- Rugged, easily maintainable construction.

INPUT-OUTPUT CHARACTERISTICS

The programming methods employed in the NTDS are very dependent on the input-output characteristics of the unit computer. The features of interest are discussed in the following paragraphs.

There are fourteen input and fourteen output channels, each containing thirty data lines and three control lines. Twelve input and twelve output channels are used to communicate with peripherals. In most cases, one input and one output channel communicates with an equipment type which may consist of many items of equipment. The remaining two input and two output channels are specially designed for intercomputer communication.

Data transfer on one input or output channel is activated by a single instruction. Thus the initiation of the data transfer is under program control. The data itself is transferred at a rate determined by the peripheral equipment. Normally a number of words are sent. No attention is required of the computer program after initially activating the transmission. This method of data transfer is referred to as buffered transmission.

All input and all output channels may be active at the same time.

Input or output of a word from memory delays the computer program being executed. Two memory cycles are required per word.

Peripheral equipment must request each input or output word transfer. On output (see Figure 3), such a request (called an Output Data Request) tells the computer that the peripheral equipment can accept a word. When the computer places data on the lines, it also sends a control signal (called an Output Acknowledge or an External Function) to tell the peripheral that the data is available. It is then the responsibility of the peripheral equipment to accept the data in a short time period. On input to the computer, the request (called the Input Data Request or Interrupt) tells the computer that data is on the input lines. The request and data must be held until accepted by the computer, as evidenced by a signal sent from the computer (the Input Acknowledge).

The Output Acknowledge control signal accompanies each word from the computer sent during a buffered transmission. The External Function control signal accompanies commands sent to the peripheral (i.e., a Read Forward One Block command to a magnetic tape system).

The Input Data Request control signal accompanies each data word sent to the computer during a buffered transmission. The Interrupt control signal accompanies status type information whose occurrence cannot be predicted (i.e., an End of Tape signal from a magnetic tape system).

For intercomputer transfer, the transmitting computer behaves like the peripheral. This requires the transmitting computer to hold the data until accepted by the receiving computer.

The buffered transmission process transfers words to or from consecutive memory addresses starting at a given initial address through a given terminal address. In order to inform the program of completion of a particular buffered transmission, an Internal Interrupt can be

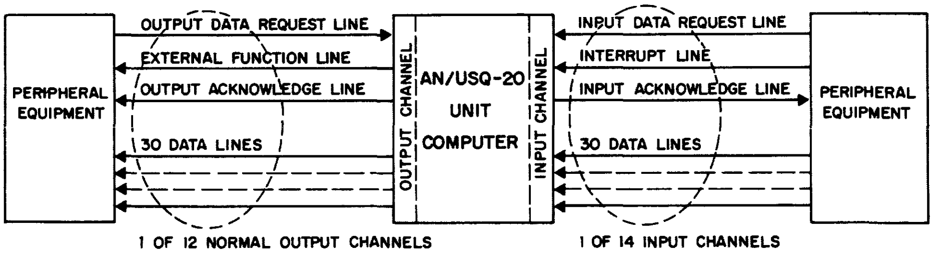


Figure 3. Normal Channel Configuration

generated (at the option of the programmer) when the buffer mode terminates. An Internal Interrupt is associated with each input channel and each output channel; a separate entrance register in computer memory is reserved for each Interrupt. The unique entrance address thus defines the source of the Internal Interrupt request for action by the appropriate Interrupt routine.

NATURE OF NTDS DATA AND PROCESSING REQUIRED

The input-output problem consists of a "simultaneous" transfer of data with as many as twelve equipment groups consisting of up to fifty equipment items. The word "simultaneous" means that in a short period of time (say, two hundred microseconds) data may be transferred between the computer and several equipment groups. Data transfer is "random", that is, has no predetermined order, as distinguished from "sequential", where a group of data must be transferred to or from one equipment before a transfer with a second equipment begins.

The maximum input-output data rate experienced by any computer is 12,000 words per second or about twenty percent of the computer time. The peak data rate over a period of a few milliseconds may be higher than this but is considerably less than the computer's peak data rate of 60,000 words per second. The system is so designed that it is not possible to miss a transmission of data because of an input-output overload. This is possible primarily because of the input-output characteristics of the computer. On an average, the computers spend less than ten percent of their time on input-output.

In general, many items of data are received from or sent to the same equipment. All data words must contain information identifying their contents. The data transferred (other than control information) are of two basic types. One type is normally sent only once. Care must be taken to have positive acknowledgement of such data so that they can be sent again if not received properly. For example, entry of status information by a keyset is done only once. Such data are normally retained by the system until a new entry is made. A request for computation is made just once by an operator and is acknowledged by a display of the results of the computation. The second type of data is somewhat "cyclic" in nature, that is, similar data are sent on a more or less periodic basis. Since most NTDS data are cyclic, the ratio of data received by the system to data stored by the system is quite high compared to "information retrieval" systems. Therefore, it is possible to store all data in the computer. Magnetic tape is available but is used for such things as program storage, simulated inputs, and data recording.

The real-time inputs to the system are mostly physical quantities. After conversion to digital form, a typical quantity is no more than twelve bits as it enters the computer (with the exception of time, which is carried as a thirty-bit quantity to the nearest millisecond). Allowing for round-off, multiple updating between successive inputs of the same quantity, etc., no more than fifteen bits of computer memory are required for most items derived from a physical quantity.

Since the computer permits direct arithmetic operation on a fifteen-bit half word without masking and shifting, most items requiring frequent arithmetic manipulation (such as coordinate and velocity information) are stored as half words, even though they are less than fifteen bits. In order to efficiently utilize the computer memory, some less frequently used data items are packed into stores, thereby often requiring masking and shifting prior to use.

Each Operational Program must work with many different kinds of data items, normally over one thousand. Some data items represent numbers, other computer addresses, and still others are coded representations of information. In this latter category, consider a quantity like Identity, in which such things as Friend, Hostile, Unknown, etc., are assigned digital codes.

In designing the data stores, a trade-off can be made between efficient utilization of storage and efficient utilization of time, based on the relative availability of computer storage compared with computer time. The problem is that the optimum design of the data stores cannot be predicted completely in advance. As a result, it is almost a certainty that changes in the data design will be required to make up for omissions, changes in problem solutions, and errors. What is needed is a means of changing the data designs which has minimum effect on the programs using the data. The solution used on NTDS is the CS–1 Compiling System which separates the data design from the input program. Many new programs can be obtained after a change in the data design by a recompilation of the same input programs.

The data available to the system must undergo considerable processing. Much of this processing involves "bit-fiddling" as contrasted to arithmetic operation. Bit-fiddling means such things as masking, selective setting or clearing of bits, shifting, sorting, indexing, etc. In a typical Operational Program, less than one-half of one percent of instructions are multiply and divide and only about ten percent involve additive or subtractive processes.

An interesting and somewhat surprising fact is that in a typical two computer program less than two percent of the data items are common to both computers. Thus, there is very little storage efficiency lost by properly segmenting the problem into two computers.

PROGRAM ORGANIZATION AND CONTROL

Master Control

It is important to have a means of exercising overall internal control over the computing system, especially in a multi-computer system. Such a means was provided by the Systems Monitoring Panel (SMP) (Figure 4) and its associated programs. The SMP incorporates the following capabilities:

- A data entry facility, permitting the SMP operator to direct computer complex operation. Commands originate at the SMP to load system programs, execute or terminate system operations, initiate emergency recovery procedures, etc.
- A data output facility, permitting display of system status and alarm information. Selected computer-detectable faults in equipment and programs are displayed.

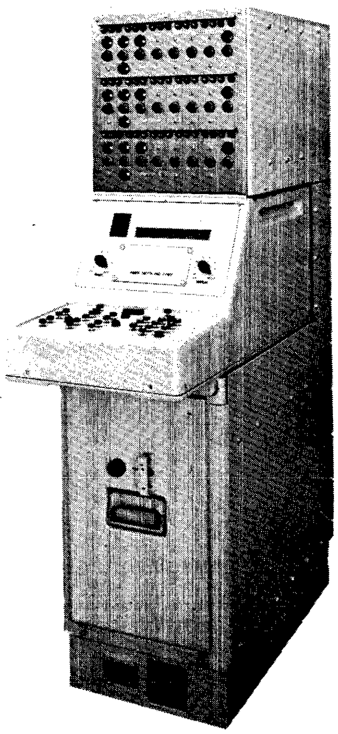


Figure 4. System Monitoring Panel

This gives the SMP operator an indication of overall system performance and permits him to decide when subsystem maintenance and/or an alternate mode of operation is required.
- Remote computer controls, making the SMP a central location from which to operate each individual computer.

The SMP has a unit for each computer which includes certain manual control functions from the computer maintenance panel. Only controls needed for efficient remote computer operation are included, for example, Bootstrap activation, Stop and Jump switches, Stop, etc. The SMP operator uses a keyset to perform the SMP data entry function with each computer. The keyset is connected to a normal input channel on each computer, and keyset entries are directed to a specific computer by means of computer selection buttons on the keyset. The keyset initiates communication by means of a computer Interrupt and does not require programmed interrogation.

The data output facility takes the form of indicator panels, one for each computer. Each panel consists of a fifteen-bit register with each bit position of the register controlling the state of an associated indicator lamp. The indicator panel is connected to a normal computer output channel over which the computer sends a fifteen-bit word to activate the indicators. These bits are coded to indicate status and alarm conditions and to provide feedback acknowledging an operator action.

Program Control—The Executive Routine

A real-time system can be defined as a system in which the delay in responding to an input is negligible in the time reference of the user equipment or operator. If, because of overload, the system loses inputs or fails to process the inputs properly, saturation has occurred. Since many inputs occur randomly in a man-machine system such as NTDS, the computer must be able to handle a peak load situation where many requirements are essentially simultaneous. At the same time, the data processing capability of the system should be as close as possible to the theoretical minimum, that is, sufficient to handle the average input rate from all input functions, based on a long time period. The problem, then, is how to schedule the functions to be performed so that peak loads are smoothed out, while at the same time performing all tasks within their real-time requirements.

To solve this problem an Executive Control Philosophy has been implemented which includes recognizing system tasks in order of *system* priority, distributing them among the system's computers, and then controlling them in the individual computers of the system by an *Executive Routine* within each computer. The Executive Control Philosophy is based on the fact that each function and associated handling of peripheral equipment defines a set of tasks that must be performed by the computers. (Functions are such things as Navigation, Automatic Tracking and Intercept Control.) Each task is performed by a computer subroutine (or group of routines) called a "subprogram". By definition, a subprogram is any subroutine referenced directly by the Executive Routine. An Executive Routine is contained in each computer of a multicomputer program and maintains complete control of the operational program within that computer. Overall control of a multicomputer program is not employed; each computer controls execution of its assigned tasks.

Subprograms within a computer are considered for execution on the basis of their priorities as system components. System priority for subprograms is relative, and is determined in the following manner: If tasks corresponding to subprograms A and B have response requirements of one hundred milliseconds and one second, respectively, then subprogram A would have a higher system priority than subprogram B. System priorities for all other subprograms are chosen similarly.

Consider a list of all subprograms ordered by priority, with the highest priority at the top. The Executive Control Philosophy, applied in an elementary form, would dictate that this list be scanned sequentially starting at the top, executing subprograms when they are needed. After each subprogram execution, the scanning process would be resumed, again starting at the top of the list. Thus, under a peak load condition where many subprograms require execution, the top priority subprogram would be executed instantaneously, the next highest immediately following, and so on. The resultant response time of each subprogram would be the sum of previous execution times for all higher priority subprograms, recognizing that some of the higher priority subprograms may be executed more than once. The assumption must be made that sufficient computational capability exists to perform all assigned tasks. If this is not true, then complete penetration of the list will not be realized and the only solution is additional computing capability. If computational capability is sufficient to handle the average input rate for all functions, then under peak load conditions higher priority tasks are performed first and lower priority tasks are delayed until time is available. Thus, a lower priority task may be defined as not having a critical response time; late execution will not seriously degrade the system. An example of such a task is output to displays which should occur at a rate of twenty times a second to maintain a flicker-free presentation. However, during peak load, which probably would last for less than a second, reducing the output rate to nineteen, eighteen or less would normally go unnoticed.

The Executive Routine which implements the Executive Control Philosophy consists of an eight instruction main control loop, some interrupt processing and "flag" setting routines, and three tables per computer. One of the tables contains time entries which are scanned by the main control loop program to determine the subprogram that must be executed next. It is here that the real-time clock of the computer is used; without the clock, the simply-implemented Executive Routine would not be possible.

Program Organization

An Operational Program is organized into component subprograms in each computer. There are from ten to twenty-five subprograms in a typical program. The subprograms are of varying complexity, some having as many as six subroutine levels. Subprogram entries are always controlled by the Executive Routine. This rule implies that no subprogram calls on another subprogram; however, a subprogram by a Return Jump instruction may use a subroutine which is common to other subprograms.

The overall organization of a typical program is shown in Figure 5. The Executive Routine calls on one of the subprograms, A, B, ... N, based on a need for its execution. The subprogram is executed and returns control to the Executive after completion or after a certain maximum time even though not completed. The subroutine also may "flag" another subprogram for execution or may "flag" itself, but execution is postponed until the Executive Routine selects the subprogram.

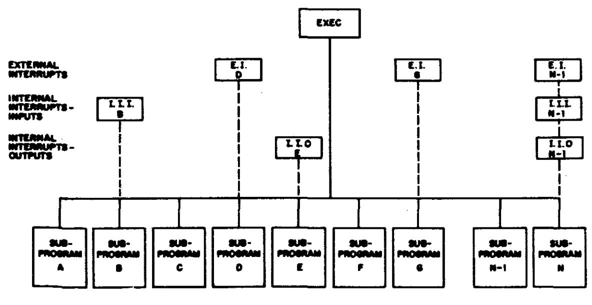


Figure 5. Typical Program Configuration

At any time, one of the many Interrupts may be activated. A dotted line indicates the subprogram effected by each Interrupt; that is, each Interrupt triggers a subroutine in the Executive which sets a "flag" causing a later entry to that subprogram. The upper level of Interrupts (EI–D, EI–G, etc.) are the External Interrupts received from external equipment. The other two types of Interrupts shown are output and input Internal Interrupts. The capability of overriding the Executive Routine by System Monitoring Panel action is not shown.

In a two computer complex each computer has two subprograms for intercomputer data transfer. One subprogram is associated with encoding data on output, the other for decoding data on input. The basic technique used is "ping-ponging", that is, each computer alternately sends and receives a variable length block of data at an approximately periodic rate. The first word transferred contains the beginning and end addresses for storage of the block. Subsequent words make up messages containing two words or a variable number of words. Each message has information identifying its length and content as well as error detection bits. After the last message is transferred, internal interrupts are generated in each computer. These interrupts are used by the Executive to initiate at the appropriate time the reverse transfer of data and the necessary encoding and decoding subprograms.

ASSIGNMENT AND DISTRIBUTION OF TASKS BETWEEN COMPUTERS

Basic questions that must be solved in any multicomputer system are:

- How many computers are required to obtain the capacity and function capability specified for each operating mode of the system?
- How are the tasks to be distributed between computers in a multicomputer program?

Initial estimates for NTDS showed that two computers in one case and three computers in another case would be required to obtain maximum system capacity and capability specified for the first ships. Capacity and capability as specified for reduced operation would require one and two computers. These initial estimates have proved surprisingly accurate.

Program design commenced and tasks were defined and distributed—always with the three-computer program in mind. Whenever more than one function requires the same task, then this task is called a Service Task. For example, the computer interrogates all of the display consoles at frequent intervals to determine if any operator has entered data or asked something of the system. The process of interrogation and decoding the results is common to the many functions which utilize display inputs.

All tasks were grouped into three categories called Service (as defined above), Tracking, and User. Included as a Service Task was the encoding, decoding, and buffering associated with intercomputer data transfer. In a three-computer complex, one computer was then assigned to each category of tasks. The only exception to this distribution is that subprograms associated with inter-computer data transfer (classified as a Service Task) are contained in each computer. Equipments associated with a specific task are connected to the computer performing that task; the majority of equipments are thus connected to the Service computer.

Most tasks having a critical response time are associated with the Service computer. In general, Service tasks associated with inputs (and outputs where stringent timing requirements are imposed by external equipment) have highest system priority, followed by Tracking tasks, User tasks, and Service tasks associated with output. Each computer also contains a subprogram of lowest priority to perform such operations as program checking when no other Service task requires execution. As a result, very rigid timing restrictions are required on all subprograms within the Service computer, i.e., allowable time away from the Executive Routine is very small, on the order of ten milliseconds. Since most subprograms corresponding to the Service tasks are short, this presents no real problem. Conversely, most tasks assigned to Tracking and User computers do not have critical response times (those that do are processed by interrupt routines) and rigid timing restrictions are not necessary. Since most subprograms corresponding to Tracking and User tasks are long and time-consuming, to impose timing restrictions on the order of ten milliseconds would reduce overall system efficiency.

On a three-computer program the Service computer acts as a focal point for all system data and transmits all pertinent data to the Tracking and User computers for processing. Intercomputer transfer occurs between Service and Tracking computers and between Service and User computers. No direct transfer occurs between Tracking and User computers. Tracking data are transferred to the Service computer, which in turn relays the appropriate information as necessarily modified to the User computer along with other system-generated data. Similarly, appropriate User data are relayed to the Tracking computer.

The primary differences between a three-computer and a two-computer complex are capacity and the degree of sophistication and extent of the Tracking and User functions. There are also differences peculiar to various classes of ships. Generally, however, the Service tasks remain relatively unchanged. Therefore, in distributing tasks for a two-computer program, the Tracking and User tasks are merged into a single Tracking/User computer, with the Service computer remaining as before.

Although there is no distribution problem in a one-computer complex, many multi-computer programming techniques are used. Intercomputer messages are packed in intercomputer buffers (which are never transmitted) and unpacked by an unpacking routine which is similar or identical to the routine used in a multicomputer program. The advantages gained in programming and automatic system recovery negate the normal inefficiencies in this design.

ERROR DETECTION AND CORRECTION

One ground rule for the program is that all input data be checked in order to minimize the probability of incorrect data entering the system. This ground rule is imposed even though the equipments themselves make certain checks on data because, in most cases, it is better not to have data at all than to have incorrect data. This statement is applicable to all control information, to all data entered only once, and to some "cyclic" data. With "cyclic" data, it is fairly easy to eliminate large errors by applying reasonableness checks. In the case of data originating on ownship, the probability of entering correct data can be made quite large by reliable equipment, operators being the most frequent error sources. It is sufficient to use such things as reasonableness checks, check summing, parity checks, etc., and often to "close the loop" by acknowledging the information.

The more difficult situation occurs when information is received over a data link. In this case the probability of an error in the data is significantly higher because of radio link transmission problems. There are many methods of increasing the probability that a specific item of information received over a data link is correct. All such methods involve the transmission of redundant information.

If redundant data or control signals are transmitted, it is often possible to detect errors in the received information and in some cases to correct these errors. The amount of error detection and correction possible depends on the amount of redundancy, which in turn is dependent on such factors as net load, effect of incorrect data on the system, etc. For example, a low capacity net can afford to send the same information many times and possibly to acknowledge receipt of the data. A high capacity net, however, must operate more efficiently.

The techniques used on NTDS vary with the data link. A combination of logic built into the communications terminals and "filters" supplied by the computer program are used to decode the received information. It is possible to design the program filters to reduce to any desired level the amount of incorrect data accepted as correct. The penalty of a very low error rate is that some correct data are rejected. Thus a compromise is necessary between the desire for a low error rate and the desire to minimize the amount of correct data rejected.

SYSTEM RECOVERY AND CHANGEOVER

An obvious but important design principle for any complex system is that the system will fail. Failures may be intermittent or catastrophic and, in a system such as NTDS, may be caused by malfunction of the computer, computer program, any other equipment, or operators. Malfunction of the computer program is particularly difficult to prevent, since the logical complexity of the system is such that only a fraction of the paths through the program can be examined prior to system operation, even with the assistance of extensive simulation. Also, the failure of peripheral equipment, whether intermittent or catastrophic, can cause false data or program errors in addition to loss of the use of the equipment itself.

Methods of compensating for these failures should be considered early in the system design. In the NTDS, the detection and correction of malfunctions while the system is in operation is accomplished by the System Recovery function. This function includes the processes of:

- Detection and classification of a system malfunction caused by the computer, computer program, or any other equipment;
- Isolation of the malfunction to an item of equipment or a portion of the computer program, with a detailed indication of errors to an operator if required by the situation;
- Correction of the malfunction with or without changing the system configuration. If a change of computer is required, the process of change is called Computer Changeover.

Among the programming techniques provided for error detection are program checksums, data reasonableness checks, equipment diagnostic tests, and checks for irregularities in anticipated equipment and operator responses. Although automatic error detection is generally used, certain errors may be most effectively detected manually, such as program looping, lack of power on certain equipments, and unreasonable display patterns. Program loops will tend to cause improper equipment response and thus will quickly come to the notice of an operator. In some instances, a fault condition can be rectified by automatically dropping the particular equipment from the system complex. An operator is notified of the action but does not initiate it.

Other situations require manual intervention by the operator at the System Monitoring Panel. The operator has controls which allow: selection of program modifications; elimination of functions, if necessary; and rapid system restart. From the indicators available at this panel, the operator can quickly ascertain the operating condition of the system and the specific nature of errors. In the case of *computer program* failure, a wired-in bootstrap program is available. The bootstrap program initiates an action which reloads the appropriate programs from magnetic tape, sometimes via an intermediate computer. The data stores may or may not be re-initialized. In some cases of *peripheral equipment* failure, it may suffice merely to drop that equipment from the system. In other cases, the system must be modified. If *computer* failure should occur, controls at the System Monitoring Panel permit a Computer Changeover process to be initiated.

Computer Changeover, when required for recovery or for normal operation (such as maintenance and mode change), includes such processes as:

- Entry of programs;
- Movement of data stores so that the current status of the system is preserved;
- Switching of equipments in a well-defined manner;
- Re-establishment of equipment synchronization in some cases;
- Establishment of communications with any equipment added to the system.

Techniques have been developed which permit Computer Changeover in most cases without significant interruption in system operations, provided no System Recovery is involved. A few seconds to several minutes are required, depending on the changeover being accomplished. The most significant time delays are caused by Magnetic Tape movement and by semi-automatic equipment switching (that is, switching requiring an action by a man as contrasted to an action by the computer program).

When Computer Changeover is required for System Recovery, the process gets much more involved. The principal difficulty is in preserving data in cases of computer or computer program malfunctions. The exact nature of the difficulty determines whether some, part, or all data stores can be recovered. In the worst case where no data stores can be recovered from a malfunctioning computer, critical information can be obtained from magnetic tape. Other data must be reacquired, either by starting certain processes over again or by using data obtained from other ships.

Another problem arises when the number of computers is changed. Since system capacity varies with the number of computers, some complicated juggling of the data stores is required as part of the transition.

CONCLUSION

This paper has discussed some of the programming techniques developed for the NTDS, a multicomputer, multisite, real-time system. The approaches used to assign and distribute tasks between computers, to organize each program, and to control the over-all system as well as the individual programs have been described along with the important problems of system recovery and changeover and error detection and correction.

A MULTIPROCESSOR SYSTEM DESIGN

Melvin E. Conway
Directorate of Computers, USAF
L. G. Hanscom Field
Bedford, Mass.

INTRODUCTION

Parallel processing is not so mysterious a concept as the dearth of algorithms which explicitly use it might suggest. As a rule of thumb, if N processes are performed and the outcome is independent of the order in which their steps are executed, provided that within each process the order of steps is preserved, then any or all of the processes can be performed simultaneously, if conflicts arising from multiple access to common storage can be resolved. All the elements of a matrix sum may be evaluated in parallel. The *ith* summand of all elements of a matrix product may be computed simultaneously. In an internal merge sort all strings in any pass may be created at the same time. All the coroutines of a separable program[1] may be run concurrently.

The problem is not so much finding procedures employing parallelism as it is finding computer systems which could handle the procedures without undue preplanning. A desirable system which flexibly accommodates a collection of identical, concurrently operating sequential processors should exhibit the following properties.

1. At every point in time the number of active processors should be the minimum of the number of processors in the system and the number of parallel paths in the program at that time.
2. If insufficient processors are available, program paths specified to be parallel should be executed serially.
3. Although the coder must specify all parallelism, he should have little concern about the number of processors in the system at execution time.
4. The means of specifying parallelism should be simply coded and rapidly handled by the system so that for highly parallel programs the processing time is inversely proportional to the number of processors, subject to the boundary condition that a one-processor system would run only slightly faster if the specifications of parallelism were removed from the program.

In short, a system which accommodates programs with parallel paths by means of a plurality of sequential computing elements should be dynamically self-scheduling. This paper suggests a design for such a system.

SPECIFYING PARALLELISM

Because a procedure can be thought of as originating at one point in its flowchart, all parallelism is the result of forks in flowchart paths. Figure 1 provides a convention for specifying such forks. Hereinafter, the word *fork* will have the meaning suggested by Figure 1. Parallel paths may rejoin at a *join*. A convention for drawing joins is also given in Figure 1.

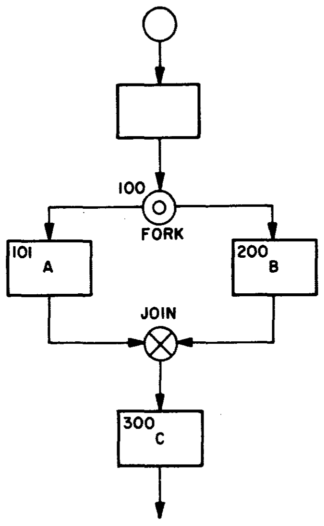


Figure 1. Conventions for drawing fork and join.

The fork and join in flowcharts have their counterparts in the FORK and JOIN instructions which are added to the instruction set of the system.* FORK is simply an instruction with two successors. It is written and acts like a branch instruction. However, if location 100 contains a FORK 200 instruction, then instructions at 200 *and* at 101 will be subsequently executed. The execution of a FORK instruction calls another processor into activity, if it is available. Notice that FORK has an associativity property; N parallel paths may be specified equally well by many possible arrangements of N—1 forks.

The JOIN, which is, in effect, the reverse of the FORK, has a vital additional job: it waits. In Figure 1, box C must not be begun until boxes A and B are completed. Assume that the coding for box A runs from location 101 through 105, that the coding for box B runs from 200 through 219, and that the coding for box C begins at 300. After the FORK at 100 is executed two processors are called to participate; one executes five instructions from 101 to 105, the other executes twenty instructions from 200 to 219. The processor finishing first, say at 105, should be released; the one finishing last then simply branches to 300.

In the case of an N—ary fork the only processor with a distinguished role is the one which finishes last, for it is the one which must branch. All others are released. If the N processors are operating independently, how does each know whether it is last to finish? Returning to the example, the information required to notify the last processor is available in the form of a counter at location 299. The FORK at 100 sets the counter to 2. Each processor, when it comes to the end of its parallel path, decrements the counter by one. The processor which produces zero as a result knows that it is last to finish. There are two JOIN instructions, at 106 and 220. Each one reads: JOIN 299. This means, "Decrement the counter at 299 by one. If the result is zero branch to 299 + 1. Otherwise release this processor." The FORK at 100 reads: FORK 200, 299, 2. This means, "Set the contents of 299 to 2. Then fork to 101 and 200."

If location 500 began a four-way fork to 503, 520, 540, and 560, all to recombine at a counter (called the *junction*) at 600, the coding would be thus:

```
500: FORK 520, 600, 4
501: FORK 540
502: FORK 560
```

This illustrates that a second kind of FORK is necessary. It forks but does not affect the junction.

So far we have described three new instructions peculiar to the system being developed. Here are two more. The first is a variation of JOIN which, instead of releasing its processor if it is not on the last path to finish, keeps it to do busy work. It reads: JOIN J, B and means, "Branch to B if something which ends up at J is still going on. Otherwise, JOIN J." It acts as follows. The counter at J is removed from storage and 1 is subtracted. If the result is nonzero, the *original value* is returned to J and a branch to B is executed. If the result is zero, J is set to zero and a branch to J + 1 is executed. We shall see later that this instruction is a generalization of the class of "Branch

* The fork-join notion has been around for a while. The equivalent of FORK is elsewhere given these names: in CL-II[2] and the Burroughs D825 AOSP[3], BRANCH; in the GAMMA 60[4], SIMU; in a conceptual machine discussed by Richards[5], BRT (Branch Transfer).

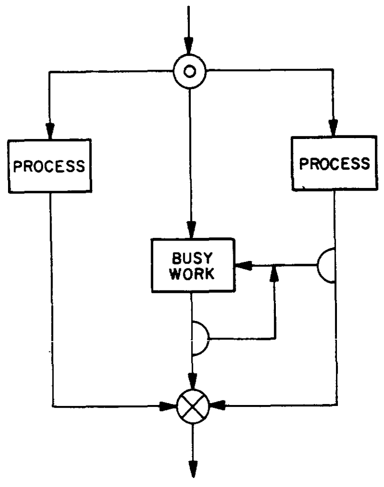


Figure 2. A convention for the "Branch to Busy Work" operation.

on busy I/O device" instructions. Figure 2 suggests a flowchart notation.

The other new instruction adjusts the value of the junction in the event that the possibility of execution of a FORK is conditional. It reads: FORK A, J and means, "Increment the value at J by one, then FORK A."

In summary, the five following instructions permit an adequate specification of parallelism:

```
FORK A, J, N
FORK A, J
FORK A
JOIN J, B
JOIN J
```

THE STATE WORD

One question which occurs to people thinking about multiprocessor systems may be stated thus: How much of the main memory should be private to each processor and how much should be "community" storage? The design to be presented here makes it clearly uneconomical to reserve any of the main memory for each processor. Indeed, if private storage is required, it belongs not to each processor but to each parallel flowchart path. The distinction between processors and paths is a crucial one; confusion over this matter seems to be muddying up much contemporary thinking about parallel processing. The four criteria stated in the introduction to this paper demand that the distinction be made. Processors have no identity of their own. During a computation they can be swapped, added, or removed without altering the results of the computation. What does have an identity of its own is a set of bits in each processor determining the state of the processor between instruction executions. This set of bits is called the *state word.* The notion of the state word will now be elucidated.

When the executive routine of a multiprogrammed single-processor computing system takes control from program A and gives it to program B it establishes an appropriate environment for the new program by storing all the processor registers used by program A in an area reserved for that program, and loading these registers from a similar area for program B. The content of such a reserved area preserves the state of a program at the time it is taken off the processor so that the same program can be later returned to the processor without any disturbance to the computation as a result of the interruption. We may call the content of a program's reserved area the state word for that program. Normally, a state word consists of at least an address (the sequence counter), and generally includes several arithmetic and index registers and a few indicator bits. We may call the aggregation of all the reserved areas for holding state words of inactive programs the *control memory.*

When the state word is loaded into the processor from control memory it occupies a set of storage positions the aggregation of which we might call the processor's *state register.* Thus, switching control from program A to program B may be conceptualized as a sequence of two processor operations: store state register into program A area in control memory; load state register from program B area in control memory.

When n programs share a common memory and a single processor the scheduling function consists of choosing the time intervals during which each of the n state words will occupy the single state register. This description of the scheduling function as a resource allocation can be generalized to the case wherein n programs share a common memory which is equally accessible to k identical processors: the scheduler attempts to optimize, according to some value scheme, the time-allocation of n state words to k state registers.

Consider a system with k processors, where $k \geq 2$. To take advantage of the potential overall speed increase without paying for state word transfer time we must build k flow paths, one for each processor, each capable of simultaneous operation between its processor and control memory. Ignoring timing restrictions in the control and main memories, we can see that for $1 \leq k \leq n$, where n is the number of state words (presumed fixed), the system speed (number of standard instructions executed per unit time) is proportional to k. For $n \leq k$ the system speed is constant.

In real life, of course, k is fixed and n varies with the amount of parallelism in the total system at any given time. What varies n? The FORK and JOIN instructions. FORK makes two state words from one, and JOIN (except the last one executed, for which the junction becomes zero) makes state words disappear. This suggests an easily implemented processor allocation algorithm: a processor executing a FORK sends one state word to control memory; one executing a JOIN halts until it receives a new state word from control memory. Richards[6] has shown that when the control memory is a single queue and when halted processors are given state words as soon as they are available, the allocation is not optimum in the sense of minimizing total execution time or maximizing processor duty cycle.

Figure 3 shows the system configuration as derived so far. Notice that there are two information subsystems. The control subsystem circulates state words, and the program subsystem shuttles instructions and operands back and forth. Note also that the only time that a channel between a processor and the control memory need be busy is after the processor executes a FORK or JOIN.

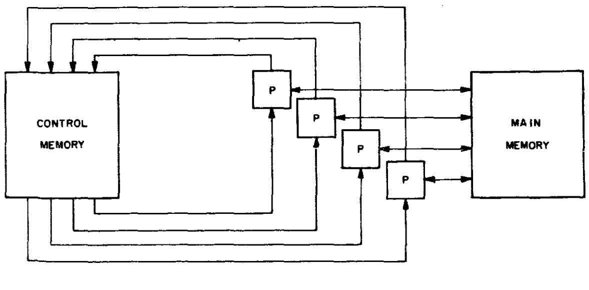


Figure 3. A tentative system configuration.

This design is so far unsatisfactory because serious flow bottlenecks can be expected at the two memories unless precautions are taken to avoid them.

Before we address this difficulty some observations are in order. First, notice that the FORK-JOIN approach provides no justification for distinguishing between parallelism within a program and parallelism between programs. The difference between simultaneous multiprogramming and a parallel algorithm is simply the position of the FORK instruction. This observation raises the hope that executive programs for a system of this sort will not be complicated by the parallel structure and may even be simplified by it.

Second, we might contemplate the role of the interrupt in this system. To rephrase the words of Buchholz[7] and others, interrupts have two distinct functions. The internal interrupt is triggered by the execution of a particular instruction and demands the insertion of a portion of code immediately following completion of this instruction. Overflow, divide check, and invalid address alarms exemplify the internal interrupt. The external interrupt is triggered by an external event not closely timed to the instruction currently being executed and which demands execution of a portion of code not necessarily related to the code being executed at interrupt time. The I/O operation complete and time clock interrupts are examples of this type.

External interrupts came into popular use when concurrent, program-controlled I/O was introduced. They attempted to control sequencing of parallel operations in a basically serial system. Because the code activated by an external interrupt could be executed in parallel with the code which is active at the time of interrupt (see the rule of thumb in the introduction to this paper), one might expect the handling of external interrupts to be different in the proposed system. In fact, external interrupts are unnecessary. Consider that an I/O instruction is simply a very lengthly one which may be executed in parallel with other code, such as computation and editing. Then simply precede it with a FORK and follow it with a JOIN, and all the functions of external interrupts are accounted for in a much more elegant manner. It is now seen how the "Branch to Busy Work" variation of the JOIN can be used as an I/O activity test.

Here is the first concrete illustration that the present structure simplifies executive programming. The elimination of the external interrupt provides simpler handling of interrupts for two reasons.

1. There are fewer interrupts to process and they all permit similar handling.
2. Internal interrupts are simpler to process because they are known not to occur at random. In particular, the routines processing internal interrupts can control further occurrences of interrupts during their durations.

While we are on the subject of simplification of the executive function, we might note that the extensive use of hardware in the processor allocation function can greatly simplify the executive program in comparison to a conventional multiprocessor system. Also, processor allocation in hardware helps make feasible goal number 4 stated in the Introduction.

The third observation we can make is that at any point in time during a computation there is no particular distribution of programs on the set of processors. Furthermore, observing the time history of any single processor reveals no particular sequence of programs or paths being serviced by that processor. This makes the problem of logging system usage by each program a nontrivial one for the system being discussed. This apparent anarchy also suggests that storage protection among programs might be hopeless; it will be seen that this is not the case.

Finally, we observe that with appropriate relief of certain bottlenecks in the system and with a certain class of highly parallel computations it may actually make sense to talk about speeding up the system by adding processors. The next two sections will attempt to show that the principal bottlenecks of the system are not essential. That is, they can be designed wide enough to match any prior choice of memory size and number of processors.

THE STORAGE SUBSYSTEM

In a system of the type being considered, particularly if its application involves servicing many independent programs, three problems related to the high-speed storage arise.

1. Complete storage protection must be incorporated in order to isolate the several programs.
2. Scavenging and allocation of storage to newly entering programs should not be an expensive process.
3. The effective service rate of the memory should not seriously depreciate the speed increase gained as a result of the addition of processors.

The first two problems can be solved fairly straightforwardly. The third problem is the kind which could keep a system like this from leaving the drawing board. However, there seems to be a sizeable class of applications for which the memory design presented here constitutes a solution to the third problem.

Assume for the sake of discussion that no individual program would require more than 2^{14} (16,384) words of storage and that the high-speed memory will never contain more than 2^5 (32) programs (including the executive) at any one time. All programs will be coded using a contiguous block of storage beginning at address zero.

When a program enters the high-speed memory it is assigned a five-bit program number by the executive. (Let zero be reserved for the executive program itself.) This program number is a constant of all state words in that program. Clearly then, the storage protection problem can be solved in principle by providing 2^{19} (524,288) words of storage and addressing the memory with a 19-bit "system address" which is a concatenation of the 14-bit address obtained from the program and the 5-bit program number. Such a worst-case design is of course not economical; there would frequently be large blocks of unused storage in the memory.

Now assume that the memory is divided up into small independent modules of, say, 2^8 (256) words each, such that the high-order 11 bits of a system address specify a module and the low-order 8 bits specify a word within a module. Now, instead of being forced to buy 2048 modules, let us elect to buy only 100 modules in a system. If no problem mix requires more than 25,600 words of storage (assuming also that no two programs share a module) then the left hand 11 bits of every meaningful system address form at most 100 possible module-selecting combinations. (See Figure 4.) We can

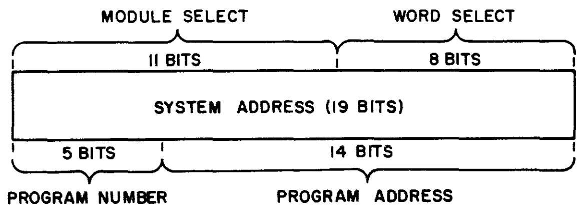


Figure 4. Showing the origin and use of th϶ 19-bit system address.

then use a 100-word associative memory with 11-bit words to map the upper 11 bits of the system address into a module specification. See Figure 5.

The storage allocation function of the executive consists of writing appropriate words into the associative memory, thereby assigning storage modules to programs. Notice that there is no inherent order or adjacency to the storage modules, so that scavenging unused modules for assignment to a new program does not require moving any existing programs.

In a multiprocessor system each processor would have its own associative memory, and conflict resolution would be accomplished by a switch called the Memory Exchange in Figure 6.

The arrangement of Figure 6 meets the three storage problems, as explained below.

1. The "system address" notion, together with the ground rule that no two programs can coexist within a module, insure that a program can access only its assigned modules.

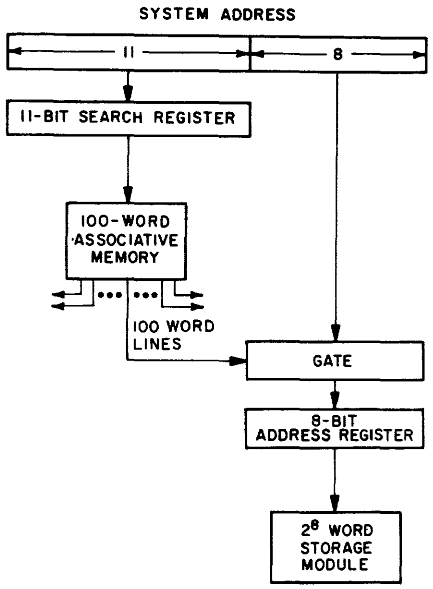


Figure 5. The associative memory selects a storage module from the eleven high-order bits of the system address.

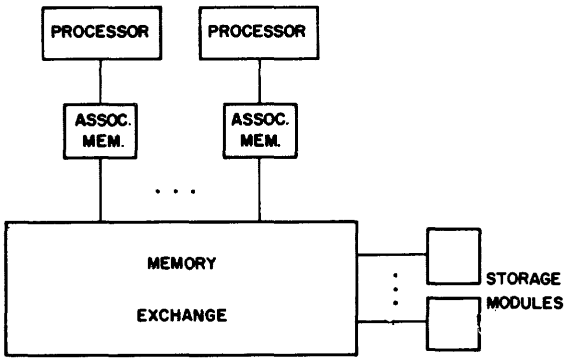


Figure 6. A storage subsystem.

2. The allocation of storage requires no movemet of memory conntents, only the simultaneous writing in all associative memories of as many words as there are modules being allocated.

3. If there is only one problem in the system and it contains many simultaneous references to the same module (as might happen with matrix operations) then this memory system provides no advantage. The other extreme, in which the memory is no bottleneck, is that wherein the system contains many serially coded programs.

A SYSTEM CONFIGURATION

One more major change over the system of Figure 3 remains to be made: the control and main memories will be consolidated.

The control memory has the following properties.

1. It contains logic for queuing state words.
2. Although it is active only when a FORK or JOIN is being executed, there are brief times when it might be very busy.

It should also have the following property.

3. The executive should have the facility to allocate modules from the system "storage pool" to the control and main memories as required. In this case the control memory would belong to program zero, the executive.

The system is made homogenous by isolating the logic functions of control memory into a second kind of processor, the control processor (CP). The control processor has a fixed program, presents the same interface to the memory as an arithmetic processor (AP, formerly called processor), and provides a communication path for state words between AP's and memory. Figure 7 shows the system configuration. The Dispatcher is a switching device for connecting AP's and CP's.

Figure 8 shows the flow of state words. Path A gives the flow of a state word after AP_1 executes a FORK. Path B gives the flow of a state word after AP_2 executes a nonfinal JOIN. The system should accommodate a number of CPs' which will be in balance with the number of AP's.

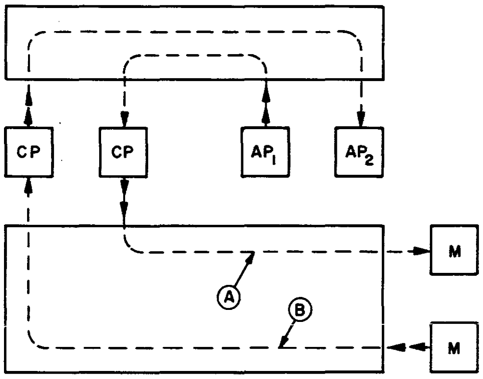


Figure 8. Path of state words.

Finally, an I/O processor (IP) provides a communication channel to the outside world. When an AP decodes an I/O instruction it sends the state word and the decoded instruction down the Dispatcher to the appropriate IP. This frees the AP. When the IP finishes the I/O instruction it sends the state word to control memory or to an AP. Figure 9 shows the final system configuration.

SOME PROGRAMMING CONSIDERATIONS

One of the most intriguing aspects of programming the system developed here arises from the possibility that the same section of code can be executed simultaneously in two parallel paths. This might happen n^2 times in the addition of two $n \times n$ matrices, or it could happen a small number of times in the simultaneous use of the same cosine routine in a tracking calculation. The usefulness of this possibility (indeed, the difficulty of avoiding

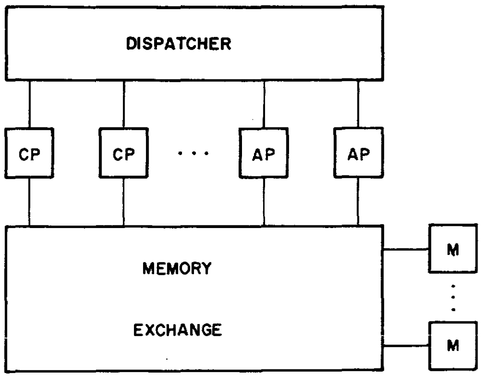


Figure 7. A more homogeneous system configuration.

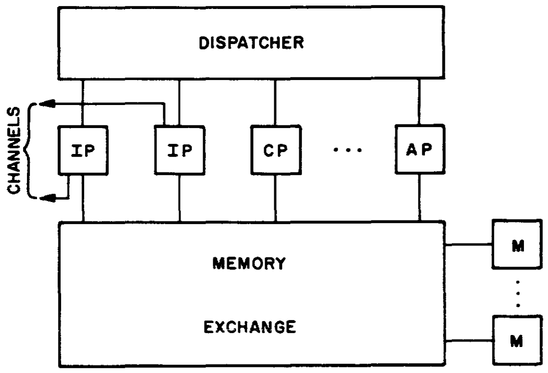


Figure 9. Final system configuration.

it) provides arguments for including index registers in the state word.

Consider the multiple use of the cosine subroutine. The exit address cannot be stored in a fixed memory location; it could be wiped out at any time by another call to the subroutine. A little reflection reveals that if there is any information unique to the subroutine call at the time of entrance to the subroutine it is in the state word. (Consider that the two calls to the subroutine might be the same instruction.) In the more general case, consider the subroutine's use of parameters and temporary storage. Neither can any of these be in fixed memory locations. One answer is to stack parameters (including the exit address) in memory and to use an index register to point to the stack. This is not a very good answer, however, because the stack is not in general last-in-first-out. Another possibility which could bear investigation is the use of control memory for subroutine parameters.

Now consider the n $\times$ n matrix addition. In a FORTRAN expression of this process the DO implies serial repetition of the addition coding. This process could be done in parallel, and so we arrive at the parallel DO instruction whose implementation generates n state words, each with a different value in a specified index register. When the loop is short the use of a junction counter to determine the end of the loop would create a bad traffic jam in the memory; this and other reasons make the parallel DO instruction impractical in spite of its appeal. However, if there are enough instructions in the scope of the loop, it would be justified to use a DO or simply to loop on a FORK instruction.

CONCLUDING REMARKS

Fundamental to the concepts presented here is the principle, not yet commonly accepted, that parallel paths in a program need not bear fixed relationships to the processors of a multiprocessor system executing that program. In many applications, if the number of parallel paths generally exceeds the number of processors, adding a processor will increase the system's effective speed. This fact emphasizes that there are two research objectives whose fulfillment will render the system design presented here a practical improvement over the present state of affairs.

1. A search should be made for parallelism in commonly used algorithms. The effort of such a search would be greatly reduced by the addition of the equivalent of FORK and JOIN to the common publication languages, for example, ALGOL.

2. Memories permitting simultaneous access to any set of words should be developed. As long as memories are slower than processors, simultaneous access is the only alternative to higher memory speed for increasing overall processing rates.

REFERENCES

1. CONWAY, M., Design of a Separable Transition-Diagram Compiler, *Comm. ACM* 6 (July 1963), p. 396.
2. CHEATHAM, T. E., JR., and LEONARD, G. F., An Introduction to the CL–II Programming System, Computer Associates Incorporated Report No. CA-63-7-SD, August 1963.
3. THOMPSON, R., and WILKINSON, J., The D825 Automatic Operating and Scheduling Program, *Proc. SJCC,* 1963, p. 41.
4. DREYFUS, P., Programming on a Concurrent Digital Computer, Notes of University of Michigan 1961 Engineering Summer conference, *Theory of Computing Machine Design.*
5. RICHARDS, P., Parallel Programming, Technical Operations Incorporated Report No. TO–B 60–27, August 1960, p. 4.
6. *Ibid.*, p. 6.
7. BUCHHOLZ, W. (Ed.), Planning a Computer System: Project Stretch, McGraw-Hill, 1962, p. 136.

ANALYSIS OF COMPUTING-LOAD ASSIGNMENT IN A MULTI-PROCESSOR COMPUTER*

Masanao Aoki, Gerald Estrin, and Richard Mandell
Department of Engineering
University of California, Los Angeles

I INTRODUCTION

Many of today's numerical problems are too time-consuming to be done on conventional computers. In some cases it is possible to reduce the overall computation time significantly by assigning part of the computation to one or more computers which work in parallel. If there is any uncertainty or random variation in the time required by either of the processors for completing its share of the computation, one cannot always divide the work between the computers in such a way that they will always finish at approximately the same time. Thus, it is possible for a considerable degradation in performance to occur.

This becomes more serious when a problem requires that a two processor computer system is used in such a way that partially processed information must be passed from one processor to the other to complete computation.

In evaluating computer performances, past analyses dealt mostly with queuing problems with a single server.[1] Aside from a rather straightforward extension of queuing analyses with multiple (and perhaps heterogeneous) servers,[2] systems with multiple processing units have produced an entirely new kind of problem in which the processors work interdependently.

In this paper we analyze a computer system with two processing units for which certain amounts of information move from processor 1 to processor 2 and investigate the effectiveness of an algorithm for work-load assignment which determines the amount of information transfer between the processing units.

As an example, the case of two computers performing Aitken-Neville bivariate interpolation[3] will be considered. The model will, however, be general enough to apply to a number of two-computer problems. The criterion of system performance is taken to be the expected time required to complete a given task.

II THE AITKEN-NEVILLE METHOD

In order to make the terminology of our analysis more concrete, we shall first consider how Aitken-Neville interpolation would be performed by a two-processor computing system.

The basic Aitken-Neville formula is given by

$$I_i^{(j+1)} = I_i^{(j)} + \frac{(I_{i+1}^{(j)} - I_i^{(j)})(x - x_i)}{x_{i+j+1} - x_i}$$

where, for nth order interpolation the $I_i^{(0)}$ $(0 \leq i \leq n)$ are the original table entries and I_0 is the final result of the interpolation. When this is specialized to an equal increment table, the formula becomes

$$I_i^{(j+1)} = I_i^{(j)} + (I_{j+1}^{(j)} - I_i^{(j)})\, M_{i,j} \qquad (1)$$

*The work reported in this paper was supported in part by the Office of Naval Research Contract Nonr 233(52). Reproduction in whole or in part is permitted for any purpose of the U.S. Government.

where

$$M_{i,j} = \frac{x - x_0}{(j+1)\,\Delta x} - \frac{i}{j+1}$$

In performing an nth order bivariate interpolation, we first perform n+1 univariate interpolations employing (1). The final step is to perform an nth order univariate interpolation using the n+1 values, which have just been computed, for the initial table of interpolants. For example, if we wish to perform a fourth order interpolation as shown in Figure 1, we first employ formula 1 to find the value of the function at (x, y_0), (x, y_1), (x, y_2) and (x, y_3). We then interpolate in the y direction over these values to find the value of the function at (x, y). Later the schematic representation for (1) shown in Figure 2 will be of considerable use. The circles in Figure 2 represent the computed values of $I_i^{(j)}$. The rectangles represent the values of $I_i^{(o)}$ which are tabulated in the original table. Each triangle, such as the one in heavy lines, represents a single application of Equation (1).

1. Task Division

In the remainder of this paper we will be considering the properties of a computing system such as the one shown in Figure 3. Two computers will cooperate to share the task of performing bivariate interpolation between them. Each computer will work on a portion of the problem. Intermediate results and original data will be passed between the two computers whenever necessary.

Clearly there are several ways of splitting the computation between the processors. One possible task division is to have each processor work on a separate set of univariate interpolations. For instance, one computer could do the univariate interpolations indicated by the broken lines in Figure 1 and the other could process the interpolations indicated by the solid lines. Alternatively both computers could work on separate portions of the same univariate interpolation. As a third alternative the computers could cooperate in the computation of each of the I_i. The first task division is not flexible enough to take advantage of small differences in speed between the computers. The third alternative requires that a great deal of information be passed between the computers.

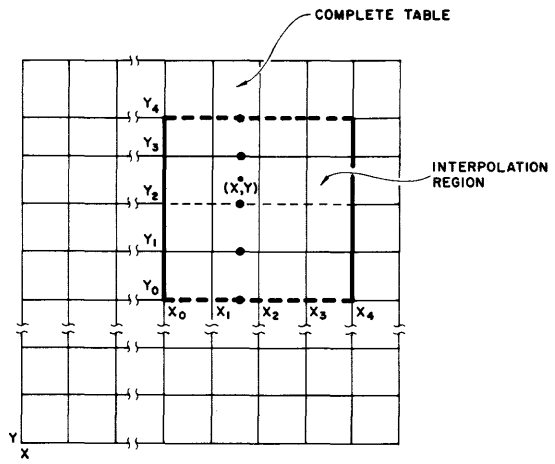


Figure 1. A fourth order interpolation region.

Due to the large number of transfers there is a high probability that one of the computers may be forced to stand idle frequently while waiting for data from the other machine. In this discussion we shall focus our attention on the task division in which each computer is assigned a portion of each univariate interpolation. This type of task division has merit because it takes advantage of small speed differences between the two computers and at the same time does not require too much information to be passed between the computers. The three obvious ways of dividing the work between the machines are shown in Figure 4. The arrows indicate the

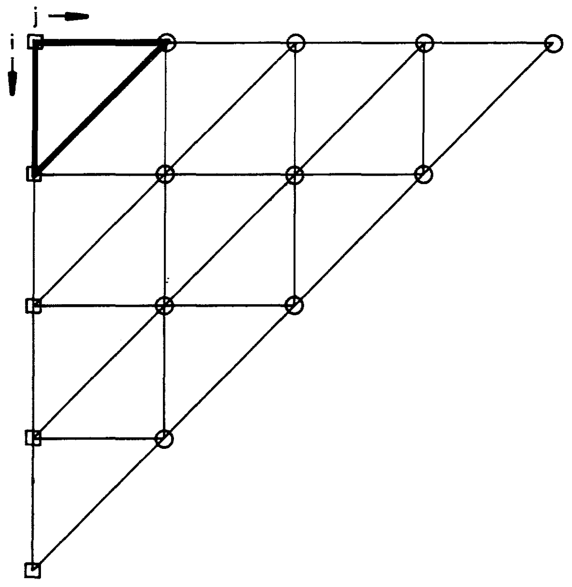


Figure 2. Typical stage in processing of data in an univariate interpolation.

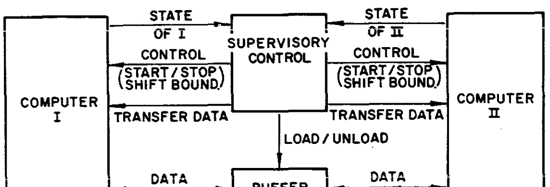


Figure 3. Schematic diagram of a two-processor computer system.

direction of information flow. Of course it is not necessary to have straight boundaries. However, when the boundaries are not straight lines more information must be passed between the computers, or information will have to flow in the opposite direction. Since floating point computation times are data-dependent, it may be advantageous to automatically shift the boundaries shown in Figure 4 in order to equalize the computational load. To accomplish a boundary shift during the execution of a single univariate interpolation with either the horizontal or diagonal boundary shown in Figure 4, it would be necessary to move a portion of the boundary and thus create a boundary which is not straight. The mechanism for accomplishing such a shift would be quite complicated. In the case of the vertical boundary, however, it is possible to move the whole boundary.

III MECHANIZATION

1. *Supervisory Control*

A two-computer system for performing problems such as bivariate interpolation may consist of two general purpose computers or of a general purpose computer and a special purpose computer as in the case of the fixed-plus-variable-structure computer at UCLA.[4]

The system must have, in addition to the two processors, some type of supervisory control

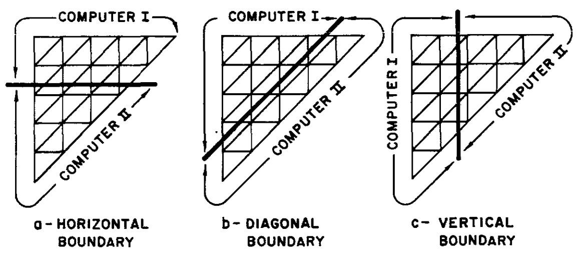


Figure 4. Three possible computing task divisions between the two processors of the system.

which senses the state of the two processors and determines when computation may proceed and when it must be halted. A block diagram of such a two-computer system can be represented by Figure 3.

The amount of hardware in the supervisory control depends upon how much of the processor's time one is willing to devote to supervisory functions. For instance, in order to execute a boundary shift, it is necessary to know the following facts:

(1) that one computer has finished its tasks;
(2) that the computer which is still computing has more than a fixed number of instructions to execute.

Fact 1 is easily obtained by simply having a computer send out a signal when it is halted. The second fact may be determined in several ways. First of all, it is possible to interrupt the computer which is still operating and to have it analyze its progress and render a decision as to whether or not a shift would result in any time savings. Such an interrupt scheme might well nullify any gain achieved by boundary shifting since one computer would have to be interrupted every time its partner comes to a halt. Alternatively it is possible to have the supervisory control analyze the progress of each computer by monitoring the stream of instructions flowing from the memory of each computer or by having each computer send out tally signals at check points in the program. With this continuous record of the state of each machine the supervisory control would be able to make a rapid decision whether or not to execute a boundary shift.

In addition to making decisions about boundary shifting the supervisory control must monitor data transmission and supply control signals and addresses which are necessary to expedite such transfers.

2. *Boundary Shifting*

The initial location of the boundary is found by assuming that all floating point operations will take their average completion time. The boundary is then placed at a point at which the computers will both finish their tasks at the same time. We will limit our discussion to straight boundaries, and thus, it is not always possible to adjust the number of computations assigned to each machine so that the average computation time for each machine will be exactly the same. Thus, in general, as the computation proceeds, one machine will lag further and further behind the other. Actually, neither computer will complete each computation in exactly the average time. Hence, the lag cannot be compensated for by a fixed programmed boundary shift after a fixed number of univariate interpolations has been completed. Instead we shall employ a supervisory control which shall examine the progress of the machine which is still computing each time the other machine reaches the boundary. If the machine which is still computing has more than a certain number of iterations left to perform, some of these iterations will be performed by the machine which has reached the boundary; otherwise the machine that is standing idle will wait until the machine that is still computing has also reached the boundary. The number of iterations which must remain before a boundary shift takes place depends on the location of the boundary, upon the relative speed of the two machines, and upon the order of the interpolation.

IV MATHEMATICAL ANALYSIS

1. *Introduction*

In the analysis that follows we shall formulate methods of answering the following questions:

1) What is the mean computation time with and without boundary shifting?
2) What is the distributon function for the computation time?
3) What is the probability that one of the computers will stand idle while waiting for the other computer to complete its computations?
4) What is the distribution function for the delay experienced by each machine?
5) What is the probability of a boundary shift?

As mentioned above the difficulty in determining how long a series of floating point operations will take is that the computation times are strongly data-dependent. In all of the following analyses it will be assumed that the numbers being processed by the system are uniformly distributed between the limits of the smallest and the largest numbers representable in the processor. Thus, even though for most problems the operands are deterministic, we shall free our analysis from being tied to a particular program with a particular set of operands by assuming the property of randomness.

The analysis will be performed in two steps. First, two simplified models will be considered. Both of these models will be based upon the assumption that the computation times in the two computers may be characterized by continuous probability density functions. The analysis of these two models will point out the important system parameters which influence questions one and two. In the third model it will be assumed that the computation times of the processors may be characterized by discrete probability functions. This assumption is justified particularly when synchronous computers are considered. The probability distributions for the IBM 7090 are available[5] and will be used in an example.

It is convenient to define units of computation for processing unit 1 (PU1) and the processing unit 2 (PU2) as the parts of the computation that are assigned to PU1 and PU2 respectively. PU2 cannot start processing unit i unless PU1 finishes its ith unit of computation. In Figure 4c, the trapezoidal shaped portion is a unit of computation for PU1 and the triangular shaped portion is a unit for PU2.

N such units composes a task of computation. In general problems, if PU1 must process a quantity of information (data words) before PU2 can start its processing, then that amount of processing is called a unit. Thus, one is interested in the time for the n units to go through two processors (servers) in cascade.

The problem, therefore, resembles a stochastic version of the book binding problem[6] except for the possible complications in the present problem which arise from the way information is to be transferred between the two arithmetic units. One of the purposes of the analysis is to see the effect on the total computing time of some dynamic load assignment algorithm. We will consider three models of the system.

2. *Model 1*

In order to obtain some insight into the operation of the computing system, we first consider a simplified analysis. The following roles govern the operation of the model.

1. PU2 never begins its ith unit until PU1 has completed its ith unit.
2. PU1 may not begin its i+1th unit until PU2 completes its ith unit. This rule corresponds to the case in which data generated by PU1 would be destroyed by beginning the new computation before it could be removed for use by PU2.

Let a_i be the random time required to process the ith unit by PU1. Let b_i be the corresponding time needed by PU2.

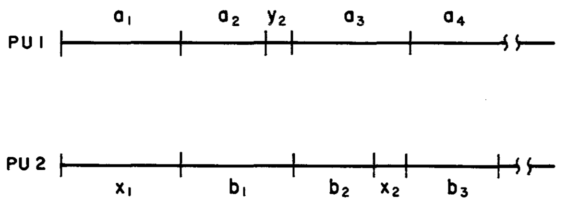


Figure 5. Schematic representation of computing times in Model 1.

The random times x_i denote the waiting time of PU2, before PU1 and PU2 can start on the next unit. Some of them will be zero. Similarly y_{i-1} denotes the waiting time by PU1. Figure 5 shows one such situation schematically

The total time T required to process n units is given by

$$T = a_1 + \sum_{i=1}^{n-1} \text{Max}(a_{i+1}, b_1) + b_n \tag{1}$$

Let a_i be distributed independently and identically with distribution function F(a). Let G(b) be the distribution function for b_i. Let f(a) and g(b) be the respective probability density functions assumed to exist. Later, a discrete probability case will be discussed in connection with computer simulations. Because of the assumptions on a_i's and b_i's the distribution function for $\text{Max}(a_{i+1}, b_i)$ is given by

$$\begin{aligned} M(t) &\overset{\triangle}{=} \Pr(\text{Max}(a_{i+1}, b_i) \leq t) = \Pr(a_{i+1} \leq t, b_i \leq t) \\ &= \Pr(a_i \leq t, b_i \leq t) = \Pr(a_i \leq t)\Pr(b_i \leq t) \\ &= F(t)\,G(t) \end{aligned}$$

The characteristic function of M(t) is denoted by

$$\phi_M(u) = \int_{-\infty}^{\infty} e^{iut}\, dM(t) \tag{2}$$

Therefore, the characteristic function for the random variable T, denoted by ϕ_T, is given by

$$\phi_T(u) = \phi_1(u)[\phi_M(u)]^{n-1}\phi_2(u) \tag{3}$$

where ϕ_1 and ϕ_2 are characteristic functions of a and b respectively.

Assuming that means and variances exist for F, G and M, the mean of T $m_T \overset{\triangle}{=} E(T)$ is given by

$$\begin{aligned} m_T &= m_1 + (n-1)m_M + m_2 \\ \sigma_T^2 &= \sigma_1^2 + \sigma_2^2 + (n-1)\,\sigma_M^2 \end{aligned} \tag{4}$$

where

$$m_1 = E(a),\ \sigma_1^2 = \text{Var}(a)$$

$$m_2 = E(b),\ \sigma_2^2 = \text{Var}(b)$$

and the subscript M refers to the random variable M.

In order to obtain some indications of the dependence of m_T on system parameters, assume that a and b are distributed uniformly between (α_1, α_2) and (β_1, β_2) respectively. Then

$$m_1 \overset{\triangle}{=} E(a) = \frac{\alpha_1 + \alpha_2}{2}, \qquad m_2 \overset{\triangle}{=} E(b) = \frac{\beta_1 + \beta_2}{2}$$

$$\sigma_1^2 \overset{\triangle}{=} \mathrm{Var}(a) = \frac{(\alpha_2 - \alpha_1)^2}{12}, \qquad \sigma_2^2 \overset{\triangle}{=} E(b) = \frac{(\beta_2 - \beta_1)^2}{12}$$

If it is further assumed that $\alpha_1 \leq \beta_1 \leq \alpha_2 \leq \beta_2$, then

$$m_M = \frac{1}{72\sigma_1\sigma_2}\left\{\alpha_2^3 - \beta_1^3 - 3\alpha_1(\beta_2^2 - \beta_1^2) + 3\alpha_2(\beta_2^2 - \alpha_2\beta_1)\right\}$$

The derivation is given in Appendix 1.

If PU1 and PU2 are assumed to be identical, then

$$m_M = \frac{1}{6\sqrt{3}\,\sigma}\left\{2\alpha_2^2 - \alpha_1^2 - \alpha_1\alpha_2\right\} = m\ 1 + \frac{1}{\sqrt{3}}\ \frac{\sigma}{m} \tag{5}$$

Thus, from Equation (3)

$$E(T) = 2m + (n-1)\, m\left(1 + \frac{1}{\sqrt{3}}\frac{\sigma}{m}\right) \tag{6}$$

If a single computer is used to process the same n units, then since the mean time to process a single unit is now given by 2m, the expected time will be 2nm. For large n, the ratio, R of the computer time, therefore, is

$$R \simeq 0.5 + \frac{1}{2\sqrt{3}}\frac{\sigma}{m} \tag{7}$$

TABLE 1

$\frac{\sigma}{m}$	R
0	0.5
0.2	0.53
0.3	0.59
0.5	0.65
1	0.79

Table 1 shows the dependence of R on σ/m. Thus, when there is no randomness involved and when such things as data transfer times between the two computers are ignored, the ratio achieves its theoretical limit of 0.5. The ratio σ/m is usually of the order 0.2-0.5.

To see the effect of possible difference in speeds of PU1 and PU2, E(T) is also computed with the assumption that

$$\beta_1 = \frac{\alpha_1 + \alpha_2}{2},\ \beta_2 = \beta_1 + (\alpha_2 - \alpha_1) \tag{8}$$

Then

$$m_M = m\left(1 + \frac{75}{72}\frac{\sigma}{m}\right) \tag{9}$$

Compared with Equation (5), the time is increased by the factor

$$\left(1 + \frac{75}{72}\frac{\sigma}{m}\right)\Big/\left(1 + \frac{1}{\sqrt{3}}\frac{\sigma}{m}\right).$$

The ratio R for this case is given by

$$R \simeq 0.5 + \frac{75}{144}\frac{\sigma}{m} \tag{10}$$

as indicated in Table 2.

TABLE 2

$\frac{\sigma}{m}$	R
0.2	0.60
0.3	0.66
0.5	0.76

To see the effect when the parameters are such that $\alpha_1 < \beta_1 < \beta_2 < \alpha_2$, m_M and R are computed when

$$\beta_1 = \frac{3\alpha_1 + \alpha_2}{4}$$

and (11)

$$\beta_2 = \frac{\alpha_1 + 3\alpha_2}{4}$$

With these values, $E(a) = E(b)$ and $\mathrm{Var}(a) = 2\,\mathrm{Var}(b)$

$$m_M = m\left(1 + \frac{9\sqrt{3}}{16}\frac{\sigma}{m}\right) \tag{12}$$

and

$$R = 0.5 + \frac{9\sqrt{3}}{32}\frac{\sigma}{m} \tag{13}$$

where σ refers to the random variable b.

By comparing Equations (7), (8) and (11), the parameter values of Equations (11) give smaller mean completion time than those of Equation (8).

Let us now discuss the effect of possible boundary shifts.

Since the boundary shift will occur when the supervisory unit examines the expected time to finish the current computation in PU1 and PU2 at a certain critical time and when it finds that they differ more than a certain threshold value, the overall effect is to make

$$\text{Max}\,(a_i, b_i) \approx a_i \approx b_i$$

Therefore, from Equation (1) the total elapsed time of computation is roughly

$$E(T) = (n + 1)m \tag{14}$$

Therefore, from Equations (6) and (14) the ratio in this case becomes

$$R = \frac{n+1}{2n} \simeq 0.5$$

which is very close to the ideal factor 0.5. Thus, as expected, the boundary shifts tend to reduce R further towards its theoretical limit, 0.5. The effect of boundary shifts is more pronounced for systems composed of two different computers.

The variance is computed for the case of two identical computer systems and is given by

$$\sigma_M = \left\{\frac{\sigma^2}{6}\; 1 + \frac{1}{\sqrt{3}}\left(\frac{m}{\sigma}\right)^3 - 3\left(\frac{m}{\sigma}\right)^2 + \frac{9}{\sqrt{3}}\left(\frac{m}{\sigma}\right)\right\} \tag{15}$$

3. *Model 2*

In this model, it is assumed that the two arithmetic units are connected via a buffer of infinite capacity. Although this assumption is admittedly unrealistic, it is nevertheless a useful one especially if one investigates the number of words in the buffer and its fluctuations to determine the buffer size.[1] The result of this assumption is that PU1 can proceed with the next unit of computation as soon as it finishes with the present unit without being delayed by the condition that the buffer is full. Figure 6 shows one possible situation schematically.

The total elapsed time T is given by

$$T = T_1 + T_2 \tag{1}$$

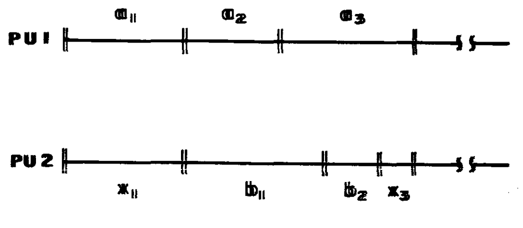


Figure 6. Schematic representation of computing times in Model 2.

where

$$T_1 = \sum_{i=1}^{n} b_i \tag{2}$$

and

$$T_2 = \sum_{i=1}^{n} x_i \tag{3}$$

T_1 gives the total processing time by PU2 and T_2 gives the total idle time of PU2.

It can be shown[6] that

$$\sum_{i=1}^{n} x_i = \underset{1 \leq k \leq n}{\text{Max}} \left(\sum_{i=1}^{k} a_i - \sum_{i=1}^{k-1} b_i \right) \tag{4}$$

As before,

$$\phi_1(u) = \int e^{iau}\, dF(a) \tag{5}$$

and

$$\phi_2(v) = \int e^{ibv}\, dG(b) \tag{6}$$

They can be expanded in series as

$$\phi_1(u) = 1 + iu\, m_1 - \frac{u^2}{2\,!} m_2 + \ldots$$

and

$$\phi_2(u) = 1 + iu\, \mu_1 - \frac{u^2}{2\,!} \mu_2 \ldots$$

Then, because of the identity and independence assumptions, the characteristic function of the random variable T_1 is given by

$$\phi_{T_1}(u) = [\phi_2(u)]^n \tag{7}$$

Therefore, the mean and the variance of T_1 is given by

$$E(T_1) = n\ E(b) = n\mu_1 \qquad (8)$$

where

$$\mu_1 = \int b dG(b)$$

and

$$\begin{aligned} Var(T_1) &= E(T_1 - E(T_1))^2 \\ &= n\ \sigma^2(a) \qquad (9) \end{aligned}$$

where

$$\sigma^2 = Var(b)$$

The mean and the variance of T_2 can be computed as follows:

$$Pr(T_2 < t) = Pr(z_1 < t, z_2 < t, \ldots, z_n < t) \qquad (10)$$

where z_i can be rewritten as

$$\begin{aligned} z_1 &= a_1 \\ z_2 &= a_1 + k_1 \qquad (11) \\ z_n &= a_1 + k_1 + k_2 + \ldots + k_{n-1} \end{aligned}$$

where

$$k_i = a_{i+1} - b_i$$

Note a_i and k_i $1 < i < n-1$, are independent since a_i and b_i are independent among themselves respectively. Note also that a_i and b_i are independent. Also the k_i are identically distributed.

As shown in Appendix 2 the probability density function of k, h(x), is given by

$$h(x) = \int_{-\infty}^{\infty} f(y - x)\ g(y)\ dy \qquad (12)$$

The characteristic function for k is denoted by ϕ_3 and is given by

$$\phi_3(u) = \phi_1(u)\ \phi_2(u) \qquad (13)$$

where the bar denotes the complex conjugation.

From Equation (10),

$$\Psi(t) \overset{\Delta}{=} Pr(T_2 < t) = \int_{-\infty}^{t} f(a_1) da_1 \int_{-\infty}^{t-a_1} h(k_1) dk_1 \ldots \int_{-\infty}^{t-a_1-k_1 \ldots k_{n-2}} h(k_{n-1}) dk_{n-1}$$

$$= \int_{-\infty}^{t} da_1\, f(a_1)\, \Phi(t - a_1) \qquad (14)$$

where

$$\Phi_{n-1}(t - a_1) = \int_{-\infty}^{t-a_1} h(k_1) dk_1 \int_{-\infty}^{t-a_1-k_1} h(k_2) dk_2 \ldots \int_{-\infty}^{t-a_1-k_1 \ldots k_{n-2}} h(k_{n-1}) dk_{n-1} \qquad (15)$$

By taking the Fourier transform, the characteristic function of T_2 is given by

$$F(\psi) = \phi_1(u) \quad F(\Phi_{n-1}) \qquad (16)$$

where

$$\Phi_{n-1}(t - a_1) = \int_{-\infty}^{t-a_1} dk_1 h(k_1) \phi_{n-2}(t - a_1 - k_1) \qquad (17)$$

and where

$$\Phi_{n-2}(x) = \int_{-\infty}^{x} h(k_2) dk_2 \int_{-\infty}^{x-k_2} h(k_3) dk_3 \ldots \int_{-\infty}^{x-k_2 \ldots k_{n-2}} h(k_{n-1}) dk_{n-1}$$

Therefore, from Equations (14) and (15)

$$F(\Phi_{n-1}) = \phi_3(u)\ F(\Phi_{n-2}) \qquad (18)$$

Finally we have

$$\Phi_2(x) = \int_{-\infty}^{x} dk_{n-2} h(k_{n-2}) \int_{-\infty}^{x-k_{n-2}} h(k_{n-1}) dk_{n-1} = \int_{-\infty}^{x} dk_{n-2} h(k_{n-2}) H\ (x - k_{n-2})$$

where H is the distribution function of K. Thus, Equation (14) becomes

$$\begin{aligned} F(\psi &= \phi_1(u)\,[\phi_3(u)]^{n-2}F(H) \\ &= \phi_1(u)\,(\overline{\phi_1(u)})^{n-2}\phi_2{}^{n-1}(u)\;F(H) \end{aligned} \qquad (19)$$

Thus, $Pr(T_2 < t)$ is obtained by taking the inverse Fourier transform of Equation (19). Without specifying the forms of F and G, the formula becomes very intractable.

Since it is rather cumbersome to carry out the integrations, we will not pursue the analytical treatment further but turn our attention to computer evaluation. To do this, we now discuss our third model using discrete probability.

4. *Model 3*

To begin the analysis of the system in its generality we shall assume first that no boundary shifting mechanism is active and that the system operates according to the rule described for Model 1. In order to proceed we need to know the probability distribution for the completion time of the unit assigned to each computer. We may obtain this by the following argument.

Let z_i be the time required to complete the ith instruction in the sequence of instructions used to perform a unit. If the program of performing one unit in one of the computers has M instructions, then the total time required to complete one unit in that computer is

$$T = \sum_{i=1}^{M} z_i \qquad (1)$$

The probability distribution for T is given by the convolution of the distribution functions for the z_i. This convolution is most conveniently computed by means of generating polynomials.[7] If the probability that z_i will be n clock cycles is $P_{i,\,n}$ then the generating polynomial $(P_i(s))$ for the probability distribution of z_i is given by

$$P_i(s) = \sum_{j=o}^{M} P_{i,j}\,s^j;\quad M = \text{the maximum possible number of cycles for the i}^{\text{th}}\text{ operation.} \qquad (2)$$

The average value of z_i is

$$\overline{z_i} = \left[\sum_{j=1}^{M} j\,P_{i,j}\,s^{j-1}\right]_{s=1} = P_i'(1) \qquad (3)$$

where a prime denotes differentiation with respect to s, and the variance of z_i is given by

$$Var(z_j) = P_i''(1) + P_i'(1) - \{P_i'(1)\}^2 \qquad (4)$$

The probability that z_i will have the value n may be obtained by evaluating

$$P_{i,\,n} = \left.\frac{d^nP(s)}{n!\,ds^n}\right|_{s=0} \qquad (5)$$

The generating polynomial for the probability that $z_i \leq r$ is given by

$$Q_i(s) = \{1 - P_i(s)\}/\{(1-s)\} -1 < s < 1 \qquad (6)$$

The generating polynomial for the distribution of the sum of several independent random variables is the product of the generating polynomials for the variables. Thus, T has the generating polynomial $\overline{T}(s)$ given by

$$\overline{T}(s) = \prod_{i=1}^{M} P_i(s) \qquad (7)$$

We may now use these generating polynomials to compute the probability distribution for the delay which the system experiences.

Let $G_m(s) \overset{D}{=}$ the generating polynomial for the probability distribution of the completion time for mth unit assigned to PU2.

$H_m(s) \overset{D}{=}$ the generating polynomial for the cumulative distribution for the mth unit assigned to PU2.

$I_m(s) \overset{D}{=}$ the generating polynomial for the probability distribution for the completion time for mth unit assigned to PU1.

$J_m(s) \overset{D}{=}$ the generating polynomial for the cumulative distribution that may be obtained from $I_m(s)$.

Let $g_{m,n}$, $h_{m,n}$, $i_{m,n}$ and $j_{m,n}$ be the coefficients of s^n in $G_m(s)$, $H_m(s)$, $I_m(s)$ and $J_m(s)$ repectively.

By our previous definition, $g_{m,n}$ is the probability that PU2 will complete its mth unit in exactly n memory cycles, $h_{m,n}$ is the probability that PU2 will complete its mth unit in n or less cycles, and $j_{m,n}$ is the analogous probability for PU1. Similarly $i_{m,n}$ is the probability that PU1 will finish its mth unit in exactly n cycles.

Using these definitions we may compute the probability, r(n), that n memory cycles will elapse between the time PU1 begins the mth unit and the time it begins the (m+1)th unit. There are two ways in which the computation may require n cycles. PU1 may complete its portion of the mth unit and find that PU2 has already completed the (m—1)th unit. Then both computers may proceed immediately. On the other hand, PU1 may complete its portion of the mth unit and find that PU2 has not finished the (m—1)th unit. In this case the interval (n) between the time PU1 begins the mth unit and the time it is ready to begin the (m+1)th unit is the same as the time required for PU2 to complete the (m—1)th interpolation.

The probability that PU2 will finish at the same time as PU1 or before PU1 and that PU1 will require n cycles to complete the mth unit is given by

$$h_{m-1,n}\, i_{m,n} \tag{8a}$$

and the probability that PU1 will finish at the same time PU2 does or before PU2 finishes and that PU2 will require n cycles to complete the (m—1)th unit is given by

$$g_{m-1,n}\, j_{m,n} \tag{8b}$$

Finally

$$r(n) = h_{m-1,n}\, i_{m,n} + g_{m-1,n}\, j_{m,n} - g_{m-1,n}\, i_{m,n} \tag{9}$$

The third term in (9) is the probability that PU1 and PU2 both require n cycles to complete their respective units. The term must be introduced into (9) since both of the first terms include the probability that the two processors will complete their assigned tasks at the same time. Thus, the average time PU1 spends on the mth univariate computation is given by

$$\overline{r} = \sum_{n=1}^{A_m} n(h_{m-1,n}\, i_{m,n} + j_{m,n}\, g_{m-1,n} - g_{m-1,n}\, i_{m,n}) \tag{10}$$

$A_m = \max(A^{(1)}_m, A^{(2)}_m)$

$A^{(1)}_m$ = maximum time PU1 can actually spend performing the computations assigned to it for the mth unit

$A^{(2)}_m$ = the maximum time PU2 actually spends performing the (m — 1)th unit.

The generating polynomial for r(m) is given by

$$R_m(s) = \sum_{i=0}^{A_m} r(i)s_l = \sum_{l=0}^{A_m} (h_{m-1,l}\, i_{m,l} + j_{m,l} - g_{m-1,l}\, i_{m,l})s \tag{11}$$

Since the total time elapsed between the time PU1 begins the bth unit and the time it begins the (c+1)th unit is the sum of the times spent on (c—b+1) units, we may employ (7) and (11) to obtain the generating polynomial (y(s)) for the total time required to complete (c—b+1) units.

$$y_{(c-b+1)} = \prod_{K=b}^{c} R_K(s) \tag{12}$$

The time which PU1 must spend waiting for PU2 on any unit is also of interest to us. The probability that PU1 will have to wait a length of time τ is given by

$$P_m(\tau) = \sum_{n=1}^{A_m} g_{m,n+\tau}\; i_{m,n} \tag{13}$$

where $g_{m,n+}$ is the coefficient of s^{n+} in $g_m\, G_m$
$i_{m,n}$ is the coefficient of s^n in $I_m(s)$.

The average delay for the mth unit is

$$\overline{P}_m = \sum_{\tau} \sum_{n} \tau\, g_{m,n+\tau}\; i_{m,n} \tag{14}$$

Similarly the probability that there will be no delay is given by the probability that PU1 and PU2 will finish their assignment computations at the same clock period.

$$D = P_m(0) = \sum_{n=1}^{A_m} g_{m,n}\, i_{m,n} \tag{15}$$

We may use (14) to compute the average delay that PU1 will experience in a complete bivariate interpolation. This is given by

$$P = \sum_{m=1}^{m+1} P_m = \sum_{m=1}^{m+1} \sum_{\tau} \sum_{n} \tau \; g_{m,\, n+\tau} \; i_{m,\, n} \qquad (16)$$

We now turn to an investigation of what will happen if boundary shifting is allowed to take place. The boundary shifting will be governed by the following rules:

1. Only whole boundary shifts will be allowed.
2. A boundary shift will occur only if PU2 lags behind PU1 by a_r iterations or PU1 lags behind PU2 by a_l iterations.
3. After each boundary shift the boundary returns to its original value for the next unit.
4. The boundary may be shifted one position left or right.

Under these conditions we can evaluate the distribution function $\omega(T)$ of the time required for the total computation. However, we need to modify our notation somewhat.

Let $L_m(s)$ = the generating polynomial for the completion time of m iterations in PU2

$M_m(s)$ = the generating polynomial for the cumulative distribution associated with L_n

and let $N_n(s)$ and $Q_m(s)$ be the corresponding generating functions for PU1. x iterations are assigned to PU1 and y iterations are assigned to PU2. The probability of a left boundary shift is then the probability that when PU2 finishes the (m—1)th unit PU1 has not yet finished a_l iterations of the mth unit. This probability is given by

$$P_l = \sum_{T=\infty}^{\infty} (1 - q_{a_l,\, T})(l_{y,\, T}) \qquad (17)$$

where ${}_{y,\,T}$ and $q_{a_l,\,T}$ are the coefficients of s^T in $L_y(s)$ and $Q_{a_l}(s)$ respectively.

Similarly the probability of a right shift is

$$P_r = \sum_{T=\infty}^{\infty} (1 - m_{a_r,\, T})(n_{x,\, T}) \qquad (18)$$

We may now obtain $\omega(t)$. There are three mutually exclusive events which we must account for. These are 1) the case in which a right boundary shift occurs, 2) the case in which no boundary shift occurs, and 3) the case in which a left boundary shift occurs. The probability that no boundary shift occurs and that the computation requires T cycles is given by (19). If PU1 finishes before PU2 and PU2 reaches the threshold a_r before PU1 completes x iterations, no boundary shift occurs and the total time required for computing the unit is given by the time required by PU1 to finish its portion of the computations. The probability that this time will be T is given by

$$\sum_{t_2=\infty}^{T} n_{x,\, t_2} \sum_{t_1=\infty}^{t_2} l_{a_r,\, t_1} \, l_{y-a_r,\, T-t_1} \qquad (19)$$

Similarly the probability that the computation will require T cycles, that PU2 will finish first and that no boundary shift will occur is given by

$$\sum_{t_2=\infty}^{T} {}^{l}y,\, t_2 \sum_{t_1=\infty}^{t_2} {}^{n}{}_{a_r},\, t_1^{n} x - a_l \,,\, T - t_1 \qquad (20)$$

The terms that account for left and right boundary shifting may be arrived at by similar reasoning. With these terms included, the equation for the probability that a unit will require T cycles for completion is given by Equation (21).

$$\begin{aligned}
\omega(T) = & \sum_{t_1=\infty}^{T-1} \sum_{t_2=t_1}^{T-1} n_{x,\, t_1} n_{x_r,\, t_2-t_1} \, l_{a_r,\, T} + \sum_{t_2=\infty}^{T} \sum_{t_1=t_{2+1}}^{T-1} n_{x,\, t_2} n_{x_r-\, x} l_{a_r,\, t_1} \\
& + \sum_{t_2=\infty}^{T} \sum_{t_1=\infty}^{t_2} n_{x,\, t_2} l_{a_r,\, t_1} l_{y-a_r,\, T-t_{1_2}} + \sum_{t_2=\infty}^{T} \sum_{t_1=\infty}^{2} n_{a_l,\, t_1} n_{x-a_l,\, T-t_1} l_{y,\, t_2} \\
& + \sum_{t_1=\infty}^{T-1} \sum_{t_2=t_1}^{T-1} l_{y,\, t_1} l_{y_l,\, t_2-t_1} n_{a_l,\, T} + \sum_{t_2=\infty}^{T} \sum_{t_1=t_{2+1}}^{T-1} n_{a_l,\, t_1} l_{y,\, t_2} l_{a_l,\, T-t_2}
\end{aligned} \qquad (21)$$

The distribution function for an nth order bivariate interpolation can be obtained by convolving n+1 expressions like Equation (21) by means of generating polynomials.

V SPECIAL PURPOSE COMPUTER

In the foregoing discussion we have considered the possibility of using two general purpose computers and have shown the degree to which it is possible to approach the theoretical maximum speed improvement of 50%. In some applications, such operations as interpolation occur often enough and are time consuming enough to make it practical to employ a special purpose computer for one of the processors in the system. A block diagram of such a computer is shown in Figure 7.

The arithmetic unit is relatively slow since it must perform a series of floating point operations. Thus, it is essential to employ techniques which free the arithmetic unit from waiting for operands from memory. A sequence of wired-in instructions causes operands to be accessed while the arithmetic unit is computing. The operands are held in the data registers ready for immediate access by the arithmetic unit. The address generator is really an index arithmetic unit which generates the addresses of all operands that must be supplied to the arithmetic unit. The initial addresses required by the address generator are supplied by the general purpose computer under the direction of the supervisory control. The final element of the system is the output buffer which serves as a sink for results from the arithmetic unit and thus frees the arithmetic unit from having to wait for the availability of memory.

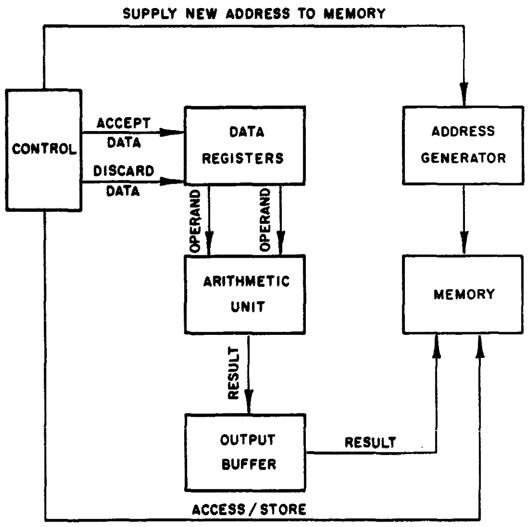


Figure 7. Block diagram of a special purpose computer in a two-processor computer system.

Since the accessing and storage of operands is done in parallel with computation, and since the program is wired-in, the special purpose computer can perform the inner loop of the interpolation problem much faster than a general purpose computer. When such a special purpose computer is teamed with a 7090, the composite system operating as we have described earlier can perform the bivariate interpolation in from (33-55)% of the time required by a 7090 alone. The exact savings depend upon the order of the interpolation.

VI EXAMPLE

As an example of the use of equations 7-15 the time required for executing a 5th order bivariate interpolation on a system consisting of two IBM 7090 computers was considered. A straight boundary case and a crooked boundary case were investigated. The parameters for the two cases are given in Table 3 and the results of the study are given in Table 4. Only the inner loop of the interpolation program was considered. This loop consisted of three float instructions (FSB, FMP, and FAD) and four fixed time instructions (CLA, XCA, STO and TIX). The results given in Table 4 apply to each univariate interpolation separately. The averages for a bivariate interpolation may be obtained by multiplying the averages in Table 4 by 6.

TABLE 3 System Parameters for 5th Order Bivariate Interpolation

Parameter	Straight Boundary	Crooked Boundary
x	9	8
y	6	5
a_l	5	3
a_r	3	6

In the straight boundary case the probability of a boundary shift is negligible because the

TABLE 4 Computation and Delay Times for 5th Order Bivariate Interpolation *

	Straight Boundary	Crooked Boundary
Average Delay for PU1	0	27 cycles
Average Delay for PU2	82 cycles	0.02 cycles
Probability of PU1 Delay	0	.991
Probability of PU2 Delay	.9999	.007
Variance of Distribution of PU1 Delay	0 cycles ^{2}	189 cycles^2
Variance of Distribution of PU2 Delay	304 cycles ^{2}	0.22 cycles^2
Average Computation Time	251 cycles	223 cycles
Variance of Computation Time	586 cycles ^{2}	519 cycles^2
Probability of Right Boundary Shift	.002	0
Probability of Left Boundary Shift	0	0.482

* Table entries apply to each individual univariate interpolation and not to the bivariate interpolation as a whole.

threshold values (a_l and a_r) are so much smaller than x and y. In the crooked boundary case the thresholds for boundary shifting were moved to within one iteration of the boundaries. Consequently the probability of boundary shifting during any given univariate interpolation was increased considerably. It is also to be noted that in both cases it was practically impossible for one or the other of the processors (i.e. PU1 in the straight boundary case) to experience a delay. This is due to the fact that it is impossible to assign the same number of iterations to each processor because the number of iterations is odd. Thus the processor with fewer iterations to do has a very high probability of experiencing a delay.

Additional studies indicated that the expected delay and the probability of a boundary shift are larger for computer systems in which the variance of the computation time distributions is larger than the variance for the IBM 7090 system.

VIII CONCLUSIONS

The problem of load assignment in a two-processor computing system has been studied with the aid of three models. By means of one model, assuming two identical processors with uniformly distributed computation times, it was shown that the ratio of computing time in a one-processor system to that in a two-processor system is a function of the ratio of the standard deviation of the computing time distribution to the expected computing time for the single computer.

An analysis of the first two models also indicated the effectiveness of dynamic load assignment (boundary shifting) superimposed upon the static load assignment.

An example based upon a third model showed that the effectiveness of dynamic load assignment is strongly dependent upon the original static assignment.

APPENDIX 1

For the assumed distribution functions

$$dM(t) = \begin{cases} 0 & ; t < \beta_1 \\ \dfrac{(t-\beta_1) + (t+\alpha_1)}{(\alpha_2-\alpha_1)(\beta_2-\beta_1)} & ; \beta_1 < t < \alpha_2 \\ \dfrac{1}{\beta_2-\beta_1} & ; \alpha_2 < t < \beta_2 \\ 0 & ; t < \beta_2 < t \end{cases}$$

the mean therefore, is given by

$$\int_{\beta_1}^{\alpha_2} \frac{2t^2-(\alpha_1+\beta_1)t}{(\alpha_2-\alpha_1)(\beta_2-\beta_1)}\,dt + \frac{1}{\beta_2-\beta_1}\int_{\alpha_2}^{\beta_2} t\,dt$$

$$= \frac{1}{6(\alpha_2-\alpha_1)(\beta_2-\beta_1)}\{\alpha_2^3-\beta_1^3-3\alpha_1(\beta_2^3-\beta_1^2)+3\alpha_2(\beta_2^2-\alpha_1\beta_1^2)\}$$

APPENDIX 2

Let x and y be two independent random variables with distribution functions H(x) and K(y). The distribution function for the difference $z=x-y$ is given by

$$dF(z) = \int_{-\infty}^{\infty} w(x-z, x)\,dx$$

where

$$w(x, y) = H'(x)K'(y)$$

where primes indicate the derivatives with respect to the arguments.

REFERENCES

1. P. E. BOUDREAN and M. KAC, "Analysis of a Basic Queuing Problem Arising in Computer Systems", *IBM J. of Research and Development, 5* 132-140, April 1961.
2. H. GUMBEL, "Waiting Lines with Heterogeneous Servers", *Op. Res. 8* 504-511, 1960.
3. MILNE, W.E., W. ARNTZEN, N. REYNOLDS, and J. WHEELOCK, *Mathematics for Digital Computers, Volume I, Multivariate Interpolation.* WADC Tech. Rept. 57–556, Wright Patterson Air Force Base, Ohio, United States Air Force, Feb. 1958.
4. ESTRIN, G., "Organization of Computer Systems—The Fixed-Plus-Variable Structure Computer", *Proceedings of the Western Joint Computer Conference,* New York, New York: The Institute of Radio Engineers, May 1960, pp. 33-40.
5. MANDELL, RICHARD, "A Study of Some Methods of Performing Aitken-Neville Bivariate Interpolation with a Fixed-Plus-Variable Structure Computer", Unpublished M. S. Thesis, Library, University of Calif., Los Angeles 24, California.
6. R. BELLMAN, Scheduling Theory, *J.S.I.A.M.* 4 168-205, Sept. 1956.
7. W. FELLER, *An Introduction to Probability Theory and its Applications,* J. Wiley and Sons, New York, 1957.
8. B. BUSSELL, "Properties of a Variable Structure Computer System in the Solution of Parabolic Partial Differential Equations", Department of Engineering, University of California, Los Angeles, Report No. 62-46, Sept. 1962, pp. 112-113.

A DISCRIMINANT METHOD FOR AUTOMATICALLY CLASSIFYING DOCUMENTS

John H. Williams, Jr.
Washington Systems Center
International Business Machines Corporation
Federal Systems Division
Bethesda 14, Maryland

1. INTRODUCTION

This paper discusses a continuing effort within IBM devoted to developing and testing statistical techniques for automatically classifying documents. The work has progressed from a computer evaluation, Meadow[7] (1962) of the classification equations proposed by Edmondson and Willys[4], (1961) through analyses of documents from the fields of psychology, law, computers and international relations, to a recent classification experiment to test our discriminate hypothesis.

The objective of an indexing technique is to find a set of keywords to represent a document; whereas, the objective of a classification technique is to classify a document into one or more subject headings. A further extension is to connect related subject headings into a tree structure. Classification of incoming documents is then performed at each level of the structure.

Several statistical approaches to indexing and classification have been reported on. One of the most widely known is Luhn's[5]. Luhn's hypothesis was that the most frequently occurring uncommon words are the best keywords. He arbitrarily eliminated a set of common words as well as words occurring only once or twice. This method resulted in indexing a document only by the words occurring within each individual document. Maron[6], introduced the use of probability techniques to classify documents and performed an experiment using 405 computer abstracts and 32 categories. The classification decision was based on the probability of occurrence of 90 keywords. These keywords are selected manually from the corpus. Maron included some excellent comments on methods of improving statistical techniques of classification. Borko[3] has conducted several experiments using a factor analysis technique on 90 selected keywords, with increasing success. Rather than assume a classification structure, Borko attempted to construct a classification system on the basis of a reference set of documents. Then using the established set of categories and 90 keywords be classified the incoming documents. Baker[1] proposed a method based on latent class analysis. Even though the method is highly dependent on the selection of appropriate keywords, Baker did not offer any effective solution to the word selection problem. Aside from these few experiments, there is a definite lack of sufficient empirical evidence to support the various approaches proposed by other investigators.

The purpose of the present study has been to examine alternative techniques and accumulate empirical evidence with which the techniques can be evaluated. Although the statisti-

cal approach is readily acceptable intuitively, it cannot be proved as easily. When statistical analysis is applied to actual documents, many intuitive assumptions are shown to be false. The study has revealed many parameters affecting the automatic classification of documents. A few of the problems that arise immediately are: The non-normal distribution of word frequencies; the fact that many documents cannot be categorized into mutually exclusive classes; the varying depth of detail in documents; the homogeneity of subject headings, and the imprecise definition of word populations. One of the practical problems that develops when applying some of the rigorous statistical methods is in the inversion of large matrices. In certain methods the order of the matrix will equal the number of different word types in the population, which is usually in the thousands. Therefore, while investigating rigorous statistical techniques, we have also been testing some empirical expressions.

The discriminant method consists of starting with a hierarchical classification structure and a small set of reference documents previously classified into each category by human indexers. All the words are counted in each of the reference documents, and theoretical frequencies for each word type are computed for each category. The most significant words in each group of categories are selected statistically. Each subject can now be represented by the theoretical frequencies of its most significant words. .With the theoretical frequencies established, an incoming document can then be classified by comparing the observed frequencies of each word type in the document with the set of frequencies representing each subject. This statistic, the Relevance Value, is computed for the document with respect to each category, thus providing for documents that are relevant to more than one category.

2. DATA BASE

For the experiment 400 computer abstracts prepared and published by Cambridge Communications Corporation * were selected. Each of the abstracts was classified by CCC in their normal operation. Thus, the data used in the experiment was classified by professional indexers for the purpose of actual retrieval. The entire CCC classification structure consists of fifteen categories at the major level. Each of these major categories is divided into 10 subcategories. For the actual experiment a truncated structure was generated based on the criterion that each subcategory must comprise at least 20 documents. Figure 1 shows the experimental structure of the computer categories that were used. Each of the major categories was subdivided into five minor categories. 300 of the 400 abstracts were used as reference documents, and were equally divided among the twenty subcategories. The remaining 100 were used as the test documents.

The objective of the experiment was to classify the 100 test documents into their correct categories.

For the experiment a word was defined to be a string of alphanumeric characters separated by two spaces. Every legal alphanumeric character was keypunched, except those appearing in equations. For each equation XXX was punched. Suffix, prefix, plural and spelling corrector programs were not used. Each word was truncated at 6 characters because of the 36-bit word length in the IBM 709 computer. The word counting program converted each numeric and punctuation symbol to a space. Hyphenated words were counted as two distinct words. The author's name and document title were excluded from the word count.

3. SELECTION OF SIGNIFICANT WORDS

The key problems in all statistical approaches to classification and indexing center on the selection of the set of most significant words.

Should such a set contain all the words that occur in a document or all the words that occur in the reference vocabulary for the various categories? If not, what techniques can be used to identify and delete insignificant words? In the most frequently used technique Luhn[5] an arbitrary set of common words is deleted. In the present approach a statistical technique is developed for identifying the insignificant

* These abstracts are taken from the same set of data used by Maron and Borko. *Computer Abstracts on Cards*, Cambridge Communications, Croporation, Cambridge, Massachusetts.

Experimental No. *Major Level*	*Minor Level*	*CCC No.*	
810			
Programming	811	5.0	General; 5.1 Principles; 5.6 Multi-Problem
	812	5.2	Machine Programming
	813	5.3	Programming Languages
	814	5.4	Automatic Programming
	815	5.9	Non-Math Algorithms
820			
Theoretical Design	821	1.2	Switching Functions
	822	1.3	Combinational Networks
	823	1.8	Sequential Networks
	824	2.3	Digital Arithmetic
	825	2.5	Reliability
830			
Mathematics, Statistics, etc.	831	8.4	Algebra and Analysis
	832	8.6	Calculus and Differential Equations
	833	9.2	Statistics
	834	9.5	Information Theory
	835	9.7	Communication Systems
840			
Hardware Design	841	3.1	Electromechanical and Linear Components
	842	3.3	Semiconductor Devices
	843	3.6	Nonlinear Magnetic Devices
	844	3.7	Cryogenic Devices
	845	3.9	Sequential and Waveforming Circuits

Figure 1. Experimental Computer Classification Structure.

words. Because words change in their discrimination power with the context, it is necessary to compute a discriminant coefficient at each level and within each group. To illustrate how that coefficient varies from level to level, one might consider this experiment on computer abstracts part of a larger experiment in which a decision must be made at the next higher level to determine which incoming documents pertain to psychology, law, or computers. At that higher level the word "computer" will have a high coefficient whereas on the original level the word "computer" has the same frequency of occurrence in all categories and therefore, it is not a good discriminator on that level.

The expression used for a discriminant coefficient for the ith word in a group is

$$\lambda_i = \sum_{j}^{n} \frac{(p_{ij} - \bar{p}_{ij})^2}{\bar{p}_{ij}}$$

where:

$p_{ij} = f_{ij} \Big/ \sum_{i}^{m} f_{ij}$ the relative frequency of the ith word in the jth category.

and $\bar{p}_{\cdot j} = \frac{1}{n} \sum_{j}^{n} p_{ij}$ the mean relative frequency per category of the ith word.

Figure 2 shows a few words from group 810 and how their coefficients vary. Figure 3 demonstrates the fact that context is an important factor in determining the significance of a word. The most discriminating words at the major level and their discriminant rank at the minor level are shown. "Programming" was the best discriminator at the major level, but it was sixth on the minor level. Similarly, the word "language" was third at the major level, but had a rank greater than 30 at the minor level.

Word	*Discriminant Coefficient*	*Total Word Frequency (Per Category)* 811	812	813	814	815
EFFICI	.0004	6	8	4	6	5
ESTIMA	.0100	0	0	0	0	6
CHARAC	.0407	2	37	0	6	3
SORTIN	.0926	2	0	0	0	60

Figure 2. Comparison of significant Values of the Discriminant Coefficient.

Programming
810

Word	*Discr. Coeff.*	*Lower Level Rank*
PROGRA	.0435	6
SORTIN	.0171	1
LANGUA	.0164	>30
ADDRES	.0108	>30

Theoretical Design
820

Word	*Discr. Coeff.*	*Lower Level Rank*
NETWOR	.0164	1
SYNTHE	.0110	22
FUNCTI	.0093	15
BOOLEA	.0092	4

Math, Stat, Etc.
830

Word	*Discr. Coeff.*	*Lower Level Rank*
EQUATI	.0174	3
ERROR	.0161	14
CODES	.0103	1
SOLUTI	.0095	>30

Hardware Design
840

Word	*Discr. Coeff.*	*Lower Level Rank*
CIRCUI	.0209	6
TRANSI	.0130	13
MAGNET	.0124	16
PULSE	.0093	>30

Figure 3. The Four Most Discriminating Words at the Major Level.

Once the discriminant coefficient has been computed, it is used to set up discriminant thresholds determining which words will be used in the classification equation and to assign a weighting factor to the word itself.

4. CLASSIFICATION EQUATION

The computer program classifies documents by comparing the observed with the theoretical word frequencies and computing a Relevance Value (RV) for each document with respect to each category. The RV equation is:

$$RV_j = 1. - \left[\frac{.01}{m}\sum_i^m\left(\lambda_i\frac{(p_{io}-p_{ij}^*)^2}{p_{ij}^*}\right)\right]$$

where p_{io} is the relative observed frequency in the document, p_{ij}^* is the relative theoretical frequency of the ith word in the jth category, after transformation to document size, and m is the number of word types in the group. Figure 4 shows a summary of the number of correctly classified documents in our experiment. Of the 100 test documents initially selected, 17 were not completely indexed within the experimental structure. Therefore, complete results are available on only 83 of the original 100 documents. Type I documents were classified into only one category at both the major and minor levels. Type II documents were classified into one category at the major level, and two categories at the minor level.

Document Type	*No.*	*Major Level*	*Minor Level*
I	63	.78	.64
II	20	.95	.60/.75

Figure 4. Percentage of Correctly Classified Documents.

An additional series of experiments were conducted to investigate the value of the discriminant coefficient in the classification process. Figures 5a through 5c illustrate how the different uses of the discriminant coefficient may facilitate the classification decision by causing a greater dispersion of RW's. The increase in RW can be observed for three of the one hundred documents tested. The RW's of documents 36–38 which should be classified in category 823 are shown in Figure 5.

In the first experiment, all words were used and were given equal weight by setting $\lambda_i = 1.0$. The results shown in Figure 5a indicate that the highest RW is in Category 823, but little confidence could be attached to an assignment, because of the lack of dispersion among the categories. In the second experiment, a threshold of approximately .0100 was used to eliminate insignificant words. Again equal weight was given to each of the significant words in the summation. Figure 5b indicates some dispersion can now be expected. Finally, the computed value of λ_i for each significant word was employed in the classification equation. The resulting dispersion is shown in Figure 5c. In all these experiments essentially the same number of correctly classified documents was maintained while the confidence with which the assignment can be made was increased.

Two of the major reasons for misclassification were heterogeneous categories such as in group 830 and the small sample sizes. Since these results were obtained on only 15 reference documents per category, it is felt improvement could easily be achieved by increasing the number of reference documents. This increase will not only provide a larger reference set per category but will also provide for inclusion of more categories from the original structure and minimize the heterogeneity factor. Results of previous experiments on other data had shown that lack of homogeneity in a category could adversely affect the classification ability of the system. This conclusion was again borne out in the CCC experiment. For example, category 825—Reliability is unrelated to the four other categories in 820. When attempting the decision at the major level, four of the five test documents belonging to Reliability were misclassified. Whereas, at the minor level, four of the five were correctly classified. Apparently, at the major level misclassification was due to the fact that of the 75 reference documents only 15 pertained to Reliability and hence they had very little influence in representing the total category. Whereas, at the minor level,

Doc. No.	*CATEGORY*				
	821	*822*	*823*	*824*	*824*
36	.964	.963	.980	.959	.956
37	.961	.978	.981	.971	.970
38	.963	.969	.972	.961	.962
		5a. First Experiment—Use all Words			
36	.679	.648	.900	.684	.601
37	.661	.895	.906	.828	.828
38	.671	.761	.888	.847	.841
	5b. Second Experiment—Use of Threshold to Eliminate Insignificant Words				
36	0.0	0.0	.858	0.0	0.0
37	0.0	.650	.682	0.0	0.0
38	0.0	.069	.863	.473	.307
	5c. Third Experiment—Use of both Threshold and Weighting Factor				

Figure 5. Effect of Discriminant Coefficient on Relevance Values.

reliability was very different from the other categories, and hence discrimination was easy.

5. SUMMARY

Subjective discussions naturally arise concerning the definition of a category. Our approach considers this problem. In an operational situation the categories are determined by the total subject content of the documents rather than an arbitrary phrase of word labels. A group of documents is selected by the user as representative of what should be contained in a category. The words in these documents then become the reference library. Each document would then be considered in the same manner as a new document. It would be entered into the classification program and a Relevance Value (RV) computed for it. Those documents having a RV outside the standard deviation limits would be returned to the user for a re-evaluation. Thus the reference library will be as homogeneous as desired.

In addition to specifying the content of each category, the user also specifies the relationship in which the subjects are to be connected. The ability to generate various class structures from the same set of original documents eliminates the fixed viewpoint of certain classification schemes. Users who require differing aspects of a complex set of information may each determine the structure most suited to his own needs. Incoming documents can then be classified in accordance with a number of individual reference vocabularies. Since the number of nodes and branches in the hierarchical lattice is flexible, an individual or group specializing in a single area can specify a structure with multiple levels of details in one area, and only one or two levels in other areas.

A classification structure in connection with the discriminant method can also be used to advantage in the query phase of an information system. The user is provided a link between his own concept of a subject with the structure's concept of a subject. In querying the system he has the option of requesting all documents within a subject heading or writing a narrative description of his request. The narrative description would be input as though it were an incoming document to be classified. Relevance Values would then be output with respect to each category in the system. He can now choose the categories in which a search is to be performed on the basis of the highest Relevance Values.

REFERENCES

1. BAKER, FRANK B. "Information Retrieval Based Upon Latent Class Analysis," *Journal of ACM*, Vol. 9, (1962), No. 4, pp. 512–521.
2. BAXENDALE, P. B., "Machine-Made Index for Technical Literature—An Experiment," *IBM J. R. & D.* Vol. 2, (1958), No. 4, pp. 354–361.
3. BORKO, HAROLD, and BERNICK, MYRNA. "Automatic Document Classification," *Journal of ACM*, Vol. 10 (1963), No. 2, pp. 151–162.
4. EDMUNDSON, H. P. and WYLLYS, R. E. "Automatic Abstracting and Indexing—Survey and Recommendations," *Communications of the Association for Computer Machinery*, Vol. 4, (1961), No. 5.
5. LUHN, H. P. "A Statistical Approach to Mechanized Encoding and Searching of Literary Information," *IBM J. R. & D.* Vol. 1 (1957), pp. 309–317.
6. MARON, M. E. "Automatic indexing: an experimental inquiry," *Journal of ACM*, Vol. 8 (1961), No. 3, pp. 404–417.
7. MEADOW, HARRIET R. *Statistical Analysis and Classification of Documents*, IRAD Task No. 0353, FSD, IBM, Rockville, Maryland, 1962.

THE DIRECT ACCESS SEARCH SYSTEM

I. A. Warheit
International Business Machines Corporation
Advanced Systems Development Division Laboratory
San Jose, California

The development of mechanized information retrieval systems has been rapid. Successful operating systems have been established and new ones are being set up almost every day. Techniques not practical with manual systems have been developed, notably in the searching of deep indexes.

Most of the mechanized systems installed have been for relatively small files: little has been heard about the large files, the large libraries and the massive indexes. Only a very few sharp critics like Ralph Shaw[1] have noted this curious anomaly and have wondered about the capacity, speed and general economics of mechanized systems if applied to real library-size files. Strangely enough, these critics have emphasized peripheral things such as printing fonts, and have considered the poorer systems which are nothing more than mechanized manual methods—or what Jessica Melton has called "a rather awkward, expensive, and inaccessible analog of the present card catalog or printed index."[2] This does not face up to the real problem, which is, can we process large files in a reasonable length of time and at reasonable cost? The problem is extremely serious because many systems being started today will work well only for today's small files. If computer capabilities do not increase far beyond anything available today, such systems will certainly break down as the number of records in them increases.

In searching for an answer to the problem of handling large files, one finds, very frequently, a strong reliance on future engineering developments. What used to take a millisecond now takes a microsecond and tomorrow will take only a nanosecond. Faster switching devices and faster input-output devices are coming. Recently announced are such things as the IBM 1440, whose scan-read capability permits the processor to examine records on the fly. Optimistic forecasts describe associative memories or multiple comparators, unlimited internal memory capacities, and parallel interrogation of a whole file.[3] In addition to these hoped-for engineering developments, some pin their hopes on very elaborate classification schemes or elaborate machine programs based on list processing or chaining addresses.[4]

In other words, many people push the real problems aside, convinced that the state of the art will improve with sufficient speed to produce a solution, preferably mechanical, before their present system breaks down.

Actually, of course, in view of past performance, it is not unreasonable to expect technological developments to solve many of our problems. But such reliance is extremely hazardous if it becomes an excuse for ignoring the necessity for good system design, based on our present knowledge and on the present state of the art. The Direct Access Search System is an example of system design applied to the problems of very large files which have extensive indexes. Without elaborate and expensive machine programs or classification schemes, without exotic hardware, we can handle such large files and still retain all the capabilities we now have in manipulating small files.

Most mechanized information retrieval systems identify records stored by descriptors or keywords, though a number use special codes based on classification schemes. A descriptor, as used here, is a structured or normalized or controlled term, as distinguished from an unstructured keyword taken directly from the text. The file index in most systems is arranged either as a so-called conventional file or as an inverted file. In the former, the item—i.e., the complete bibliographic entry—usually representing a document, is followed by all the terms or descriptors describing that document. In the inverted file, each term or descriptor is followed by the items (documents) or their addresses.[5]

The conventionally arranged index with the descriptors by title must be searched serially or sequentially. That is, starting with the first entry, the whole file must be read through and each record examined to determine which entries satisfy all search criteria. For the inverted file search, only the terms of the search question need be read and the item addresses under the several descriptors compared.

The disadvantage of the serial read is that it is strictly a brute-force approach. Even with the fastest and most efficient computers, it is uneconomical to read a very large file. The serial read has this advantage, however: since every record is read, one can "browse" through the file and look for things which cannot be exactly specified, and in the process, significant relationships and associations may be observed.[6, 7]

The inverted file search is much more efficient, especially with random access, i.e., direct access to the term being sought. Most computer-based inverted files, however, are stored on reels of magnetic tape, and are therefore operated in a serial mode. The inverted file index, thus operated, is usually no more efficient than the conventional index, except for the fact that one may find all terms sought—and therefore conclude the search—before the whole file is read.

Until very recently, random access devices were so expensive per unit of storage and so limited in capacity as to be practical for storing only the most active information. Information less valuable and less frequently sought was usually kept on tape or cards. Today, with storage costs coming down and capacities rapidly increasing, it soon will be economical to store complete files in random access units, and it will be possible to realize those efficiencies inherent in an inverted file index.

Nevertheless, closer examination reveals some serious and inescapable limitations in the inverted file index. When each item address filed behind each term being sought must be read and compared, the number of reads and compares when searching against many terms can be huge, even though the whole file need not be read. The irony of it is that the more terms sought and the tighter the search parameters, the fewer hits one will get, but the more reads and compares will be necessary. In essence, then, the more terms that must be searched, the less efficient the inverted file becomes.

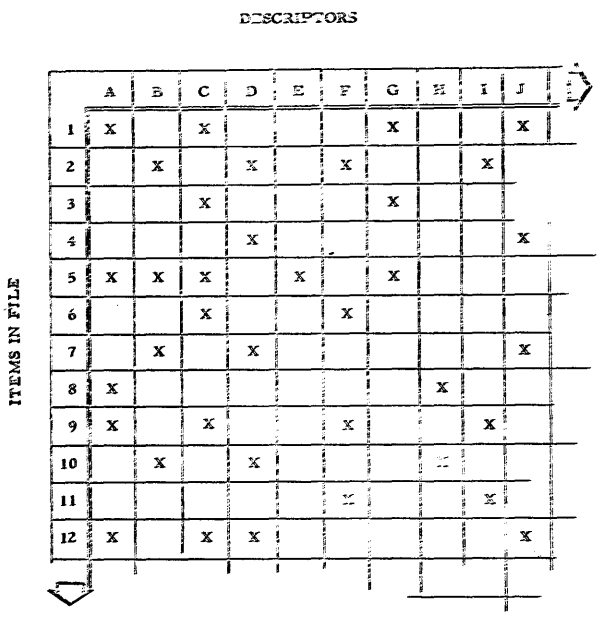
At this point, the proponents of classification and maybe even the proponents of subject headings might come forward to proclaim the advantages of their systems. In both instances, random access makes it possible to go directly to one address, as a rule, and read out a record or series of records. There is no need for multiple seeks or multiple reads and compares, and there is no searching of the complete file. No matter how big the file, a class or a subject heading will reduce the area to be searched to a practical size. In fact, this is exactly what some users of large descriptor files are doing: categorizing their documents by date or subject or both, thus setting up classes of files. When a search is required, it can then be confined to one class. How adequately such classification can characterize documents is problematical. In any event, the establishment and maintenance of classification tables represent additional expenditures of effort and cost, and nearly always impose certain restrictions.

Another way of speeding up search in serial files (being developed by some Navy groups) provides random superimposed coding for the descriptors of each item and results in very brief records, many of which can be read in a short time. Although the screening is rather coarse, with a number of false drops, all sought addresses are included, and the items found are then read in detail for fine screening. This method does increase the economic capacity of a file search but, as in all serial searches, efficiency decreases as the file increases.

When the search of a conventional file index or an inverted file index is completed, one has only a series of addresses for the hits. Since the separate references cannot be distinguished in a series of numbers, most systems maintain a bibliographic record to convert the addresses into standard bibliographic references. In addition to author, title, source, pagination, date and document number, these references may include a short abstract or, in lieu of an abstract, all the descriptors assigned to the title. Thus, a total computer-based retrieval system requires at least two sets of tapes or records: the index file, either conventional or inverted, and the bibliographic file, usually arranged in address order.

The Direct Access Search System proposed involves both kinds of records: an inverted file index with item addresses under each descriptor, and a conventional item file or bibliographic file where each item carries its pertinent descriptors. This is essentially no different from the file systems in many existing information retrieval installations, but the search strategy is not the item address match normally used with inverted files. The proposed system first reads into the memory all item addresses for the most significant descriptor of the set to be searched or, if they are all equal, then the descriptor with the fewest item addresses. Each recorded address is then read sequentially in the item or bibliographic file, to compare its descriptors against all other descriptors of the set being searched. For example, all documents described by descriptors A and B and C and D are wanted, and A is the most significant or has the fewest item addresses. The item addresses under A are examined in the item file, and all those items which also carry descriptors B and C and D are listed. Essentially, descriptor A here becomes a classificatory device or an associative memory, segregating a group of items for sequential examination, so that all the addresses listed under B and C and D need not be read and matched with the A addresses.

The example shown would apply to logical products (AND logic) and logical differences (AND NOT logic). Where a logical sum (OR logic) is required, all descriptors making up the logical sum must have their item addresses searched. For example, if (A or B) and (C and D) are sought, then all the items listed under A as well as all listed under B are examined for the presence of C and D. One way of displaying such a system is to consider an index as a matrix with the descriptors arranged horizontally and the items arranged vertically (see Table).


DESCRIPTORS
ITEMS IN FILE

	A	B	C	D	E	F	G	H	I	J
1	X		X				X			X
2		X		X		X			X	
3			X				X			
4				X						X
5	X	X	X		X		X			
6			X			X				
7		X		X						X
8	X							X		
9	X		X			X			X	
10		X		X				X		
11						X			X	
12	X		X	X						X

INDEX SHOWN AS A MATRIX

If the descriptor A, for example, is the prime term, column A is read and all the item addresses posted to it are recorded. Those item addresses which carry descriptor A are then read horizontally, and all descriptors (and additional information) posted to these items are examined in accordance with the search requirements. If a logical sum involving A or B is required, then the items under A and B must be scanned. When a list of item addresses becomes unwieldy, a second term (descriptor C, for example), can be matched with the prime term or sum of prime terms to narrow the search. If there is no logical product, that is, no significant second term to be matched, then there is no point in making an item search: the readout of the addresses posted to the single descriptor or the sum of the descriptors is sufficient.

One need not limit this approach to the classificatory use of descriptors. A date file can perform a similar function for a different application. Such a date file acts as a tickler index against a large file, and carries the identification of the item, its address in the master file and the next action date. Read daily, this file triggers action: the tagged item is retrieved from the large file for processing and its next action date is posted in the tickler file. Basically, this file is a classificatory device similar to the descriptor: both permit access at random to a set of desired items which are related in some way significant to the particular application.

Full utilization of such a system depends on a random access machine capable of going directly to specified addresses. If one must run through a whole file to find the addresses selected by the prime descriptor, then there are few or no benefits to be gained.

Combining the advantages of an inverted file with the flexibility and utility of serial search, the Direct Access Search System materially reduces the number of records which must be examined (thus, the number of reads and compares), and thereby not only shortens processing time, but can work against a file of any size, no matter how large.

This system was designed primarily to process very large files, but its flexibility offers additional capabilities which open up some interesting prospects. As has been pointed out,[6, 7] the serial search of a complete item file permits the development of association factors, i.e., relationships not anticipated by the searcher. This occurs in insurance underwriting, for example, where a file may be read for other significant facts about policyholders who are *male, under 21 years of age* and drive *sports cars.* These can be considered as three basic characteristics or descriptors: A, B and C. Now what other characteristics are associated with A, B and C, i.e., what is the experience of the insurance company with policyholders who match these classifications? Assume the insurance company has three million policyholders, each identified by 150 characters of coded information. With the standard tape file, the proposed analysis would mean reading 450 million characters. Assume also that this insurance company has two million male policyholders (descriptor A), 40,000 under 21 (B), and 30,000 who own sports cars (C). Under each descriptor is a list of 15-character item addresses. Reading and matching these three descriptors in an inverted file index would involve $(2{,}000{,}000 \times 15) + (40{,}000 \times 15) + (30{,}000 \times 15) = 31{,}050{,}000$ characters of reads and compares. With the Direct Access Search System, one need only look at 30,000 records of 150 characters each (or 4,500,000 characters), plus the original read and compare $(30{,}000 \times 15)$ for a total of 4,950,000 characters. This is little more than 1% of the whole file, and less than 20% of the number of characters involved in the inverted file match and compare. But, more important, the inverted file approach cannot develop the associated factors, e.g., geographic distribution and marital status, which could go into the rating of a policy. In many cases, furthermore, the advantages of Direct Access Search would be even greater. The example chosen was limited to three descriptors, but more might often be required; and with every additional descriptor, the efficiency of the inverted file drops appreciably.

A similar example could be developed for interrogating criminal records. Again, a descriptor or two applied as category definitions produce a small set of addresses which may be analyzed. Weather data, medical records and a whole host of other kinds of files which contain information of statistical significance can be similarly processed. Essentially, this technique has been applied, but not widely, being very expensive for large files with tape systems. And in essence, the ability to browse in a file is not new. However,—and this has been a basic criticism of mechanized information storage and retrieval systems—the searcher must formulate his question precisely, or the response will not satisfy his needs, because his question cannot "remind" the system of information he does not ask for. Now the requester can partially formulate his question and ask the computer to suggest associations and relationships which might be further explored. Thesauri and cross reference lists serve a similar purpose, but these represent semantic relationships only. Now, actual relationships, which may be semantic, syntactic, or statistical, can be brought to light by the computer, and thus the experience inherent in the information stored in the file can be displayed, permitting the requester to draw additional inferences. This is the way people browse in manual files, following various kinds of connections between entries; but the computer can more easily develop statistical criteria to help them in their browsing.

The example chosen had one short list of item addresses, which made the search time in the item file relatively short. A search question may, however, involve descriptors with many addresses, and logical sums (OR questions) which added together make for long lists of addresses. In such cases, assuming at least one logical product (one AND question), then a partial coordination would be advisable first (matching at least two terms in the inverted file) to reduce the number of addresses to be read. The item file would then be searched for only the matched addresses.

In addition, if many terms have been coordinated and search parameters have become too restrictive, the search can be conducted in such a way that the output lists, first, those hits where all terms were coordinated, then the hits for all terms but one, for all terms but two, and so on, in order.

In an inverted file, a term must be provided for every conceivable search concept. This becomes difficult, especially when numeric information may be sought. In such cases, the usual approach is to provide subdivisions of numeric ranges under applicable descriptors. For example, electrical resistors are indexed so that the individual types are grouped, by ohms, into various ranges. Where the inverted file descriptor is used as a classification device, such numeric subdivision of the individual descriptor is unnecessary, since the exact data can be located when the bibliographic file is examined in detail.

Employing a descriptor as a classificatory device opens up as an area for speculation the relationship of descriptors to class symbols. Using faceted classification terms or a semantic factor, one could use a facet just as readily as a descriptor in the proposed search system. The system would allow complete freedom in selecting the facets; i.e., their sequence, a difficult problem in manual systems,[8] would be quite immaterial to the computer, which could permute them as needed. This would offer extreme flexibility and remove one of the most serious objections to present classification schemes.

The use of a descriptor as a classificatory device becomes very important when the complete texts of documents are to be searched. Some programs and proposals contend that searching by means of any kind of descriptor is only a makeshift, and that the ultimate systems will search the complete text.[9] Since complete texts would be too voluminous, the usual approach is to set up special categories or classes and confine the search to such segments of the total file. However, as the Patent Office classification scheme is proving, it is practically impossible to construct classes that are mutually exclusive for every type of search. Cross references are a limited solution, but do not really overcome this difficulty. Categorization based on the actual occurrence of words and word combinations would not only make special classification schemes unnecessary, but would also make every significant portion of text directly accessible, regardless of sequence.

One must also keep in mind that with the present configurations of available equipment, a seek takes infinitely longer than a read and compare. Theoretically, a series of short serial searches which involve a number of seeks might seem more efficient than a very long read-and-compare, which has no seeks; whereas in actuality the opposite could be true. Therefore, one must be careful not to apply a technique which involves a number of seeks when a read-and-compare would be more economical. Where a great deal of processing must be done, various overlapping and batching techniques are valuable. For example, where the information is stored in a large slow file, the first batch of seeks can be read out into a fast file for processing while the next batch is being collected.

Essentially, the Direct Access Search System is not a single method, but offers a series of approaches from which the optimum is selected to meet the demands of the existing situation. The flexibility of the system makes it possible, for example, to look for pertinent syntactical relationships when reading the item file and examining the bibliographic record, since the record includes the title of the item and, in more sophisticated systems, often even an abstract. That is, the computer can tell the sequence of words and it can decide if the record is about *blind Venetians* or *venetian blinds*. This would obviate the need for recording links and roles and other elaborate paraphernalia often used to show syntactical relationships.

Many of us have, in the past, rejected Vickery's somewhat unconvincing contention[10] that mechanical indexing is a separate category. Yet with the machine now giving us certain special capabilities, we must reconsider that rejection and recognize mechanical indexing as different from the known manual systems. The search strategy of the Direct Access Search System makes it possible to approach a file from various sides and to manipulate the terms, enabling us to conduct searches in a manner not practical with manual systems, and more economically (in terms of characters read) than with other mechanized systems.

REFERENCES

1. SHAW, RALPH R., "Parameters for Machine Handling of Alphabetic Information," *American Documentation*, vol. 13, No. 3, p. 267-9, July 1962.
2. MELTON, JESSICA, "Vague New World," *Library Journal*, vol. 87, No. 13, p. 2493, July 1962.
3. WATSON-WATT, SIR ROBERT, "Are Computers Important?" *Proceedings of the Eastern Joint Computer Conference*, December 10-12, 1956, New York, New York, p. 67-8.
4. "Information Retrieval and the Design of More Intelligent Machines," Final Report No. AD59URI, University of Pennsylvania, The Moore School of Electrical Engineering, The Institute for Cooperative Research, July 31, 1959. (ASTIA AD-235999).
5. COSTELLO, JOHN C., JR., "Computer Requirements for Inverted Coordinate Indexes," *American Documentation*, vol. 13, No. 4, p. 414-9, October 1962.
6. BAKER, FRANK B., "Information Retrieval Based upon Latent Class Analysis," *Journal of the Association for Computing Machinery*, vol. 9, No. 4, p. 512-21, October 1962. (Note especially the bibliography, p. 521).
7. SALTON, GERARD, "Some Experiments in the Generation of Word and Document Associations," *AFIPS Proceedings—Fall Joint Computer Conference*, 1962, p. 234-50. (See bibliography p. 249-50).
8. *Proceedings of the International Study Conference on Classification for Information Retrieval*, Beatrice Webb House, Dorking, England, 13-17 May 1957. London ASLIB, 1957. (See especially p. 93).
9. NEWMAN, SIMON M., "Information Retrieval, Toward an Ultimate Universal System," *Revue Internationale de la Documentation*, vol. 29, No. 3, p. 7-9, August 1962.
10. VICKERY, B. C., *Classification and Indexing in Science*, 2nd ed., London, Butterworths, 1959.

A FLEXIBLE DIRECT FILE APPROACH TO INFORMATION RETRIEVAL–TEXT EDIT, SEARCH OR SELECT AND PRINT ON AN IBM 1401

Jane Olmer and Robert Rich
The Johns Hopkins University Applied Physics Laboratory
Silver Spring, Maryland

INTRODUCTION

Much has been published on the theories of information retrieval and various means of handling and expediting the ever-growing accumulations of information which are swamping our civilization. Not much has been said, however, about any attempt to make use of a small to medium scale computer as a tool. This is an eighteen months' progress report of such an attempt undertaken at the Johns Hopkins University Applied Physics Laboratory by a team consisting of computer people and librarians. Dr. Robert P. Rich, head of the University Computing Center, and the speaker represented the computer viewpoint. Mr. Fenton L. Kennedy, head of the APL Document Library, and Mrs. Mary E. Brown are the librarians on the team. Mr. Robert A. Lambert, formerly of the APL Document Library, took part in the early stage of development.

PROBLEM DEFINITION

Although the library people involved had an immediate interest in a particular application, the problem of keeping up with the huge number of scientific and technical reports received by the document library from within and outside the Laboratory, a broader view was taken in defining the problem. It was decided to develop a set of programs which would be general and flexible enough to handle a wide variety of applications in a multiplicity of input and output formats within a certain framework—the chief limitation being the size of the computer. The programs must create, maintain and update a file containable on one reel of magnetic tape and to retrieve information from it as effectively as possible. Time was available for research at the computing center on IBM 1401 computers. Economics of the solution would be investigated as experience developed during the research phase of the work.

Two other criteria were established for the project. The system must be simple for the user and the output easily and fully usable. That is, all output files must furnish required information directly without reference to other documents and all output files must be acceptable as input files for further, increasingly restrictive searches for information. Editing and printing routines able to accept all the files must also be available. In addition, the user must be able to communicate with the computer in a manner that would require little or no computer orientation on his part.

Immediately it was recognized by those taking part in the experiment that the problem divided—not always quite so neatly as might be wished, but in general—into two parts. One of these was what might be called semantics, that is, those decisions which involved a choice of material to be included in the file and of a terminology in which to express that material —within the IBM 1403 character set of the capitalized alphabet, numeric digits and 10 special characters. (The virgule or slash / and dollar sign $ were reserved for use of the system as two characters which could most easily be spared.) The computer person's part consisted of the syntactic portion of the problem. It was assumed by the program designers that the semantics had been determined in an acceptable manner leading to unambiguous and clearly defined symbols.

It was the job of the computer people to manipulate these so as to produce the specified results in a way that would take advantage of the characteristics of the equipment—the IBM 1401 computer, peripheral punch card machines, magnetic tapes, punch cards—to deliver the best results: 1) efficiently, 2) inexpensively, 3) and not least important, soon.

GENERAL SYNTACTIC APPROACH

The direct file approach was used, with a single master file containing all required information, one item for each document, including search terms and any desired "free" text such as abstracts and bibliographic information. Much has been written, stated and even proclaimed concerning the advantages of the alternative use of a record per term and cross referencing, manually or automatically, from an abbreviated indication to another master file containing the information sought. Without attacking or defending this latter view, let us consider some of the advantages of the direct file approach. It seems obvious that more sophisticated searches with a variety of logical operators are possible by processing each search on the complete set of descriptors for each document. By this method and by a separate pass for each search or group of searches, memory space is conserved. The output consists of exactly what the cataloguer sent to the computer without the necessity of sorts, merges or other additional computer passes.

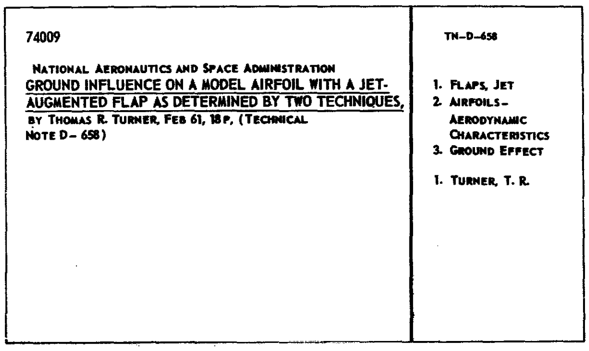
On a relatively small and inexpensive computer the simplicity of approach leading to easier, faster programming seems advisable until more complex methods are proved better or cheaper. It would be interesting to make test comparisons, but in the meanwhile experience with our method has shown us no evidence that more time is required—Groch's Law[1] not applying to this type of comparison.

MASTER FILE

By a master file we mean any file consisting of variable length records such as described above, whether these records each represent a document, a paragraph, or any desired grouping of characters or symbols which can be printed on the IBM 1403 with the limitations on length mentioned below in the description of the several programs in the package. The use of the dollar sign ($) in this file is generally restricted to the last character of each record, although we managed to outwit the program in writing this paper.

Key terms known as descriptors may be grouped together in each master file record. They may be preceded and/or followed by free text. A descriptor consists of a variable number of characters enclosed between a pair of slashes (/), but a single slash suffices between successive descriptors. The use of the virgule or slash (/), while it must precede and follow all key terms for the purpose of the search, will cause no difficulty if encountered elsewhere, except that in the search some free text or comments may be interpreted as key terms. If they are not being sought, no harm is done insofar as correct retrieval is concerned.

Figure 1, for example, shows a familiar 3×5 catalog card which may serve as the basis for such a master file record. Figure 2 shows the contents of the 3×5 card of Figure 1 expanded


74009

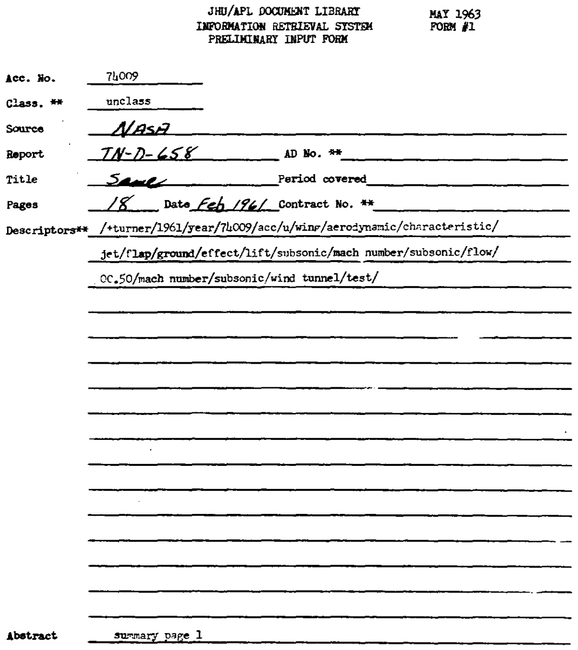
NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
GROUND INFLUENCE ON A MODEL AIRFOIL WITH A JET-AUGMENTED FLAP AS DETERMINED BY TWO TECHNIQUES, BY THOMAS R. TURNER, FEB 61, 18P, (TECHNICAL NOTE D-658)

TN-D-658

1. FLAPS, JET
2. AIRFOILS-AERODYNAMIC CHARACTERISTICS
3. GROUND EFFECT

1. TURNER, T. R.

Figure 1. Example of a 3×5 catalog card.


JHU/APL DOCUMENT LIBRARY
INFORMATION RETRIEVAL SYSTEM
PRELIMINARY INPUT FORM

MAY 1963
FORM #1

Acc. No.	74009
Class. **	unclass
Source	NASA
Report	TN-D-658 AD No. **
Title	Same Period covered
Pages	18 Date Feb 1961 Contract No. **
Descriptors**	/+turner/1961/year/74009/acc/u/wing/aerodynamic/characteristic/ jet/flap/ground/effect/lift/subsonic/mach number/subsonic/flow/ 0C.50/mach number/subsonic/wind tunnel/test/
Abstract	summary page 1

Figure 2. Key-punch form for Document Library Master File.

to contain a large number of specific descriptors in place of a few general headings. The material has been typed on a preprinted form designed for convenience of the cataloguer and of the keypunch operator. The form has recently been revised and improved.

THE SEARCH PROGRAM

The heart of the retrieval of the information on the computer is search Program PARSe (Paragraph Search Extended). The earlier version PARS, written for the 4K computer, has been incorporated into this program with a few vital features such as an input tape label check and tape read and write error subroutines as well as some new chimes and gongs.

Search Methods

This program searches a master file on tape in a manner specified by one or a series of search records. The latter are read into the computer on punched cards, one or more cards for each search.

A combination of descriptors and logical operators appear in Polish notation in the

$A	$O	$N
$E	$G	$L

Figure 3. Logical operators for use in Search Questions for program PARSE.

search record.[2] (Examples of this notation are given below.) They also may be preceded and/or followed by free text. A logical operator consists of a single character symbol preceded by a dollar sign ($). The $ preceding an operator replaces the / following a descriptor when an operator happens to follow a descriptor in a search record.

In addition to the logical operators "not" ($N), "and" ($A) and "or" ($O), three magnitude operators may be used, e.g., "equal" ($E), "less than or equal" ($L), "greater than or equal" ($G), as shown in Figure 3. All operators except the "not" are binary operators. The first two operators shown are each applied to the next two rightmost descriptors or results of operations. The magnitude operators are applied to pairs of descriptors in the search record. First, an identical descriptor is sought in the master file for the right hand member of the pair. If it is found, the magnitude test is applied to the next left descriptor in the master file, comparing it with the left hand member of the pair in the search record. For example, all descriptors written immediately to the left of the descriptor /YEAR/ in the master file may be tested to find those which are less than or equal to 1960 by writing

$L/1960/YEAR/

in the search record.

Magnitude for descriptors affected by operators $L and $G is based on the IBM collating sequence and may include both numerics and alphabetics as well as special printing characters except, of course, the / and $. All numerical descriptors written as the left hand member of such a pair in the master file or search record must consist of a set number of digits specified for the particular right hand member. Any pair of words in the master record regardless of length may be sought separately or linked by the operator $E. A string of several words may be linked by using an operator $E for each successive pair, repeating words as necessary.

The advantage of the Polish notation in applying logical operators to combinations of descriptors in the search question is the avoidance of nested parentheses and of ambiguity. For example, in Figure 4, the formula

$$(A+B)\times C$$

may be written in Polish notation as

$$\times + ABC$$

where, since all operators such as $\times$ and $+$ are binary operators, examining the expression from right to left clearly indicates that the pair of terms A and B are linked by the operator $+$, the resulting term and C are linked by the operator $\times$.

Using the set of operators described above for PARSE, referring to sucessive lines of Figure 5,

$A/SUBSONIC/TEST/

would constitute a search for all records containing both terms "subsonic" and "test."

$O/SUBSONIC/TEST/

describes a search for all records containing either one term or the other or both.

$A$N/SUBSONIC/TEST/

$$(A+B)\times C$$
$$\times + ABC$$

Figure 4. A formula written in standard algebraic and Polish notation.

$A/SUBSONIC/TEST/
$O/SUBSONIC/TEST/
$A$N/SUBSONIC/TEST/
$E/SUBSONIC/TEST/
$E/SUBSONIC/FLOW/
$A$O$L/ 00.55/MACH NUMBER$E/
SUBSONIC/FLOW$G/1960/YEAR/

Figure 5. Examples of Search Questions for program PARSE.

would produce a listing of all records containing "test" but not "subsonic." If it were desired to obtain all records containing /SUBSONIC/TEST/ in that sequence and to avoid such false drops as /SUBSONIC/FLOW/ or /SUPERSONIC/TEST/, the searcher would specify

$E/SUBSONIC/TEST/.

On the other hand, to find /SUBSONIC/ and /FLOW/ in juxtaposition we write

$E/SUBSONIC/FLOW/.

The operators $L and $G permit making more specific searches for ranges of values such as given years, temperatures, accession numbers, etc. For example,

$A$O$L/00.55/MACH NUMBER$E/
SUBSONIC/FLOW$G/1960/YEAR/

in a search card would create an output file of all records on the input tape which contained the terms /SUBSONIC/FLOW/ in that order or else a mach number of 00.55 or less, provided each report was dated 1960 or later. The English language description of the question may be included as free text in the search cards.

Our experience shows that a user, a librarian, for instance, easily acquires mastery of this technique of writing questions. At a demonstration of the program at APL, about 30 visitors from local special libraries and other installations were taught the method and produced 10 questions to be processed against the unclassified file of the APL documents library. All of these questions, some of which were fairly complex, were found by the computer to be grammatically correct and were correctly processed. A simple means of checking grammatical accuracy included in the half-hour workshop was as follows—counting from the right, add one for each term and subtract one for each operator. If the sum ever falls below one or the final total is not exactly one, the expression is erroneous.

The program provides also for a vacuous search. That is, a search record consisting entirely of free text will seek out all master file records consisting entirely of free text (no descriptors). By error or design, records not yet processed completely could be added to the file, and it might be worthwhile to print these out in their entirety, or to provide a listing of, say, accession numbers only. This could be accomplished by introducing into the PARSE program any search question not containing a slash—any expression from a single character to a paragraph of explanation. The appropriate parameter card would enable the PRINT or another printing program to produce the entire record or a selected part of it.

A practical method for cutting down search time consists of batching a number of searches by use of the logical operator $O. A single pass over the entire master file yields, usually, a considerably smaller tape file against which the individual search questions may then be processed. A special, much faster, single term search program SCAN has also been written to subdivide a master file and/or to reduce multi-reel to single reel files.

The search record is identified and printed on line as well as being written on an output tape. (See Figure 6 for an example of an on-line printout.) It is followed on the output tape by each master record which constitutes a hit. The number of such hits is printed on line and the master tape is rewound.

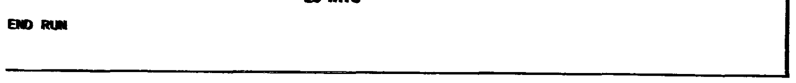
The next search is then undertaken, or if there are no more searches, the output tape is rewound. An option permits writing of successive searches on separate tapes if desired. An input label check and creation of an output label are included, but options permit bypassing this subroutine or overriding it. (Figure 7 shows a portion of such an output tape printed, including as the first hit record the familiar one created from the 3×5 card of Figure 1 and the preprinted sheet of Figure 2.)

Computer Processing of the Search

The computer program PARSE starts out by measuring the length of the program (in case new features have been added; it is designed to permit adding subroutines for additional operators, for example) and the size of the computer memory being used. The storage space available is computed including the area in which this part of the program lies since it will wipe itself out before proceeding to the main part of the program. Half of this storage area is assigned for containing the master file record as it is read in, half is assigned for the search record to be assembled from one or more successive punch cards, unless this is not enough space for the longest file record as specified by a parameter card. In this case the file record is assigned the needed space and the remainder is assigned to the search record.

The parameter card also may designate specific areas to be used if desired or may, once the program becomes a production run, specify that a preset constant be used as the maximum file size. Any attempt by the human to do something irrational such as to process a record longer than the number of characters available on the computer, leaving no room for the search record, will be rejected by the program with an angry message and a wind-up of tape.

It is at this time that the label of the input tape may be checked if memory space permits



```
SRCH
     JOHN JONES-5. MORE SPECIFIC THAN QUESTION 4. ALL RECORDS EITHER ON SUBSONIC FLOW
      OR WITH MACH NUMBER .55 OR LESS BUT ONLY FOR YEARS 1960 OR LATER.   $ASA$CSL/
      OO.55/MACH NUMBER$E/SUBSONIC/FLOW$G/1960/YEAR/U/
                                      23 HITS

END RUN
```

Figure 6. On-line printout for normal search, last of a series.

```
                                                                                                              PAGE   64
                                                                                                                 1
           1          2          3          4          5          6          7          8          9          0
  1234567890 1234567890 1234567890 1234567890 1234567890 1234567890 1234567890 1234567890 1234567890 1234567890

0 00333  740 09 UNCLASS * NASA. TN -D-658.* G RCUND INFL UENCE ON A  MCDEL AIR FOIL WITH  A JET-AUGM ENTED FLAP
1  AS CETERM INED BY TW C TECHNIQU ES.  18P.  FEB 1961.*  /TURNER/1 961/YEAR/7 4009/ACC/U /WING/AERO DYNAMIC/CH
2 ARACTERIST IC/JET/FLA P/GROUND/E FFECT/LIFT /SUBSONIC/ MACH NUMBE R/SUBSONIC /FLOW/00.5 0/MACH NUM BER/SUBSON
3 IC/WIND/TU NNEL/TEST/ FRCM THIS  INVESTIGAT ION IT APP EARS THAT  THE LOSS I N LIFT OF  AN AIRFOIL  WITH A JE
4 T-AUGMENTE C FLAP IN  GRCUND INL UENCE AS D ETERMINED  IN A WIND  TUNNEL WIT H A CONVEN TIONAL GRO UND-BOARD
5 SETUP IS C ONSICERABL Y LARGER T HAN WCULD  BE OBTAINE D IN FREE  FLIGHT.$

                                                                                                                 1
           1          2          3          4          5          6          7          8          9          0
  1234567890 1234567890 1234567890 1234567890 1234567890 1234567890 1234567890 1234567890 1234567890 1234567890

0 00334  740 10 UNCLASS * NASA. TN -D-660.* E FFECTS OF  VARIOUS AR RANGEMENTS  OF SLOTTE D AND ROUN D JET EXIT
1 S CN THE L IFT AND PI TCHING-MCM ENT CHARAC TERISTICS  OF A RECTA NGULAR-BAS E MODEL AT  ZERO FORW ARD SPEED.
2   25P. FEB  1961.* /V CGLER/1961 /YEAR/7401 0/ACC/U/GR OUND/EFFEC T/VEHICLE/ AERODYNAMI C/LIFT/AER ODYNAMIC/P
3 ITCH/AEROC YNAMIC/STA BILITY/AIR /JET/AIR/F LOW/$

                                                                                                                 1
           1          2          3          4          5          6          7          8          9          0
  1234567890 1234567890 1234567890 1234567890 1234567890 1234567890 1234567890 1234567890 1234567890 1234567890

0 00335  740 11 UNCLASS * NASA. TN -D-781.* F IRST PLANN ING CONFER ENCE ON BI OMEDICAL E XPERIMENTS  IN EXTRAT
1 ERRESTRIAL  ENVIRONME NTS.  85P.  FEB 1961. * /1961/YE AR/74011/A CC/U/SPACE /MEDICINE/ SYMPOSIUM/ COSMIC/RAD
2 IATION/EFF ECT/SUN/RA DIATION/EF FECT/BIOCH EMISTRY/CH EMISTRY/SP ACE FLIGHT /SPACE/RES EARCH/SPAC E/ENVIRONM
3 ENT/PHYSIC LCGY/$

                                                                                                                 1
           1          2          3          4          5          6          7          8          9          0
  1234567890 1234567890 1234567890 1234567890 1234567890 1234567890 1234567890 1234567890 1234567890 1234567890

0 00336  740 12 UNCLASS * NASA. TR -R-66.* A  STUDY OF T HE ASYMMET RIC TRANSO NIC FLOW P AST A SHAR P LEADING
1 ECGE.  70P . 1960.* / STINE/WAGC NER/LUGIN/ 1960/YEAR/ 74012/ACC/ U/TWO DIME NSIONAL/TR ANSONIC/FL OW/TWO DIM
2 ENSICNAL/F LOW/SUPERS CNIC/FLCW/ 02.00/MACH  NUMBER/02 .80/MACH N UMBER/FLUI D/MECHANIC /SUPERSONI C/WING/AER
3 CCYNAMIC/$

                                                                                                                 1
           1          2          3          4          5          6          7          8          9          0
  1234567890 1234567890 1234567890 1234567890 1234567890 1234567890 1234567890 1234567890 1234567890 1234567890

0 00337  740 14 UNCLASS * OHIO STA TE UNIV. R .R.* MAINT ENANCE CON CEPTS AND  STRUCTURE  FOR OPTIMU M SUPPORT
1 OF MISSILE  WEAPON SY STEMS.  62 P. 1 JUN 1 960.* /PAG E/HUSTON/P IERCE/1960 /YEAR/7401 4/ACC/U/MI LITARY/RES
2 EARCH/MILI TARY/LOGIS TIC/OPERAT ION/RESEAR CH/MISSILE /MAINTENAN CE/BM/SM/I CBM/GM/SAM /SSM/BOMAR C/ATLAS/TI
3 TAN/MINUTE MAN/WEAPCN /MAINTENAN CE/MISSILE /LAUNCH/MA INTENANCE/ DESIGN/MIS SILE/DESIG N/WEAPON/D ESIGN/MISS
4 ILE/EVALUA TION/MISSI LE/RELIABI LITY/WEAPO N/EVALUATI ON/WEAPON/ RELIABILIT Y/MISSILE/ WEAPON/SYS TEM/MILITA
5 RY/MANAGEM ENT/$
```

Figure 7. Example of a listing of Master File using program PARL.

reading the first record of the tape. If there is not enough space this is bypassed. Then all this preliminary coding erases itself and the search proper begins.

The search record is assembled from successive cards, squeezing out marginal spaces, inserting a space between the ending word of one card and the first word of the next card unless the first card ended with a / (no space inserted in order not to affect descriptors). A character by character scan records the location of the left-most and right-most slashes so that subsequent search scans need not waste time on text other than the descriptors. Word marks are placed under slashes to permit picking up an entire term at a time from right to left, permitting efficient processing on the IBM 1401.

An operations table is created of a string of symbols consisting of a zero for each descriptor and a copy of each logical operator symbol from the search record. Here again one proceeds from right to left. This table permits use of the conditional branch instruction on a single designated character to select the most efficient path through the appropriate subroutines during the actual search of each file record. The editing of the search record and creation of the operations table are carried out only once for searching an entire file.

Next, a dummy file record consisting of a single slash is created and a pseudosearch is made. This, together with a couple of other tests, automatically checks the grammar of the search question. Specific error codes will be printed on line if applicable.

Each record on the master file is edited in a manner similar to that described for the search record. Guided by the operations table, each term in the search record is checked in the master file and the results recorded in a function table, a zero if the term is not found, a one if it is. Each time a logical operator is encountered, the last two digits in the function table are combined. Depending on the operator the result of the combination is recorded as a single zero or one. Since each term increases the size of the function table by one character and each operator decreases it by one, the table can never contain less than one character and must, at the end of the search, contain exactly one. This corresponds to the grammatical check mentioned earlier.

If the final character in the function table is a one, a hit is recorded and the record copied onto the output tape. If the character is a zero, the next file record is processed. Any other character would lead to an alarm printout and discontinuance of that search. Efficiency is increased by looking ahead two characters in the operations table after each new entry in the function table. Thus the program can skip over one term for the "and" ($A) operator when a previous no-hit has been recorded. Similarly, the search is shortened when an "or" ($O) operator follows a hit. When a magnitude operator is encountered it is processed appropriately.

At the end of each complete search of a file the program winds up the input tape. If another search card follows, the program begins the next search, starting once more by assembling all cards of the next question as previously described. If an option has been exercised by depressing a panel switch, the search results are written on separate tapes. Otherwise the output files follow each other on the same tape, each being identified by the search record at its head.

Error Diagnosis

Three types of errors have been provided for in the programs. The first is the type which prevents continuing with the program. In this case, an alarm is printed out and the program winds up all tapes. An example of this first type of error would be the attempt to set up the program PARSE in its initial phase to accept a file larger than available storage permits.

The second type of error is one which will not permit processing, but which can be bypassed. For instance, an ungrammatical search or one which contains more descriptors and operators than the program can handle (the present limitation is a total of 300) is the occasion for an alarm printout, bypassing any further cards for that search, and continuing on to the next search, if any.

The third type of error is one which can be analyzed by the program and an attempt made to carry out the aims of the customer. An error is assumed to be in this category only when there is a high probability of second-guessing the user, and is always signalled by a warning printout. For example, in the PARU program the omission of an ending signal for an insertion of text followed by a correction addressed to a later paragraph causes all text to be inserted except for trailing spaces, along with a warning printout. The librarians have expressed great satisfaction with this as a time saving feature.

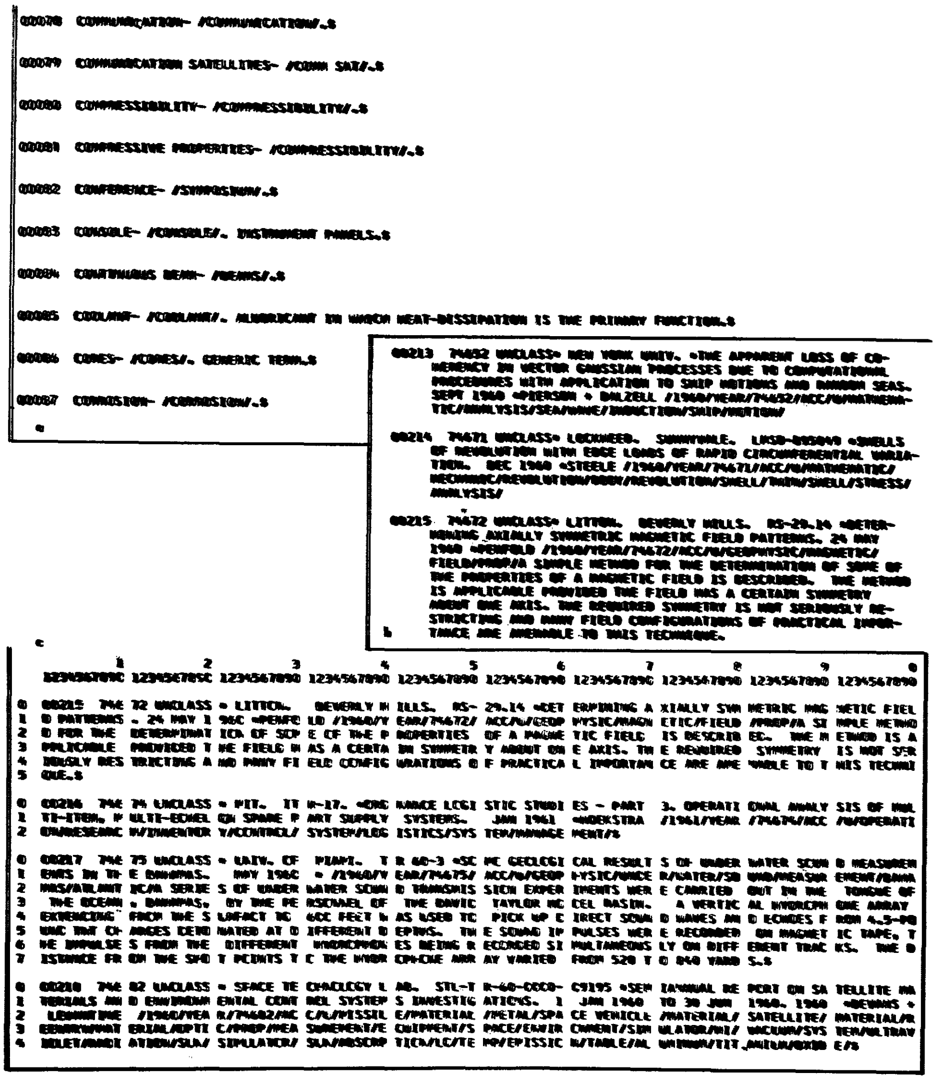
Since a variety of alarm and warning messages for printout would be expensive of memory space, all such messages in PARSE are codified by an alarm subroutine which prints out the word alarm and a number positively identifying the location and nature of the error.

We exclude from prior analysis, of course, that unforeseen bug which pops up long after final debugging.

FILE ESTABLISHMENT, MAINTENANCE AND PRINTOUT

Several variant programs have been written to accomplish the establishment, correction or enlargement of a file such as the master file described above. The EDIT program is for the 8K memory computer and accepts records as long as 3600 characters. The less versatile PARU (paragraph update) program for the 4K or 8K memory is restricted to records of not more than 1000 characters. Both of these programs accept text on punch cards of a variety of formats specified by parameter or signal cards and the dollar sign to create the file. Thereafter, the tape file is maintained by correction cards introduced with one of these same program decks. Whether in establishing the file or updating it, both programs assign to each record a five-digit sequential number which serves to identify the paragraph in correcting, inserting or deleting records. The program TEST was written to list and validate cards to be used as input to the program EDIT. Nothing paticularly original has been included in these programs.

A number of programs, each with a number of options, exists for the purpose of printing the file. Figures 7 and 8 show some of these. Figure 7 reproduces a verbatim copy, 100 characters to the line with (option, without) numbered lines for each record, 100 character lines with (option, without) a space inserted after each tenth character and with a heading of digits from 1 to 100 preceding each record (option, heading each page). This type of format is particularly useful in editing the file and writing corrections. More readable texts,



```
000078  COMMUNICATION- /COMMUNICATION/.$

000079  COMMUNICATION SATELLITES- /COMM SAT/.$

000080  COMPRESSIBILITY- /COMPRESSIBILITY/.$

000081  COMPRESSIVE PROPERTIES- /COMPRESSIBILITY/.$

000082  CONFERENCE- /SYMPOSIUM/.$

000083  CONSOLE- /CONSOLE/. INSTRUMENT PANELS.$

000084  CONTINUOUS BEAM- /BEAMS/.$

000085  COOLANT- /COOLANT/. ALUBRICANT IN WHICH HEAT-DISSIPATION IS THE PRIMARY FUNCTION.$

000086  CORES- /CORES/. GENERIC TERM.$

000087  CORROSION- /CORROSION/.$
  a
```

```
00213  74452 UNCLASS* NEW YORK UNIV. *THE APPARENT LOSS OF CO-
       HERENCY IN VECTOR GAUSSIAN PROCESSES DUE TO COMPUTATIONAL
       PROCEDURES WITH APPLICATION TO SHIP MOTIONS AND RANDOM SEAS.
       SEPT 1960 *PIERSON + DALZELL /1960/YEAR/74452/ACC/U/MATHEMA-
       TIC/ANALYSIS/SEA/WAVE/INDUCTION/SHIP/MOTION/

00214  74671 UNCLASS* LOCKHEED. SUNNYVALE. LMSD-895049 *SHELLS
       OF REVOLUTION WITH EDGE LOADS OF RAPID CIRCUMFERENTIAL VARIA-
       TION. DEC 1960 *STEELE /1960/YEAR/74671/ACC/U/MATHEMATIC/
       MECHANIC/REVOLUTION/BODY/REVOLUTION/SHELL/THIN/SHELL/STRESS/
       ANALYSIS/

00215  74672 UNCLASS* LITTON. BEVERLY HILLS. RS-29.14 *DETER-
       MINING AXIALLY SYMMETRIC MAGNETIC FIELD PATTERNS. 24 MAY
       1960 *PENFOLD /1960/YEAR/74672/ACC/U/GEOPHYSIC/MAGNETIC/
       FIELD/PROP/A SIMPLE METHOD FOR THE DETERMINATION OF SOME OF
       THE PROPERTIES OF A MAGNETIC FIELD IS DESCRIBED. THE METHOD
       IS APPLICABLE PROVIDED THE FIELD HAS A CERTAIN SYMMETRY
       ABOUT ONE AXIS. THE REQUIRED SYMMETRY IS NOT SERIOUSLY RE-
       STRICTING AND MANY FIELD CONFIGURATIONS OF PRACTICAL IMPOR-
  b    TANCE ARE AMENABLE TO THIS TECHNIQUE.
```

```
  c
           1          2          3          4          5          6          7          8          9          0
  1234567890 1234567890 1234567890 1234567890 1234567890 1234567890 1234567890 1234567890 1234567890 1234567890

0 00215  74E 72 UNCLASS * LITTON.   BEVERLY H ILLS.  RS- 29.14 *DET ERMINING A XIALLY SYM METRIC MAG NETIC FIEL
1 D PATTERNS . 24 MAY 1 960 *PENFO LD /1960/Y EAR/74672/ ACC/U/GEOP HYSIC/MAGN ETIC/FIELD /PROP/A SI MPLE METHO
2 D FOR THE  DETERMINAT ION OF SOM E OF THE P ROPERTIES  OF A MAGNE TIC FIELD  IS DESCRIB ED.  THE M ETHOD IS A
3 PPLICABLE  PROVIDED T HE FIELD H AS A CERTA IN SYMMETR Y ABOUT ON E AXIS. TH E REQUIRED  SYMMETRY  IS NOT SER
4 IOUSLY RES TRICTING A ND MANY FI ELD CONFIG URATIONS O F PRACTICA L IMPORTAN CE ARE AME NABLE TO T HIS TECHNI
5 QUE.$

0 00216  74E 74 UNCLASS * MIT.  IT R-17. *ORG ANCE LOGI STIC STUDI ES - PART  3. OPERATI ONAL ANALY SIS OF MUL
1 TI-ITEM, M ULTI-ECHEL ON SPARE P ART SUPPLY  SYSTEMS.   JAN 1961  *HOEKSTRA  /1961/YEAR /74674/ACC /U/OPERATI
2 ON/RESEARC H/INVENTOR Y/CONTROL/ SYSTEM/LOG ISTICS/SYS TEM/MANAGE MENT/$

0 00217  74E 75 UNCLASS * UNIV. OF  MIAMI.  T R 60-3 *SC MC GEOLOGI CAL RESULT S OF UNDER WATER SOUN D MEASUREM
1 ENTS IN TH E BAHAMAS.   MAY 1960  * /1960/Y EAR/74675/ ACC/U/GEOP HYSIC/UNDE R/WATER/SO UND/MEASUR EMENT/BAHA
2 MAS/ATLANT IC/A SERIE S OF UNDER WATER SOUN D TRANSMIS SION EXPER IMENTS WER E CARRIED  OUT IN THE  TONGUE OF
3  THE OCEAN , BAHAMAS,  BY THE PE RSONNEL OF  THE DAVID  TAYLOR MO DEL BASIN.  A VERTIC AL HYDROPH ONE ARRAY
4 EXTENDING  FROM THE S URFACE TO  600 FEET W AS USED TO  PICK UP D IRECT SOUN D WAVES AN D ECHOES F ROM 4.5-PO
5 UND TNT CH ARGES DETO NATED AT D IFFERENT D EPTHS.  TH E SOUND IM PULSES WER E RECORDED  ON MAGNET IC TAPE, T
6 HE IMPULSE S FROM THE  DIFFERENT  HYDROPHON ES BEING R ECORDED SI MULTANEOUS LY ON DIFF ERENT TRAC KS.  THE D
7 ISTANCE FR OM THE SHO T POINTS T O THE HYDR OPHONE ARR AY VARIED  FROM 520 T O 840 YARD S.$

0 00218  74E 82 UNCLASS * SPACE TE CHNOLOGY L AB.  STL-T R-60-0000- C9195 *SEM IANNUAL RE PORT ON SA TELLITE MA
1 TERIALS AN D ENVIRONM ENTAL CONT ROL SYSTEM S INVESTIG ATIONS.  1  JAN 1960  TO 30 JUN  1960. 1960  *DEVONS +
2  LEONARD  /1960/YEA R/74682/AC C/U/MISSIL E/MATERIAL /METAL/SPA CE VEHICLE /MATERIAL/ SATELLITE/ MATERIAL/R
3 EINFORCEMENT ERIAL/OPTI C/PROP/MEA SUREMENT/E QUIPMENT/S PACE/ENVIR ONMENT/SIM ULATOR/MI/ VACUUM/SYS TEM/ULTRAV
4 IOLET/RADI ATION/SUN/ SIMULATOR/ SUN/ABSORP TION/LC/TE MP/EMISSIO N/TABLE/AL UMINUM/TIT ANIUM/OXID E/$
```

Figure 8. Examples of printing formats. (a) Excerpt from a thesaurus; printed by LIMF program with all characters in record printed, 100 to a line. (b) Printed by PRINT program with overhanging paragraphs, single spaced with double space between records, automatic hyphenation, maximum of 67 characters per line, last character of record deleted. (c) Printed by PARL program with space after each tenth character and characters numbered with hundreds in left margin of each record, tens and units across top of page. Both (b) and (c) are excerpts from an unclassified Master File. This file was the output from a search of the entire Master File; it was then processed by PARU for the purpose of renumbering the records sequentially.

some of which are shown in Figure 8, may be produced with a repeated page heading, indented or overhanging first lines of paragraphs, beginning or ending of paragraphs suppressed, a choice of page and margin widths and regular or irregular right-hand margins with words hyphenated according to an ingenious (not yet perfect) scheme for syllabification devised by Dr. Rich. Other features would be tedious to enumerate.

These programs include LIMF (list multiple file), and PARL (paragraph lister) for the 4K or 8K memory and the more powerful PRINT for the 8K. As is the case with all the programs described, these may, of course, be applied not only to the original master file, but to the subfiles produced as a result of searches. It may be of interest to note that the several drafts of this paper were key punched, put on magnetic tape, and updated and printed in several convenient formats by means of the programs mentioned in this section.

SUBSTITUTION PROGRAM

It seemed desirable to facilitate changes in approach by the librarians during the research period, as well as to provide for the inevitable reversals of direction taken by a cataloguer during the expansion of a file. A substitution program has been written by Mr. T. O. Hiltabidle of IBM for this purpose. This program accepts as input any two strings of characters, the first string to be replaced wherever encountered in a file by the second string. The beginning and end of the strings are identified by any single character not used elsewhere in the input card.

TABLEDEX

A parallel effort has been carried out in programming for another APL computer, the IBM 7094. Another set of information retrieval programs, TABLEDEX,[3] was designed and programmed by Dr. Robert S. Ledley of National Biomedical Research Foundation, Inc. and Mr. Fred S. Zusman. This program package provides an additional tool for the cataloguer and manual search method for the customer. A listing of all terms found in a file except those designated in a stop list of entire words or first characters may be produced. These may also be listed together with a reference to appropriate paragraph numbers. Both these listings have proved valuable in correcting misspellings and redundant forms. Checked against the thesaurus, the listing helps the cataloguer to improve that document.

TABLEDEX will be applied to a reasonably sized result of a PARSE search. The tables themselves will then be printed in a convenient format. The librarian may thus provide his clientele with the means to making simple manual searches. These may be more specific than the original PARSE search.

It is felt that the APL search programs and the TABLEDEX complement each other and add a further dimension of flexibility to the solution of the problem as defined above.

STATUS OF THE EXPERIMENT

At the time of writing, the Information Retrieval Program has been under way for about eighteen months. In the major application to date, almost 3000 documents have been compiled on an APL Document Library Master File. About 175 new accessions are now being added each week. The documents average 1000 to 1500 characters with none longer than 3600 characters. They are roughly ordered by week of receipt and are usually divided into the following parts—bibliographic information, descriptors, abstract. A thesaurus of some 1200 generally used terms has been compiled, first drawn from the terms in the documents as they were catalogued. The thesaurus was continuously expanded up to about 2000 terms and then refined making use of the PARU and EDIT programs to create and update the magnetic tape on which the alphabetic list of terms, explanations and cross references to canonic forms is stored. Identifiers, which were included in the thesaurus, are now maintained in a separate list which includes definitions and amplifications of abbreviations and initials. By including all classified terms in this file it is possible to create an unclassified thesaurus which may be used outside the Laboratory. The list contains about 1000 identifiers at this time.

It is also contemplated to retain a file of interest profiles for Laboratory scientists and to make use of the search and printing program for periodic selective dissemination of additions to the master file. In six to eight weeks an experiment along these lines may be undertaken for some 20 scientists.

Other document libraries testing the program include the University of Texas M.D. Anderson Hospital and Western Electric Co. Engineering Research Center. Use has also been made of the package for such disparate files as an index of printing type specimens, a chapter of a book on the history of the Laboratory, profiles of medical personnel, program library indexes and papers such as this one. Interest has been indicated in such applications as a parts catalogue with a search to be made on parts specifications as descriptors.

A few timing runs have been made which, not surprisingly, show each of the runs to be extremely dependent on the length of the master file, although PARSE is computer limited. Roughly 3 to 10 records per second were searched and hits recorded for a single-term search. About 2 per second was the time for a fairly complex question. Further investigation is in order on the breakpoint between speed and good output. Limiting the file record size and arrangement may gain speed at the sacrifice of material sufficient to satisfy the needs of the customer. Of course, a modicum of subjectivity is inherent in such an evaluation.

In any event, the lion's portion of the expense is attributable to the share of the work so lightly brushed aside in the introduction of this paper—the semantic problems which go hand in hand with the syntactic, but which have been left to the librarian members of our team to describe.[4] The cost of processing one scientific report up to the point of file search has been estimated at $4.00.

CONCLUSION AND SUMMARY

The set of APL programs described here will simplify the routine work of the cataloguer and permit him to devote himself to more rewarding tasks. Cataloguing may be more complete than in the traditional manual card systems. The catalogue may expand and be kept up to date. Once catalogued the information may be retrieved by writing powerful searches, using as complex combinations of descriptors and logical operators as need be. False drops may be avoided by linking a number of descriptors. Ranges of values may also be indicated in the question. Successive searches on diminishing sizes of files may be undertaken, going from the general to the more specific.

On-line printouts giving the number of hits for each search indicate whether a manageable segment of information has been put on tape or whether further searches should be made. Most important, all information needed by the user is available for any document found and may be printed in a variety of formats depending on the use to be made of the material as well as individual preferences. Error diagnosis saves time for the computer and searcher by printing a definitive code, going on to the next order of business where possible. Batching of several searches for a master file also saves set-up time. Consolidating searches effects economy.

The semantic part of the information retrieval problem is the province of the cataloguer, be he librarian or other user of an accumulation of information. Insofar as the APL information retrieval program is concerned, this information may be expressed in any sensible set of symbols, abbreviated or not, coded or not, systematized by hierarchy (such as Dewey Decimal) or as uniterms or anything between—or unsystematized. Of course, while one may search for any information at all, one will find only what has been included and identified as a descriptor. Therefore the task of the cataloguer is not to be minimized.

A contribution, we feel, has been made to the solution of the syntactic part of the information retrieval problem. A flexible, rather efficient and economical program now exists for a medium sized computer for files of limited size. The program is designed to permit further expansion and development. It lends itself to a variety of applications. One of these which has been described here is estimated to cost roughly $4.00 per document for cataloguing, putting on tape, printing and making any necessary corrections.

REFERENCES

1. GROSCH, H. R. J. A view from the bridge. *Datamation,* June 1962, Vol. 8, No. 6, p. 30.
2. LUKASIEWICZ, J., and TARSKI, I. Untersuchungen uber den aussagenkalkul. *Comptes Rendus Des Seances De La Societe Des Sciences Et Des Lettres De Varsovie.* 1930, Vol. 23, Classe III, Fascicule 1–3, pp. 51–77. ROSENBLOOM, PAUL. *The Elements of Mathematical Logic.* Dover Publications, 1950, p. 202.
3. LEDLEY, ROBERT S. Tabledex. A new coordinate indexing method for bound book form bibliographies. Proceedings of the International Conference on Scientific Information, National Academy of Sciences, 1959, pp. 395–417.
4. KENNEDY, FENTON L. *The Application of the IBM 1401 Computer to the APL Storage and Information Retrieval System.* The Johns Hopkins University Applied Physics Laboratory, CF–3022, Feb. 1963.

EXPERIENCE WITH A GENERALIZED INFORMATION PROCESSING SYSTEM

Martin Kosakoff and Donald L. Buswell
U. S. Naval Ordnance Laboratory
Corona, California

INTRODUCTION

A generalized computer information storage and retrieval system has been in use at the Naval Ordnance Laboratory, Corona, California since 1960. The system, called Variable Information Processing (VIP), has permitted the establishment and use of files of many varied and interrelated types of data with a minimum of effort on the part of the user, little effort for system setup, and no effort for computer programming. It has been used for storage and retrieval of loosely defined, nebulously structured bodies of information and even plain-text natural language information.

ASPECTS OF THE SYSTEM

This system provides for generalized storage, on magnetic tape files, of information structured into two levels—records and fields. (Characters, the smallest elements of information, might also be considered a level.) Files are made up of an unlimited number of records, records of a variable number of fields, and fields of a variable number of characters. The characters may take on any of the values available in the hardware character set—minus one, which is reserved for use in separating fields and records. The addressing of fields is by their relative order with respect to preceding fields. Retrieval of information is available not only on the basis of the two structural levels of information (records and fields), but also on a content defined entity called a subfield. A subfield is a set of contiguous characters within a field which is defined by its relative position as determined either by character count or with respect to the occurrence of any character value within the field.

The manipulation and retrieval of information is performed utilizing a battery of generalized processor routines, that is, machine routines to which are specified the parameters to be used for a given job. The processors, though general, are primarily elemental in that they do one function only, such as logical selection, file sorting, counting, etc. (The flexible power of the system early in its development with only an input program, a sort/merge and a "sequence counter"/output program was really amazing.) Available now, through continued development of the system, is a fairly full spectrum of information message capabilities including full logical (and, or, not, equal, greater than, less than) retrieval, various types of record and field manipulation and association, file summarization and abstraction, flexible outputting and formatting and arithmetic and statistical operations.

In order to perform complex retrievals using elemental processors it is necessary to link several processors together to achieve the desired results. This "system set up", which is actually programming at the systems level (figure 1), requires the analyst to have a

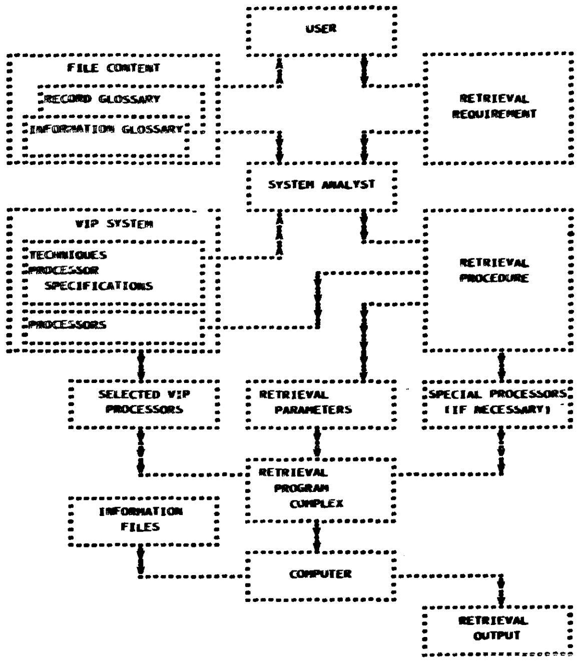


Figure 1. Retrieval via VIP System.

knowledge of the file organization and content, and of the operation of the processors available in the system. Special purpose processors may be developed when necessary. Once a process is set up it can be repeated by a re-run of the setup or used to perform a modified process with only minor parameter changes.

FLEXIBLE ORGANIZATION OF DATA

By now there has been the development of a profusion of "general" information processing and retrieval systems, such as report generation[1], GIRLS[2], and RECOL[3], for it seems that necessity has taught many of us the same lesson and has led us to similar conclusions. We feel, however, that there is one essential difference between the VIP system and others which have been noted—and that is the complete variability in length of *all* levels of information. There are absolutely no length or physical positional restrictions on the information which can be accommodated. This includes elements which are used for sequencing and identification as well as description. It is perhaps this feature which provides the added generality and flexibility which makes for extremely easy establishment and accommodation of new projects. No longer is the guessing game of conventional systems necessary where one must decide how many characters to make available for this, that, or another item of information. A field within the VIP system contains exactly the characters which make up that field's value regardless of number; there is no need to even anticipate the size. Too, the availability of (almost) the complete character set permits the use of mnemonic abbreviations and even natural language rather than difficult-to-translate coding. The possible number of unique values which a field may take on need not be fixed, thus permitting the dynamic development of terminology on an "as needed" basis.

The storage of records of varying types and formats within a VIP file is quite easily managed. There are two methods of establishment of the identity of a field: (1) by pre-established relative field location; (2) on the basis of the contents of key portions of the record. For example, a personnel file could contain various record types such as are shown in figure 2 where the identity of fields 1 and 2 are pre-established, and that of the remaining fields dependent on the contents of field 2 of the record.

The subfield concept often provides for easy structuring of information, much of which normally occurs properly set up as subfields. Dates might have diagonals as separators

Identification Record

Field 1	Payroll number	
Field 2	IDENTIFICATION (record type)	
Field 3	Name of employee (last, first, middle initial)	
Field 4	Address	
Field 5	Sex	

Skills Record

Field 1	Payroll number	
Field 2	SKILLS (record type)	
Field 3	Skills descriptors (separated by commas)	

Education Record

Field 1	Payroll number	
Field 2	EDUCATION (record type)	
Field 3	Name	School
Field 4	Degree/year	
Field 5	Major	
Field 6	Name	School
Field 7	Degree/year	
Field 8	Major	
	.	
	.	
	.	

Figure 2. Various Record Types.

(9/1/63 or 12/25/63); college degree information might be formalized with commas (AB, MATH, PODUNK U, 1907) where, for instance, the institution is defined as the data between the second and third commas. In such cases no encoding or special tagging of input data is necessary. Of course, care must be taken to insure that the "normal" structure is adhered to.

APPLICATION AND IMPLEMENTATION

The straightforward basic concepts of the system often permit the establishment of a new information project entirely by the prospective user. Certain basic principles, however, must be observed in the design of a project in order to insure an effective implementation. These involve the recognition of the record-field-subfield relationships and, of course, an appreciation of the fact that, regardless of how sophisticated are the manipulation techniques employed, it is not possible to get optimum usage from poor data or to retrieve information that is not available. Though this should go without saying, the apparent lack of such an appreciation by our customers has been one of the major problems that we have encountered in the use of the system. Perhaps an inevitable disadvantage of any general, easy to apply information processing technique is that it is likely, as our system has, to encourage the proliferation of inadequately planned, ill conceived information projects. Our experience has indicated that there is simply no easy substitute for an intelligent approach to information storage and retrieval problems.

It is for this reason that we emphasize to our customers the necessity for an intelligent approach and urge that (at least) these steps (figure 3) be followed in the establishment of a VIP file project:

1. Definition of purpose—Both short-range and long-range purposes of the file should be fully defined. This may best be accomplished by definitive description of all outputs, both routine (recurring) and special (sporadic), that may be desired from the file.
2. Determination of inputs—The inputs needed to produce the desired outputs are determined from the definitive descriptions of these outputs.
3. Review of feasibility of implementation—The feasibility of obtaining, inputting, and processing the necessary data should be reviewed from the standpoint of automation with respect to economics, timing, volume of data, programming, keypunching, machine utilization and other pertinent factors.
4. Design of file structure—Once the feasibility of application of VIP to a data project has been established, the structure of the information files can be designed. This involves the definition with respect to field assignments of the various records to be included, and the order of these records in the files. These definitions depend on the input source, the logical grouping of information, normal retrieval modes, and existing programs. Assignment of fields should be established in a record glossary containing an entry for each type of information to be recorded in each of its fields. Definition of file order requires only the designation of the fields and subfields to be used as keys in sequencing the records.

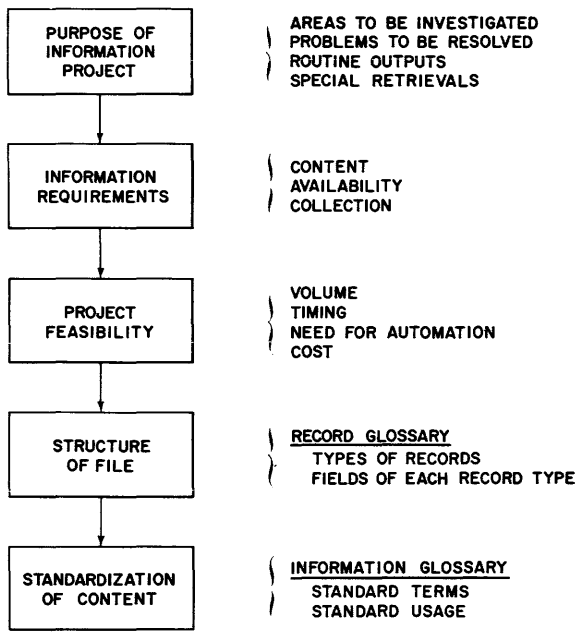


Figure 3. Establishment of an Information Project.

5. Standardization of terminology—The final step in the establishment of a file is the determination of the form in which information is to be recorded in it. Although VIP can accept any kind of information—natural language, codes, mnemonic abbreviations, etc.—the retrieval of specific information is dependent on specific values. For this reason it is desirable that standards of terminology and nomenclature be used for the recording of information. These standards should be established in an information glossary containing an entry for each type of information indicating the exact values to be used for recording.

For information to be entered into the system, it must undergo two preparatory stages—first, editing in accordance with the record and information glossaries, and second, placement on a proper medium (EAM cards) for computer input.

The editing process (figure 4) depends on the source information. At worst it entails a paraphrase transcription of source information

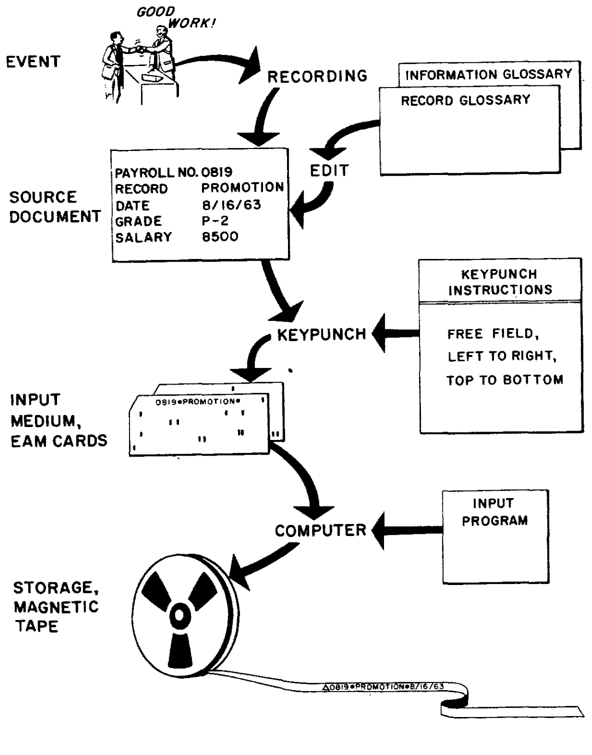


Figure 4. Entry of Information.

into appropriate records. Data may be entered formally as distinct fields or subfields, or informally as textual notes. As indicated earlier, the language used may be mnemonic and therefore easily interpreted without extensive reference to code manuals. Also, much of the source information normally occurs in properly structured form and needs little or no editing.

Input to the system is via EAM cards which are used only as a vehicle for computer entry. The transfer of information from source documents to cards is done in a manner dependent on the form of the data. It is normally punched in a packed, free field form; that is, the data areas are not specifically allotted, and items of information are identified by sequence and special separator characters. The data are usually picked up from the source documents, reading in the normal sequence—left-to-right, top-to-bottom of a page. Various techniques employed for punching efficiency include the use of a diagonal (/) essentially as a ditto mark, to indicate that the contents of a field are identical to those of the corresponding field of the preceding record; and the use of parameters and special punches which cause the insertion of a series of identical field values into each of a sequence of records with the values punched only when they change.

A possible disadvantage of a generalized approach to information processing is the probable sacrifice of efficiency. A tailor-made program to perform a specific task will almost certainly be more efficient. In the case of VIP, the generality of information format, particularly the variability of field length, has not been attained without cost. In order to isolate particular pieces of information, character-by-character scan, a relatively slow and cumbersome operation, has been employed. To speed this operation, a dual technique—high-order and low-order scan—has been utilized (figures 5 and 6). To obtain the first character of a subfield the initial entry is to the high-order scan, which operates on five alphanumeric characters (one computer word) at a time to locate the field, before switchover to the low-order scan, which operates on only one character at a time. Subsequent characters are obtained by direct entry to the low-order scan.

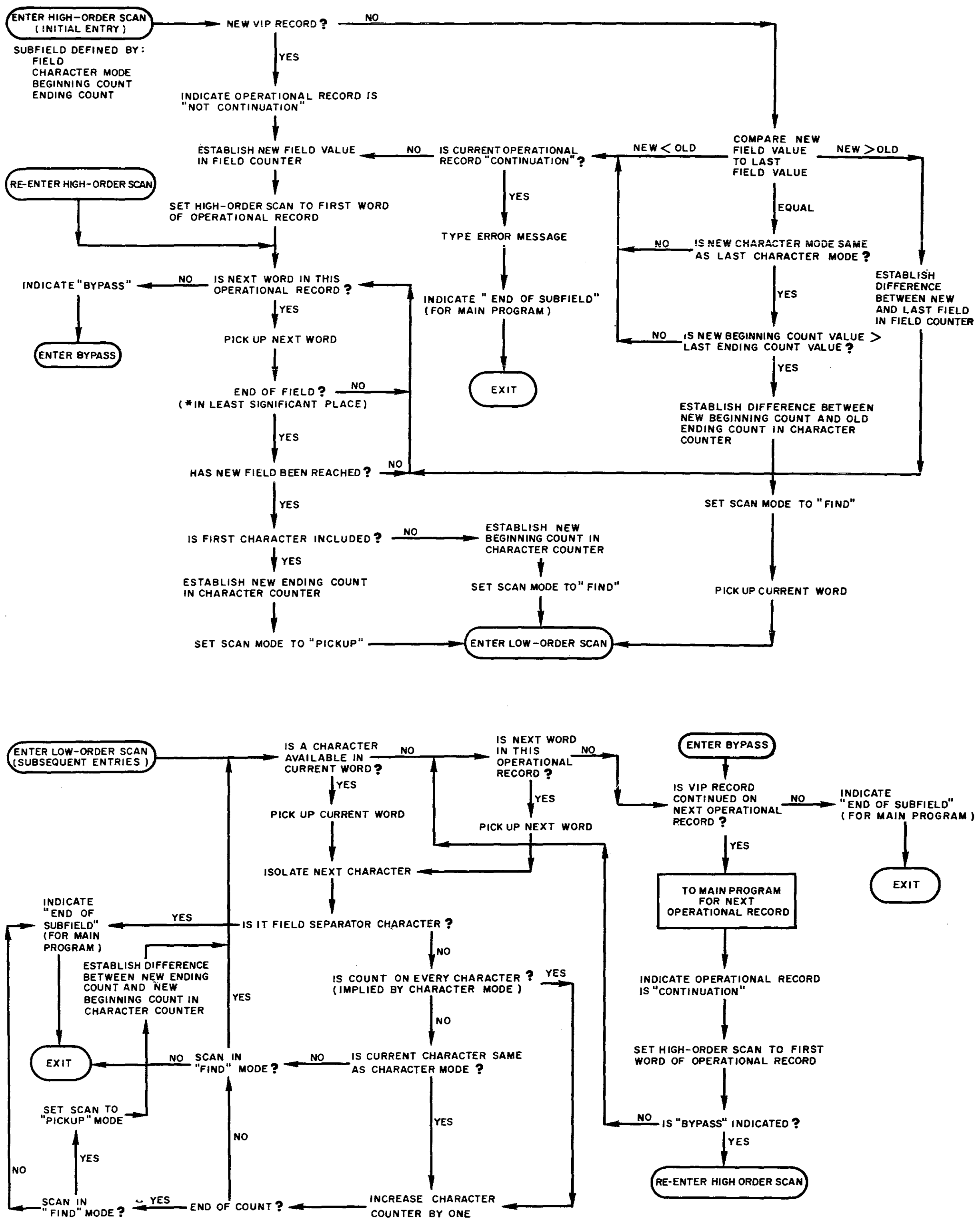

ENTER HIGH-ORDER SCAN (INITIAL ENTRY)
SUBFIELD DEFINED BY: FIELD CHARACTER MODE BEGINNING COUNT ENDING COUNT
NEW VIP RECORD ?
NO
YES
INDICATE OPERATIONAL RECORD IS "NOT CONTINUATION"
ESTABLISH NEW FIELD VALUE IN FIELD COUNTER
IS CURRENT OPERATIONAL RECORD "CONTINUATION" ?
TYPE ERROR MESSAGE
INDICATE " END OF SUBFIELD" (FOR MAIN PROGRAM)
EXIT
COMPARE NEW FIELD VALUE TO LAST FIELD VALUE
NEW < OLD
NEW > OLD
EQUAL
IS NEW CHARACTER MODE SAME AS LAST CHARACTER MODE ?
ESTABLISH DIFFERENCE BETWEEN NEW AND LAST FIELD IN FIELD COUNTER
IS NEW BEGINNING COUNT VALUE > LAST ENDING COUNT VALUE ?
ESTABLISH DIFFERENCE BETWEEN NEW BEGINNING COUNT AND OLD ENDING COUNT IN CHARACTER COUNTER
RE-ENTER HIGH-ORDER SCAN
SET HIGH-ORDER SCAN TO FIRST WORD OF OPERATIONAL RECORD
INDICATE "BYPASS"
ENTER BYPASS
IS NEXT WORD IN THIS OPERATIONAL RECORD ?
PICK UP NEXT WORD
END OF FIELD ? (*IN LEAST SIGNIFICANT PLACE)
HAS NEW FIELD BEEN REACHED ?
SET SCAN MODE TO "FIND"
IS FIRST CHARACTER INCLUDED ?
ESTABLISH NEW BEGINNING COUNT IN CHARACTER COUNTER
ESTABLISH NEW ENDING COUNT IN CHARACTER COUNTER
SET SCAN MODE TO "FIND"
PICK UP CURRENT WORD
SET SCAN MODE TO "PICKUP"
ENTER LOW-ORDER SCAN
ENTER LOW-ORDER SCAN (SUBSEQUENT ENTRIES)
IS A CHARACTER AVAILABLE IN CURRENT WORD ?
PICK UP CURRENT WORD
IS NEXT WORD IN THIS OPERATIONAL RECORD ?
PICK UP NEXT WORD
ISOLATE NEXT CHARACTER
IS IT FIELD SEPARATOR CHARACTER ?
INDICATE "END OF SUBFIELD" (FOR MAIN PROGRAM)
ESTABLISH DIFFERENCE BETWEEN NEW ENDING COUNT AND NEW BEGINNING COUNT IN CHARACTER COUNTER
IS COUNT ON EVERY CHARACTER ? (IMPLIED BY CHARACTER MODE)
IS CURRENT CHARACTER SAME AS CHARACTER MODE ?
SCAN IN "FIND" MODE ?
EXIT
SET SCAN TO "PICKUP" MODE
SCAN IN "FIND" MODE ?
END OF COUNT ?
INCREASE CHARACTER COUNTER BY ONE
ENTER BYPASS
IS VIP RECORD CONTINUED ON NEXT OPERATIONAL RECORD ?
INDICATE "END OF SUBFIELD" (FOR MAIN PROGRAM)
EXIT
TO MAIN PROGRAM FOR NEXT OPERATIONAL RECORD
INDICATE OPERATIONAL RECORD IS "CONTINUATION"
SET HIGH-ORDER SCAN TO FIRST WORD OF OPERATIONAL RECORD
IS "BYPASS" INDICATED ?
RE-ENTER HIGH ORDER SCAN

Figure 5. Character Scan Flow Diagram.

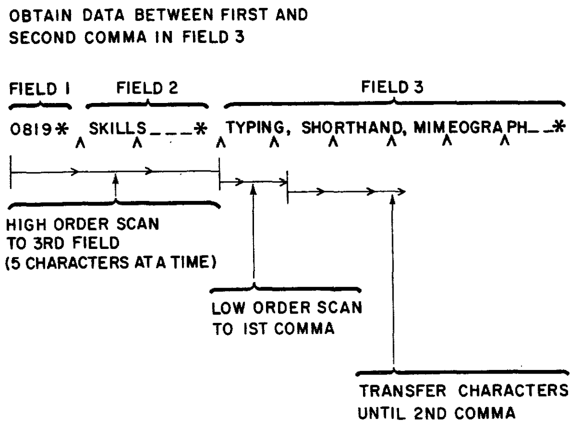

OBTAIN DATA BETWEEN FIRST AND SECOND COMMA IN FIELD 3
FIELD 1
FIELD 2
FIELD 3
0819* SKILLS___* TYPING, SHORTHAND, MIMEOGRAPH__*
HIGH ORDER SCAN TO 3RD FIELD (5 CHARACTERS AT A TIME)
LOW ORDER SCAN TO 1ST COMMA
TRANSFER CHARACTERS UNTIL 2ND COMMA

Figure 6. Subfield Isolation.

This technique permits much of the scanning to be performed relatively rapidly.

A key process in the manipulation (sort, merge, logical selection) of file records is the determination of relative field sizes. This is complicated in the VIP system by the variable field length, the normalized (left-justified) manner of storing information in a field, and the intermixing of alphabetic and numeric information. An interesting technique has been developed for comparison and determination of relative sizes of two fields of normalized alphanumeric data. Once inequality has been established via high-order scan, this technique (figures 7 and 8) involves a left-to-right character scan of both fields which continues until relative sizes are established. The character comparison logic mode is switched depending upon whether or not characters are numeric.

Memory allocation for combinations of descriptor values is accomplished using a branched locator index to a storage area containing the variable length descriptors (figure 9), a technique similar to that of de la Briandais[4]. Each index entry represents one value in the descriptor table with a possible two-way branch—either to the next or to another indicated index entry. An index entry consists of three parts, the functions of which are:

m = branch logic modifier

v = descriptor value location

l = locator index branch location

Both descriptor values and indices are laid down in order of occurrence, thus permitting utilization of contiguous memory cells. The technique also permits multiple usage of individual descriptors.

The development of a cycling technique for performing iterative procedures on variable length sequences of fields or subfields within a record (figure 10) has permitted such procedures to be specified with a single set of parameters. Only the beginning field for the procedure and the displacement factor between iterations are specified. This technique permits the individual processing of items of information stored multiply and in an unordered fashion within records or fields.

The linking together of elemental processors to perform a complex retrieval gave rise to a problem in passing tape unit assignment and file disposition along to succeeding processors. A straightforward tape unit scheduler program (figure 11) was designed which modifies the input/output tape control system by presetting certain parts of it according to a table kept in core. This scheduler table contains three entries for each tape unit: status, file name, and close option. The use of this technique has resulted in a considerable increase in operating effectiveness.

The system has been implemented on an IBM 7070, a fixed word (10 decimal digits or 5 alphanumeric characters) computer, and its adaptation to variable length information processing has required the use of certain special techniques. The storage of VIP records on magnetic tape is via operational records of up to 200 computer words in length (figure 12). Each VIP record occupies an integral number of operational records. The first word of each operational record is a bookkeeping word (ten decimal digits plus sign) which contains the record length in computer words, a sequence number establishing which operational record of a VIP record it is, and an "additional record" flag which indicates whether the VIP record is continued on another operational record. All other words of operational records contain five alphanumeric characters each. This technique permits the storage and processing of VIP fields and records of almost unlimited

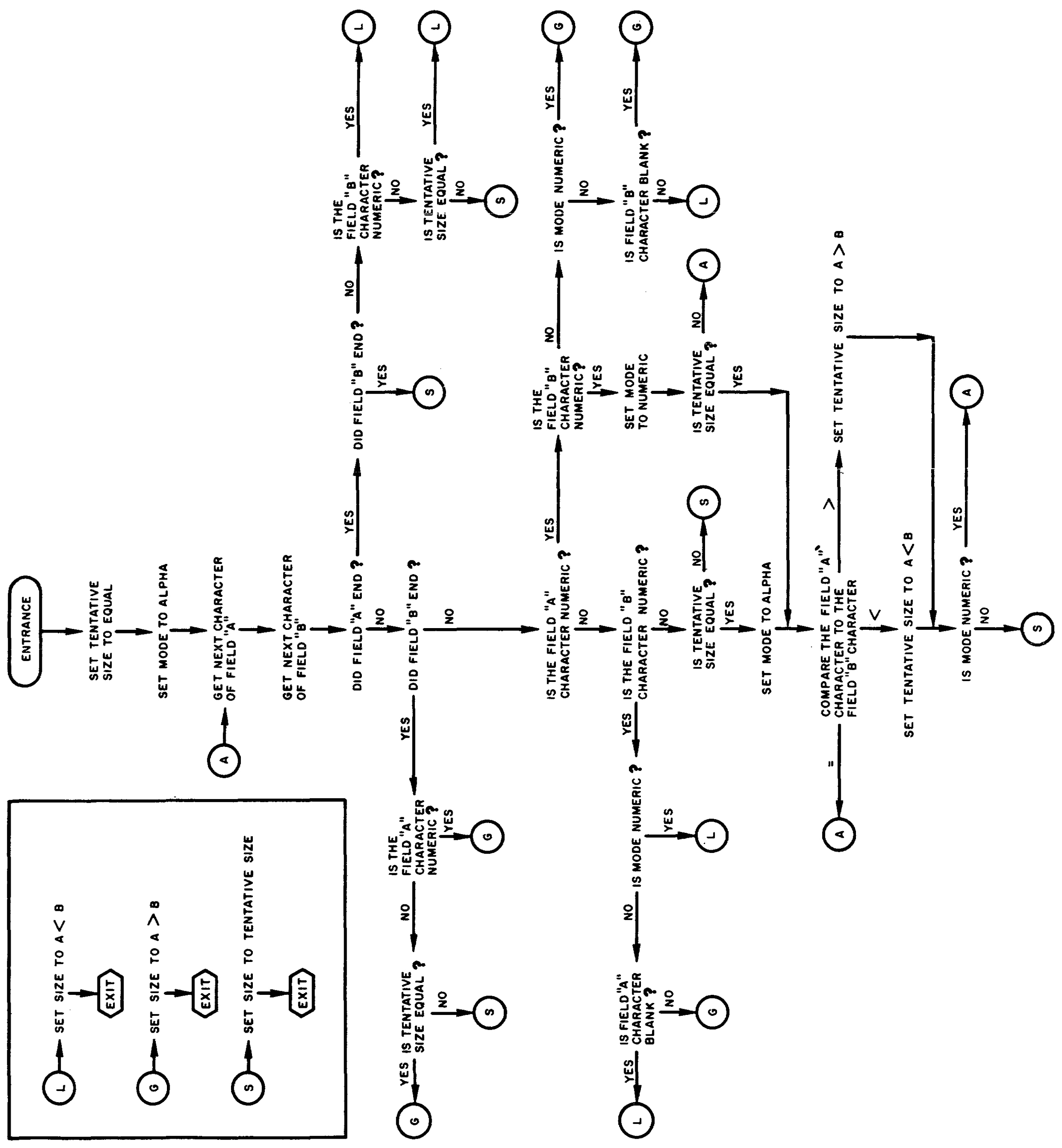

L SET SIZE TO A < B
EXIT
G SET SIZE TO A > B
EXIT
S SET SIZE TO TENTATIVE SIZE
EXIT
ENTRANCE
SET TENTATIVE SIZE TO EQUAL
SET MODE TO ALPHA
A
GET NEXT CHARACTER OF FIELD "A"
GET NEXT CHARACTER OF FIELD "B"
DID FIELD "A" END ?
YES
DID FIELD "B" END ?
NO
IS THE FIELD "B" CHARACTER NUMERIC ?
YES
L
NO
IS TENTATIVE SIZE EQUAL ?
YES
L
NO
S
YES
S
NO
DID FIELD "B" END ?
YES
IS THE FIELD "A" CHARACTER NUMERIC ?
YES
G
NO
IS TENTATIVE SIZE EQUAL ?
YES
G
NO
S
NO
IS THE FIELD "A" CHARACTER NUMERIC ?
YES
IS THE FIELD "B" CHARACTER NUMERIC ?
NO
IS MODE NUMERIC ?
YES
G
NO
IS FIELD "B" CHARACTER BLANK ?
YES
G
NO
L
YES
SET MODE TO NUMERIC
IS TENTATIVE SIZE EQUAL ?
NO
A
YES
NO
IS THE FIELD "B" CHARACTER NUMERIC ?
YES
IS MODE NUMERIC ?
NO
IS FIELD "A" CHARACTER BLANK ?
YES
L
NO
G
YES
L
NO
IS TENTATIVE SIZE EQUAL ?
NO
S
YES
SET MODE TO ALPHA
COMPARE THE FIELD "A" CHARACTER TO THE FIELD "B" CHARACTER
=
A
>
SET TENTATIVE SIZE TO A > B
<
SET TENTATIVE SIZE TO A < B
IS MODE NUMERIC ?
YES
A
NO
S

Figure 7. Field Comparison Flow Diagram.

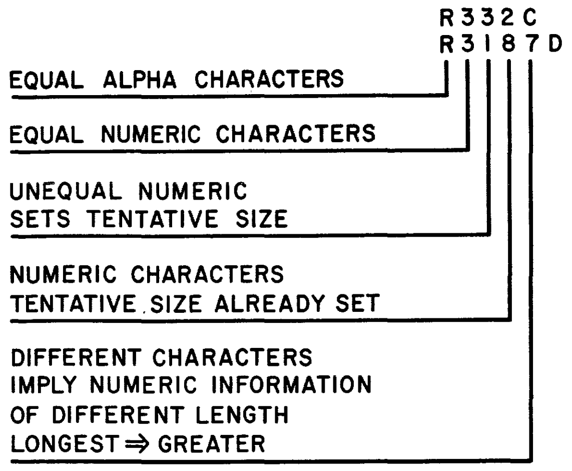

R332C
R3187D
EQUAL ALPHA CHARACTERS
EQUAL NUMERIC CHARACTERS
UNEQUAL NUMERIC SETS TENTATIVE SIZE
NUMERIC CHARACTERS TENTATIVE SIZE ALREADY SET
DIFFERENT CHARACTERS IMPLY NUMERIC INFORMATION OF DIFFERENT LENGTH LONGEST ⇒ GREATER

Figure 8. Example of Field Comparison.

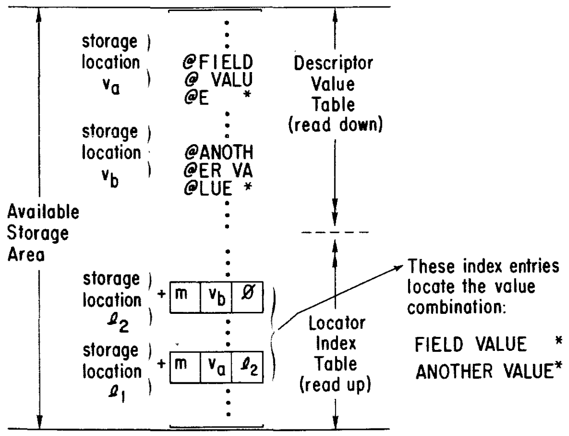

storage location v_a
@FIELD
@ VALU
@E *
Descriptor Value Table (read down)
storage location v_b
@ANOTH
@ER VA
@LUE *
Available Storage Area
storage location ℓ_2
m
v_b
Ø
storage location ℓ_1
m
v_a
ℓ_2
Locator Index Table (read up)
These index entries locate the value combination:
FIELD VALUE *
ANOTHER VALUE*

Figure 9. Storage Allocation Layout.

length. The identification of fields within a record is by relative position with respect to previous fields. Each field contains an integral number of computer words. The word immediately following the bookkeeping word of the first operational record of a VIP record is the beginning of the first field of the VIP record. This field continues until the occurrence of a word with an asterisk (*) in its least significant place which signifies the end of the field. If the record contains additional fields they each begin with the word immediately following the last word of the preceding field

FIELD	VALUE	DESCRIPTION	
1	0819	PAYROLL NUMBER	
2	EDUCATION	RECORD TYPE	
3	CITY COLLEGE, N.Y.	NAME	
4	BS/1949	DEGREE/YEAR	SCHOOL
5	STATISTICS	MAJOR	
6	COLUMBIA U	NAME	
7	MA/1951	DEGREE/YEAR	SCHOOL
8	MATHEMATICS	MAJOR	
9	U OF CALIFORNIA	NAME	
10	PHD/1955	DEGREE/YEAR	SCHOOL
11	MATHEMATICS	MAJOR	

Figure 10. Multiple Items in a Record.

and continue until an asterisk is sensed in the least significant position of a word. Perhaps the necessity for utilizing such techniques indicates that this type of system could have been more easily implemented on a variable word length machine with a sufficiently large memory.

CONCLUSION

Although there are disadvantages, and some serious ones, in utilizing a general approach to information processing, we are convinced that it has been the right direction for us, and we recommend it as the right approach especially for any information processing group which has a limited programming staff and many customers to satisfy. It has permitted us to devote our programming effort to the continued expansion of our capabilities to the point where our current complex of processor programs provides a really extensive, flexible and powerful system available for use on all our files (figure 13). This would never have been possible had we taken a tailored approach to systems design.

We do not consider that we have achieved the ultimate information processing system. We are continuing with the development of general processing capabilities. Currently underway is an attempt at extending still further the flexibility of information organization, as well as the development of a more convenient means (language) for communicating with and utilizing the system.

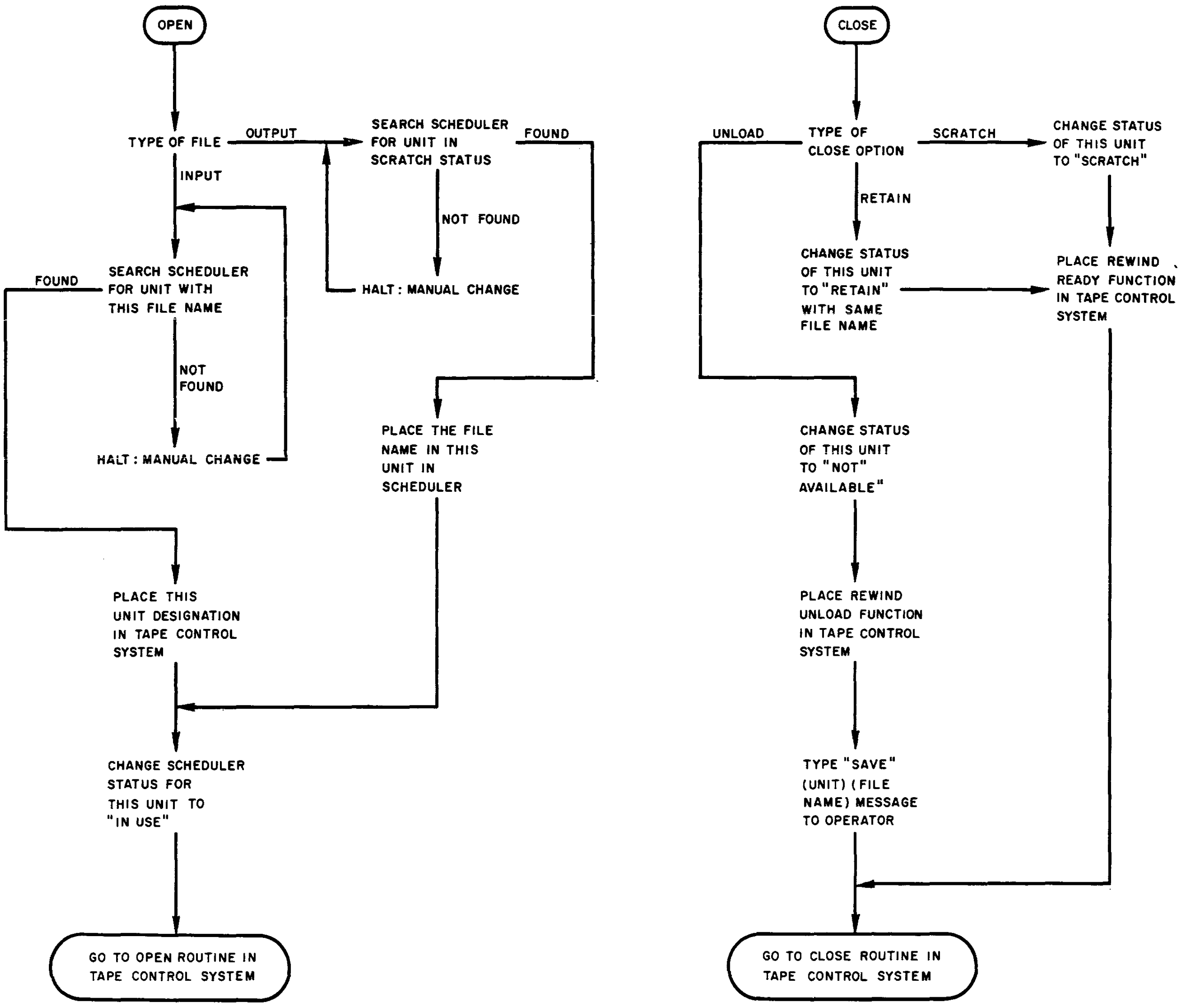

OPEN
TYPE OF FILE
OUTPUT
INPUT
SEARCH SCHEDULER FOR UNIT IN SCRATCH STATUS
FOUND
NOT FOUND
HALT: MANUAL CHANGE
SEARCH SCHEDULER FOR UNIT WITH THIS FILE NAME
FOUND
NOT FOUND
HALT: MANUAL CHANGE
PLACE THE FILE NAME IN THIS UNIT IN SCHEDULER
PLACE THIS UNIT DESIGNATION IN TAPE CONTROL SYSTEM
CHANGE SCHEDULER STATUS FOR THIS UNIT TO "IN USE"
GO TO OPEN ROUTINE IN TAPE CONTROL SYSTEM
CLOSE
UNLOAD
TYPE OF CLOSE OPTION
SCRATCH
CHANGE STATUS OF THIS UNIT TO "SCRATCH"
RETAIN
CHANGE STATUS OF THIS UNIT TO "RETAIN" WITH SAME FILE NAME
PLACE REWIND READY FUNCTION IN TAPE CONTROL SYSTEM
CHANGE STATUS OF THIS UNIT TO "NOT AVAILABLE"
PLACE REWIND UNLOAD FUNCTION IN TAPE CONTROL SYSTEM
TYPE "SAVE" (UNIT) (FILE NAME) MESSAGE TO OPERATOR
GO TO CLOSE ROUTINE IN TAPE CONTROL SYSTEM

Figure 11. Tape Unit Scheduling.

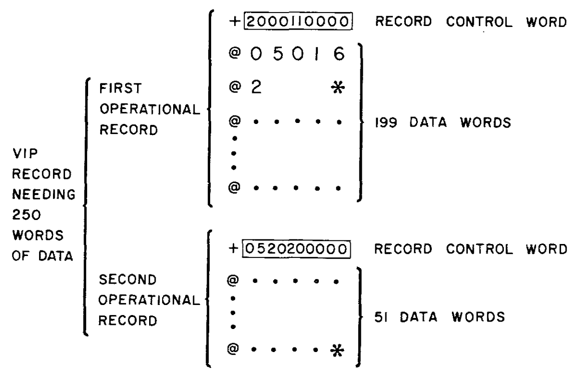

+ 2000110000 RECORD CONTROL WORD
@ 0 5 0 1 6
@ 2 *
@ . . . . .
@ . . . . .
199 DATA WORDS
FIRST OPERATIONAL RECORD
VIP RECORD NEEDING 250 WORDS OF DATA
+ 0520200000 RECORD CONTROL WORD
@ . . . . .
@ . . . . *
51 DATA WORDS
SECOND OPERATIONAL RECORD

Figure 12. Operational Records.

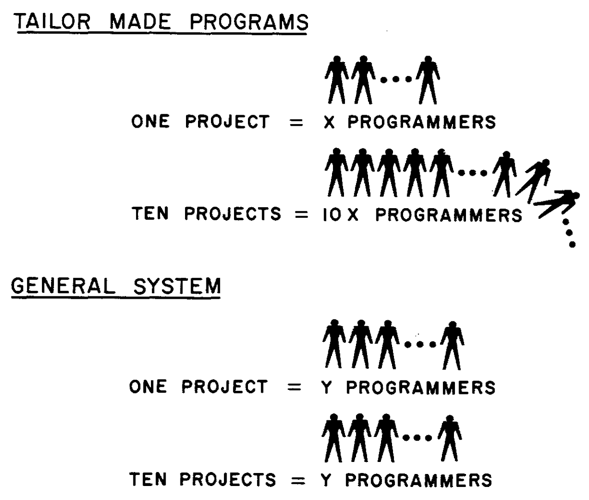

TAILOR MADE PROGRAMS
ONE PROJECT = X PROGRAMMERS
TEN PROJECTS = 10 X PROGRAMMERS
GENERAL SYSTEM
ONE PROJECT = Y PROGRAMMERS
TEN PROJECTS = Y PROGRAMMERS

Figure 13. Programmer Utilization.

REFERENCES

1. W. C. McGEE, "Generalization: Key to Successful Electronic Data Processing", Journal of the Association for Computing Machinery, January 1959.
2. J. A. POSTLEY and T. D. BUETELL, "Generalized Information Retrieval and Listing System", Datamation, December 1962.
3. W. D. CLIMENSON, "RECOL—A Retrieval Command Language", Communications of the Association for Computing Machinery, March 1963.
4. R. DE LA BRIANDAIS, "File Searching Using Variable Length Keys", 1959 Proceedings of the Western Joint Computer Conference.

A SEARCH MEMORY SUBSYSTEM FOR A GENERAL-PURPOSE COMPUTER

Albert Kaplan
UNIVAC Division
Sperry Rand Corporation
St. Paul, Minnesota

I. INTRODUCTION

The search memory provides an extremely powerful tool for data-processing applications in which files of data are manipulated—applications such as data correlation, information retrieval and data analysis. For many such problems, a moderate size search memory can increase the performance of a general-purpose computer by several orders of magnitude. This paper describes the system design of a search memory subsystem integral to a general-purpose computer. The unit will operate in much the same manner as a conventional arithmetic unit, receiving data directly from the main memory, and placing the results of a search in the accumulator. This generalized design is independent of the particular magnetic memory element used and is not constrained by any particular computer organization. However, values have been assigned to the various parameters in the system so that the concepts can be concretely described. The techniques and concepts presented in this paper are the result of a company-sponsored program. The following sections present an assumed general-purpose computer structure, the design of a search memory subsystem, and possible implementation. Appendix A presents the basic principles of the search memory organization.

II. SEARCH MEMORY SUBSYSTEM

To establish a sound basis for the design of a search memory subsystem, a number of typical applications were analyzed to determine the functional requirements which such applications impose upon the search memory. The primary application was for an information retrieval process suitable for a library, or an inventory control system. Major requirements for such an application are equally search, variable "AND"—"OR"—"NOT" logic between field searches, and the ability to analyze the results of partial searches so that search criteria can be "loosened" or "tightened" according to the number of matches. Data correlation applications require quantitative searches such as *greater than, less than,* and *between limits.* Finally, statistical analysis of data sets requires *next higher* and *next lower* as well as the other quantitative criteria. In addition, the *number of matches* is required. The composite of these requirements provided the basis for this subsystem.

In order to keep the design of the search memory subsystem as generally applicable as possible, the less assumed about the computer structure, the better. Thus, the structure shown in Figure 1 is postulated, consisting of a main computer memory which communicates via a memory register with the search memory subsystem, an accumulator, the arithmetic, control, and I/O sections. In concrete terms, a word length of 36 bits and a main memory cycle time of 4 microseconds is also postulated.

The search memory subsystem contains 4096 words of 72 data bits each. Searches are carried

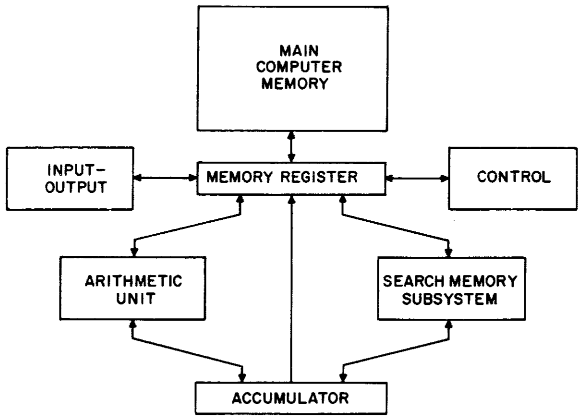


Figure 1. Postulated Computer Structure.

out on 36 bits, and the remaining 36 are entered into the A register on a match condition. The following search criteria are provided:

Equal
Greater than or equal
Less than or equal
Between limits
Next higher
Next lower
Not equal
Not greater than or equal
Not less than or equal
Not between limits

A completely variable field structure is provided so that multiple fields of variable lengths and positions within the search word can be specified under program control. A different search criterion can be applied to each field specified and the results of each field interrogation "AND"ed or "OR"ed with the cumulative result of previous interrogations. Similarly, the results of several full word searches can be "AND"ed or "OR"ed. Execution time of the search operation varies with the number of bits searched, the number of fields, and the search criteria. For a simple operation, such as a 36-bit equality search, the execution time is 8.0 microseconds.* A very complex six-field, six-bits-per-field quantitative search will require 16.0 microseconds. An average search will require 12 microseconds. An interrupted search mode is provided for interrupting the search after a specified field has been interrogated and for entering the approximate number of matches into the A register. Under program control, the search may be either resumed or terminated, in which case the match data is retrieved as in the normal mode. The exact number of matches may be requested under program control. In multiple matches, the first match data word is automatically placed in the A register at the completion of the search. Remaining match data words are requested sequentially under program control. Three modes of loading the search memory are provided. Data may be written into either or both halves of the 72-bit word at a location specified by the program, or at the location of the last match. The third mode facilitates sequential loading of the entire search memory as would be required at the initiation of the program. Figure 2 shows a block diagram of the search memory subsystem.

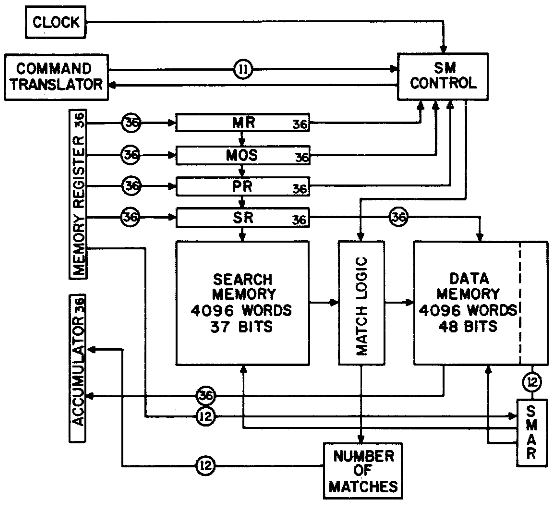


Figure 2. Search Memory Subsystem.

Major components of the search memory subsystem are the search memory, with a postulated size of 4096 words, 37 bits per word (36 bits hold search data, and the remaining bit is used as a dummy bit for the "NOT" searches); a 4096-word, 48 bits per word, word-organized data memory (36 bits hold match data, and the remaining 12 hold the address of that word); a search register to hold the search word; three control registers, PR, MOS, and MR; the match logic which interprets the signals emanating

* The operation times quoted are for the UNIVAC Bicore Non-Destructive Readout thin film memory element. They will change according to the memory element used.

from the search memory; an address register used for writing information in memory; a number-of-matches unit which produces both a high-speed approximate count and a slow-speed exact count; and control and timing circuitry.

Four data registers are used with the search memory. Search register (SR) holds the 36-bit search word. Parameter register (PR) is used for variable length field designation. Coding of PR is done according to the following rules. The first bit of each field is set to a "1", and all successive bits within the field are set to "0".

The sample PR code below would indicate eight search fields of various lengths.

10000000	1000000	100000	100	10	1	100	100000
Field 1	Field 2	Field 3	Field 4	Field 5	Field 6	Field 7	Field 8

A 36-bit mask register (MR) enables masking within specified fields. A "1" in MR enables the search of the corresponding bit in SR, a "0" masks that bit.

The mode-of-search register (MOS) is used to designate the search criteria for the various fields specified in the parameter register. This register holds up to six mode statements, of six bits each. Each mode statement specifies the search criteria for one of the search fields and is composed of four quantities. The first bit designates the logical connective to the previous searches, "AND" or "OR". The next three bits specify the search criteria: *equality, greater than* or *equal, less than* or *equal, between limits, next higher,* or *next lower.* The next bit specifies the negative of the search criteria. The last bit designates that the search is to be interrupted after that field has been interrogated and the approximate number of matches entered in the accumulator. This feature cannot be applied to the last field. The first bit of MOS is used to inhibit the clearing of the match logic so that another search can be performed and their results combined. If more than six fields have been specified in the parameter register, the last mode statement is applied to all successive fields.

Instructions to be used with the search memory are listed below; (M) represents the contents of a main memory location specified in the instruction:

Mnemonic	*Function*	*Time*
LPR	(M) → PR	4 μsec
LMS	(M) → MOS	4 μsec
LMR	(M) → MR	4 μsec
SRC	(M) → SR, search, first match data → A; clear load designator; if no match, skip next instruction; SMAR holds match address	(See below)
NMD	Next match data → A; if no more, skip next instruction; SMAR holds match address; if interrupted search, terminate search and first match data → A	4 μsec
NMA	Exact number of matches → A	4 μsec
SMA	(M) → SMAR	4 μsec
SLD	Set load designator, clear SMAR	4 μsec
WSM	(M) → SR (SR) → SM(SMAR)	16-32 μsec (See below)
WDM	(M) → SR (SR) → DM(SMAR), (SMAR) + 1 → SMAR if load designator is set	4 μsec
RIS	Resume interrupted search	(See below)

The first three instructions, LPR, LMS, and LMR, are used to enter words from main memory into the search memory control registers. Once they have been set up, an SCR instruction loads the search register and initiates the search according to the instructions in PR, MR, and MOS. Upon completion of the search, a flip-flop associated with each word will be set either to match or mismatch. These flip-flops are scanned, and the first match is located. The corresponding location in the word-organized data memory is driven to obtain the 36-bit match data and the 12-bit address of the match. The address is placed in SMAR, and the match data placed in the accumulator. If there are no matches, the next instruction is skipped. The search memory address register (SMAR) holds the match address so that new information can be written into the search memory or the data memory at this location. Next, the NMD instruction is applied repeatedly to retrieve all the matches for this search. Normally, this would be done in a loop with the skip next instruction feature used to exit from the loop when all the matches have been processed. If an interrupted search has been specified, NMD is used to terminate the search and retrieve the match data. If the program elects to continue the interrupted search, the RIS instruction is used.

The time required to execute a search operation is a function of the number of bits searched, the mode of search, and the number of fields. Total search time is 0.1 microsecond per bit searched for *equality, greater than* or *less than,* plus 1.1 microseconds per bit searched for *next higher* or *next lower,* plus 1.0 microsecond per field searched (except first field), plus 4.0 microseconds to get the next instruction. The resulting time must then be rounded to the next higher multiple of 0.4 microseconds to regain synchronization with the control section. Thus, for a 36-bit equality search, the time is 8.0 microseconds. For a fairly complex six-field, six-bit-per-field, greater or less than search, the search time is 16.0 microseconds. When an interrupted search is specified, the above formula will apply for the portion of the word searched plus 24 microseconds to generate the "approximate" number of matches. A search can be initiated 16 microseconds after a write. However, 32 microseconds are required before another write. These times apply only to the Bicore thin film memory element.

In addition to the approximate number of matches generated in an interrupted search, an exact match counter is provided for statistical data analysis. The approximate number (10%) is generated in 24 microseconds with an analog summer and A/D converter. Every word in the search memory must be scanned for an exact count which requires 410 microseconds. This time can be reduced by scanning several words at a time. The NMA instruction is used to place the exact number of matches in the accumulator. The remaining instructions are used for writing in the search memory and data memory. The SMA instruction places an address in SMAR. The WSM and WDM copy a word from main memory into the search memory or the data memory via the search register at a location specified by SMAR. A block loading feature provides for rapid loading of the search memory and data memory at the start of the program. The SLD instruction clears SMAR to location zero and sets a special load designator. A word is then written into location zero of the search memory. Then a WDM instruction loads location zero in the data memory. However, since the load designator is set, SMAR is automatically incremented and the process can be repeated to load location one, two, three, etc.. On the next search instruction the load designator is automatically cleared.

For each word in the search memory, final determination whether that word is a match or not is performed in the match logic. Match determination is performed in the following manner.

The memory is driven bit-serially, starting from the highest order unmasked bit location. Each bit-line is driven to either a "1" or a "0" corresponding to the contents of the search register at that particular bit location. If the bit in a particular word is already in the state to which it is being driven, there will be no output. Consequently, that bit will be known to be equal to the corresponding bit in the search register. If that bit is in opposition to the state to which it is being driven, there will be an output on the word line, the polarity of which depends upon the stored bit (see Appendix A). The outputs on the word line indicate that the bit in memory does not compare with the corresponding bit in the search register. Two inputs can be provided to the match logic by controlling time and duration of bit drive on each bit line, and by inverting and time-gating the word line outputs onto separate lines from the sense amplifier according to the mode of search indicated in the MOS register. An output on one line indicates that search criterion ($=$, $\geq$, $\leq$) is satisfied for the particular bit. An output on the other line indicates that the mismatch was such that the search criterion failed at that bit location. Absence of an output from the sense amplifier indicates that the bit memory is the same as the bit in the search register, a condition which indicates neither passing nor failing of the search criterion.

The match logic accepts the above outputs from the sense amplifier, mode of search information from the MOS register, timing information from the memory driver control circuits, and performs two basic functions:

1) Inhibiting the word line noise brought about by activation and deactivation of the digit drivers.
2) Determining field comparison or noncomparison according to the following rules:

   Determination of field comparison (i.e., compliance with the search criterion) can be made upon detection of the first bit found to differ from the corresponding SR bit. (Each bit of a field bears greater numerical significance than the sum of the succeeding lower order bits.)

   If no bits in the field are found to differ from their corresponding bits in SR, the field has complied with the search criterion.

A schematic of the match logic is shown in Figure 3. To simplify description, the diagram shows noninverting logic. The sense amplifier accepts signals from the sense line and, on the basis of control information from the MOS register, issues a pulse on either the pass or the fail lines. If, for example, the MOS register specifies greater than or equal (GT) and the polarity of the signal on the sense line indicates that the SR bit being interrogated is a "0", a

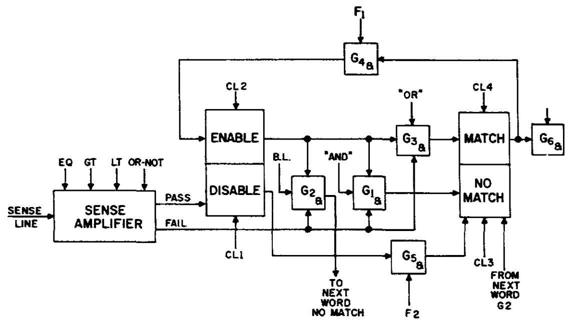


Figure 3. Match Logic.

mismatch signal on the sense line means that the corresponding bit in the search memory word is a "1" and a pulse is issued on the pass line. The match logic must then accept the first signal from the sense amplifier for each field and block all others. To trace the operation of the match logic, consider an equality, greater than or equal, or less than or equal mode of search and an "AND" coupling to the previous interrogations. The match flip-flop contains the cumulative results of the previous interrogations, and the enable flip-flop is set to enable at the start of the field. The field is interrogated bit-serially, starting with the highest order unmasked bit. As long as the bits of the field in this word are identical to those in SR, there will be no signal from the sense amplifier and the match logic will remain quiescent. When the first mismatch is found, a signal will be issued on either the pass or the fail line. If the signal was on the fail line, it will pass through G_1, which is enabled, and set the match flip-flop to no match. Since any successive signals from this field cannot change this no-match state, they are automatically ignored. If, however, the signal was on the pass line, it sets the disable state, and G_1 is blocked. Thus, any successive signals on the fail line cannot past G_1, and the state of the match flip-flop remains as it was before this field was interrogated. This satisfies the "AND" condition. To accomplish the between-limits, the odd numbered words in the search memory are driven for greater than or equal and the even numbered words for less than or equal. The no-match signals in each pair of words are cross-gated via G_2 so that a no-match on either will result in no-match for both. Only a match on both words represents a match on between-limits.

To apply the "NOT" condition to these three search criteria, the roles of the pass and the fail lines are interchanged by reversing the direction of the bit drives and thus the polarities of the signals on the sense line. A pass signal will emerge on the fail line and vice versa. Thus, with the same logic as above, a pass signal from the memory will result in setting the match flip-flop to no-match, and a fail signal will leave the match flip-flop unchanged.

This process breaks down, however, when the search memory word is identical to the search register and no signal appears on either the pass or the fail line. To overcome this, one extra bit is added to each word which is so set that when interrogated, a signal always results on the fail line.

The "OR" function is accomplished by the equivalence of "OR" to "AND-NOT". As in "NOT", the signals on the pass and fail lines are interchanged. G_3 is activated instead of G_1 so that a pass signal, if it is the first signal in the field, will set match and fail will leave the match flip-flop unchanged.

The next higher and next lower searches are the most complex and require two passes through the field. The first pass is a conventional greater-than or less-than search. After this pass is completed, all enable flip-flops are disabled and then G_4 is pulsed to enable only those which were set to match. Then the field is searched again. After each bit is interrogated, a 4096-input "OR" gate (not shown) examines the state of all enable flip-flops. If at least one is still enabled, the disable side sets the no-match state via G_5. If all are disabled, the match state again resets the enable via G_4, *and also* the field in the search register is reset to all "0"s for next-higher or to all "1"s for next-lower. To illustrate this process, consider the following next-higher search. Assume that the first pass has been completed. The states of the enable (E) and match (M) flip-flops are shown after each bit is searched and after resetting with G_4 and G_5.

	$\nearrow$00
Search Register	11011
Word 1	11100
Word 2	11101
Word 3	11110

	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5
	EM EM	EM EM	EM EM	EM EM	EM EM
Word 1	$11 \rightarrow 11$	$11 \rightarrow 11$	$01 \xrightarrow{G_4} 11$	$11 \rightarrow 11$	$11 \rightarrow 11$
Word 2	$11 \rightarrow 11$	$11 \rightarrow 11$	$01 \xrightarrow{G_4} 11$	$11 \rightarrow 11$	$01 \xrightarrow{G_5} 00$
Word 3	$11 \rightarrow 11$	$11 \rightarrow 11$	$01 \xrightarrow{G_4} 11$	$01 \xrightarrow{G_5} 00$	$00 \rightarrow 00$

E represents the state of the enable flip-flop and M the state of the match flip-flop.

III. IMPLEMENTATION

The system design presented above can be implemented with any of the present magnetic nondestructive readout memory elements with minor variations for the electrical characteristics of the particular element. There is no inherent limitation to the size or word length of the subsystem. If the word length of the search memory is greater than that of the main memory, extra instructions can be used to load portions of the various registers. For example, if the search memory were twice as long as the main memory, two instructions would be required to load each register, one for the left half and one for the right half. The data memory need not be the same size as the search memory; it could contain several words for each search memory word. Similarly, the various functions and components within the subsystem can be varied to suit the particular requirements.

Considerable logic circuitry is required for this subsystem (Figure 3). With conventional circuitry, a subsystem of this magnitude would be very unwieldy. However, the fallout technology from aerospace computer developments will enable such a subsystem to be built in a moderate size and at a moderate cost. Microelectronic circuitry, besides having such virtues as high reliability, high environmental tolerance, low power and relatively high speed, is also quite economical when procured in large quantities. A typical package, containing an average of 2.5 logic nodes, measures ¼″ × ⅛″ × 0.035″ and consumes 12 mw of power. Using such elements with a Bicore thin film search memory, and a state-of-the-art word-organized core data memory, the subsystem described above would occupy a unit 2 by 2 by 2.5 feet for a volume of 10 cubic feet.

IV. CONCLUSION

The system design for a search memory subsystem for a general-purpose computer has been presented. The design is as general as possible so as to be independent of memory element and computer structure. This design encompasses many of the functions required for such applications as information retrieval, data correlation, and statistical data analysis, and can increase the capability of the computer for these types of problems by several orders of magnitude.

APPENDIX A

Search Memory Organization

A search memory is a device which simultaneously compares an input word with all words stored in the memory and indicates whether each word satisfies the search criterion. Among the criteria which can be readily implemented are equality, greater than or equal, less than or equal, between limits, next higher, next lower, or the negative of these criteria. The time for a search varies with memory element and organization and ranges upward from 100 nsec.

Several possible organizations for the search memory include searching all bits in parallel, searching bit serial, and various series-parallel modes. The organization which appears to be optimal for this system is the bit serial mode. The reasons for this choice will be developed below. In this organization (Figure A-1) the circles represent nondestructive readout memory elements whose property is essential to the operation of the search memory since all memory elements are interrogated during the search. The rows are words and the columns bit positions. A bi-directional driver is provided for each bit position which is controlled by the contents of the search register. A "1" in the search register results in a positive current pulse on the bit line, a "0" in a negative pulse. Each word has a sense line, with the sense line and memory elements so coupled that a positive current pulse (a "1" in the search register) and a stored "1" will result in a very small output on the sense line. A positive current pulse and a stored "0" result in a large positive signal on the sense line. Similarly a "0" in the search register produces a negative bit current pulse which results in a very small signal for a stored "0" and a large negative signal for a stored "1".

The relationship of the stored information, bit current polarity, and output signal polarity are tabulated in Table A-I below.

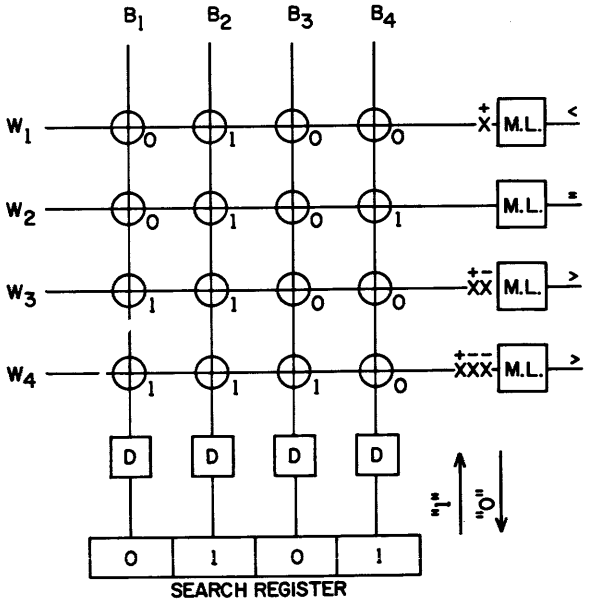


Figure A-1. Search Memory Organization.

TABLE A-I

Stored Infor- mation	Search Register	Bit Current Polarity	Output Signal	
			Switch	Reswitch
1	1	+	0	0
1	0	—	— V	+ V
0	1	+	+ V	— V
0	0	—	0	0

In the sample pattern shown in Figure A-1 the search register contains 0101 which matches word two in the memory. Bit one of the search register is a "0" and produces a negative current pulse on the bit one line. Words three and four contain "1"s in that bit position and signals result on their sense lines. These are denoted by X's. Bit two of the search register is a "1" and produces a positive current pulse on the bit line. Since all four words have a "1" in that bit position, no outputs are produced on the sense lines. Bit three of the search register is a "0" and produces a second output on the word four sense line. Finally, bit four of the search register is a "1" and the positive current pulse produces an output on the sense lines of words one, three, and four. Notice that word two, which matches the search register, has had no outputs on its sense line while the others have had at least one signal. Thus, the matching word can be detected.

To perform a greater than or less than search, the bit serial mode of operation is essential. The basic algorithm is that the highest order bit in which the word in the search memory differs from the search register determines whether that word is greater or less than the contents of the search register. If this bit in the search register is a "1", then the word is less. If it is a "0", then the word is greater than the search memory. This algorithm is easily implemented with this organization because of the bipolar output from the memory element as shown in Table A-I. The memory is interrogated bit-serially starting with the highest order bit, and match logic is implemented on each sense line to interpret the first signal and to ignore all successive signals. With the same example as in Figure A-1, bit one of the search register is a "0", and the negative current pulse produces a negative signal on the sense line of words three and four. This signal is interpreted by the match logic to mean greater than, and all further outputs from words three and four are ignored. Bit two produces no outputs. Bit three produces a negative output on the word four sense line, but this is ignored. Finally, bit four produces positive outputs for words one, three, and four. The signals on the words three and four are ignored as before. However, the positive signal on the word one sense line is intepreted as less than, and word one is also set to ignore all successive signals. Thus in one search, all words in the memory have been quantitatively compared with the word in the search memory. A masking function can be very easily implemented by inhibiting the drivers. With no pulse on the bit line, no signals can result on the sense line, and the bit is effectively masked.

Since a bit-serial mode of operation is required for the quantitative comparisons, this same mode is used for equality. The equality search can be performed on a fully bit-parallel basis. However, this would complicate the control and would require a much more sophisticated sensing mechanism because of lower signal-to-noise ratios.

THE LOGICAL ORGANIZATION OF THE PB 440 MICROPROGRAMMABLE COMPUTER

E. Boutwell, Jr. and E. A. Hoskinson
Research and Engineering Department
Packard Bell Computer
Anaheim, California

INTRODUCTION

The principal classifying feature of the PB 440 is its microprogrammed character. Like other so-called "general purpose" computers, it derives control from a program of instructions or orders which has been previously prepared and stored within the computer memory. Unlike most general purpose computers, however, the PB 440 programmer is permitted to direct the computer at a more basic level of control by logically manipulating the contents of individual registers and flip flops.

Compared to the more sophisticated (and more restrictive) manipulative capabilities of conventional digital computers, the PB 440 instructions may properly be termed "micro-orders". In accomplishing a given data processing operation, an appropriately programmed series of these micro-orders would be executed by the PB 440. A more conventional computer would achieve the same result by activating a succession of logical circuits in a fixed sequence which had been determined by the computer designer. Thus, an important consequence of this micro-programming (or "stored logic") feature is the relative freedom of the programmer to select or devise (macro) instruction sets which are appropriate to his particular problem, rather than being limited to a single set which was compromised by the designer in order to accommodate a wide variety of applications. The logical organization of a computer which provides this feature and to a lesser extent, its programming implications, are the topics of this paper.

Before elaborating on the design, several comments might be made for the benefit of the serious student of microprogramming. The authors' use of the term "microprogramming" appears to agree with the definitions introduced and employed by Wilkes[1, 3] and Glantz[2] and with the one pulse time criterion referred to by Mercer[4] in discussing the elementary nature of micro-orders. A review of the work of earlier authors also indicates that the non-destructively read memory module, used in the PB 440 for the storage of micro-order sequences, replaces the earlier concept of a wired microprogram memory.[1] It provides the capability sought by Glantz[2], for his sub-command Sequence Memory, and later by Grasselli[6], for the Control memory in his "conventional microprogrammed control unit." Finally, it should be stated that, while the virtues and merits of a microprogrammed philosophy of machine organization, expressed by previous workers, influenced the decision to organize the PB 440 control unit in this fashion, following the adoption of the basic philosophy the details of implementation developed naturally and independently.

FUNCTIONAL CHARACTERISTICS

The principal characteristics of the PB 440 are introduced in Table 1. A maximum of 64

TABLE 1
PB 440 CHARACTERISTICS

General Characteristics
- Computer type: micro-order, stored program
- Data handling mode: parallel
- Internal number system: binary
- Timing: synchronous, 1 megacycle logic clock rate

Programming Characteristics
- Instruction format: variable, determined by microprogram (64 micro-orders)
- Word lengths: micro-orders—12 bits
  data word length and format—determined by micro-program
- Index registers: as required by instruction format
- Programmable registers: 4 processing unit registers functionally defined by micro-program

Memory Characteristics
- Main Memory: 4096 words, 24 bits, random access, 5 microsecond cycle time, expandable to 28,672 words in modules of 4096 words each
- Fast Memory: 256 words, 24 bits; random access non-destructive read, 1 microsecond read cycle time, expandable in modules of 256 words each

Input-Output Equipment Options
- Typewriter, 15 characters/sec.
- Paper tape reader, 500 characters/sec.
- Paper tape punch, 110 characters/sec.
- Card reader, 800 cards/minute
- Card punch, 250 cards/minute
- Magnetic tape units 15 KC to 83.3 KC (IBM compatible)
- Line printer, 1000 lines/minute
- Also provision for:
  A-D converters, plotters, display equipment, communication links, hybrid system linkage, mass storage devices, other computers

distinct micro-order types are decoded by the PB 440 control unit for manipulative and control purposes. In addition to an identifying 6-bit micro-order code, a 6-bit modifier field is contained in each micro-order. This basic format is illustrated in Figure 1, where two micro-orders are shown occupying a 24-bit memory word. The modifier fields have been further divided into two 3-bit fields designated R1 and R2. Although the composite 6-bit field has a different significance for some micro-orders, generally, the R1 and R2 octal digits each refer to one of seven hardware registers whose contents may be manipulated in a manner specified by the associated micro code.

The ability to perform logical manipulations on the contents of selected registers permits operations to be conducted on data whose length and number form are determined by the programmer. It was recognized, however, that the execution times resulting from the use of such general capabilities can be improved upon for one or more common types of data formats, if special micro-orders tailored to these formats are included in the design. Character manipulation and arithmetic operations on signed numbers and floating point numbers are three cases which occur so frequently that this type of special treatment is in order. The data formats illustrated in Fig. 1 were selected for these purposes, and it will be noted below during an examination of the micro-order repertoire that operations on data expressed in one of these formats have been facilitated.

A representative number of PB 440 micro-orders have been grouped in Table 2 according to their function. A few general remarks will serve to illustrate the use of each group. The register manipulative operations of the first group may be applied to any of the addressable registers, permitting the role of a register to vary from operation to operation as determined by the programmer. Identified in this group is the ability to manipulate particular fields of data which facilitate the programming of operations on numbers expressed in either the sign-magnitude or floating point formats presented earlier. Like the register manipula-

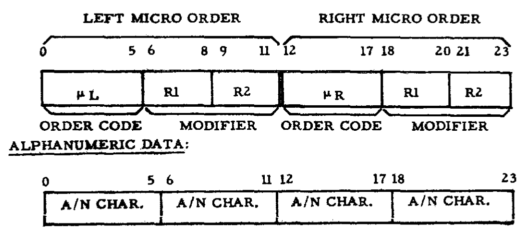


Figure 1. PB 440 Basic Word Formats.

TABLE 2
PB 440 MICRO-ORDER TYPES

GENERAL MANIPULATIVE	
CPL, CPM, CPS, CPX	Copy Logical, Magnitude, Sign, Exponent
CCL, CCM, CCS, CCX	Copy and Complement Logical, Magnitude, Sign, Exponent
CIL, CIX, CDL	Copy and Increment Logical, Exponent, Copy and Decrement Logical
AND, LOR, XOR, EXC	Logical Product, Logical Sum, Exclusive OR, Exchange
SHIFT	
SSL, SDL	Shift Single Length, Double Length
SLS, SLC, SFR	Shift Left Six, Shift Left and Count, Shift Fraction Right
ARITHMETIC	
ADL, ADM, ADS, ADX, ADF	Add Logical, Magnitude, Sign, Exponent, Fraction
AMK, AFK	Add Magnitude, Fraction with carry in
MPS, DVS	Multiply-Step, Divide-Step
MEMORY OPERATIONS	
LDM, STM	Load from Memory, Store into Memory
LDI, STI	Load, Store and Increment Address
LDW, STW	Load from Working Storage, Store into Working Storage
LDS	Load from Memory, special addressing
SKIP AND JUMP	
TZO, TNZ	Test specified register field for zero, non-zero
TCT, TCF	Test specified condition for true, false
CLP, FTR, BTR	Copy Literal address to P, Forward, Backward Transfer Relative

tions, the group of logical operations may be applied to any pair of registers and may be conducted in conjunction with a data transfer. The operations of the shifting group provide a very general shifting capability which can be applied to any register. The shift commands include micro-orders for the purpose of normalizing and equalizing numbers during floating point processing.

The arithmetic operations are generally self-explanatory. As was the case with the register manipulations above, the significance of particular fields of data in the sign-magnitude and floating point formats has been recognized and accommodated. The need to accomplish the important operations of multiplication and division rapidly led to the specialized Multiply Step and Divide Step micro-orders.

The memory operations further reflect the microprogrammed nature of the computer in that any of the addressable registers may be used to supply address information and to receive or supply data, as required by the operation. Also, eight working storage locations may be addressed directly by micro-order without reference to an address register. The Load Special micro-order (LDS) permits an entry in one of several 64-word tables to be obtained from memory on the basis of a 6-bit partial address in the D register. This special purpose order facilitates the interpretation of pseudo instruction operation codes as micro-routine starting addresses. It also permits the rapid translation of one alphanumeric code to another during input/output operations.

In the category Skip and Jump commands, several micro-orders have been collected under a group name which reflects their common feature, that of directing program control to a new sequence of micro-orders. It will be seen that the ability to specify the P register in the R2 field of a register "copy" operation provides the most general means for achieving a jump in the program sequence. The relative transfer operations (FTR, BTR) of this grou) accomplish a similar result by modifying the program counter (P register) contents rather than substituting a completely new address. The test micro-orders (TZO, TNZ, TCT, TCF) permit the programmer to test a variety of logical conditions and skip a following micro-order on the basis of either condition met or condition not met.

A final group of micro-orders, not included in Table 2, permits programmed control over input/output equipment. They provide a means for issuing commands to selected external devices, controlling the operation of an I/O channel commutator, and conducting data transfer operations with previously activated devices. The variety of data transfer modes which may be programmed using these micro-orders include:

(1) single character input/output
  (a) wait for device ready
  (b) test for device ready
  (c) interrupt program when device ready
(2) data block transfer
  (a) uninterrupted transfer at device data rate
  (b) buffered transfer to or from a pre-assigned area of storage on an "interrupt program when device ready" basis.

PROCESSING UNIT

The processing unit consists of four 24-bit registers, two 15-bit registers and a single 8-bit register. For purposes of register transfers and other manipulations, the registers are aligned logically as indicated in Fig. 2. In addition to their general programmed use, two registers have fixed roles in internal operations:

(1) The P-register serves as the microprogram counter, being incremented automatically and submitting its contents as a memory address during each micro-order-pair access cycle.
(2) The N-register serves as the repeat counter for timing shift operations, and multiply-step and divide-step micro-orders.

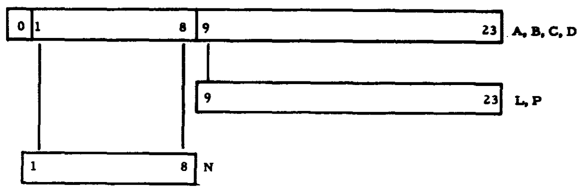


Figure 2. Processing Unit Registers.

It will be noted in Fig. 2 that the N-register is located in the bit positions corresponding to the exponent field of a floating point format word; and that the L-register, being of maximum address length can be used to good effect as a macro-instruction location counter.

The Processing Unit Block Diagram, Fig. 3, illustrates the manner in which the addressable registers of the computer may be connected to a binary full adder and other manipulative logic by use of the processing unit bus structure. Under the control of decoded micro-orders, the registers may be selectively gated onto the logic input busses (1 and 2), the desired manipulative logic enabled, and the result gated from the appropriate logic output bus (3 or 4) into the desired register. Thus, the addition of the contents of registers A and B may be accomplished by logically connecting A to bus 1, B to bus 2, enabling the full add logic and connecting bus 3 (carrying the binary sum) to the input logic of register B. Similarly, the 1's complement of the contents of register L may be obtained by connecting the outputs of L to bus 2, selecting the inversion logic input to bus 4, and connecting bus 4 to the register L input logic. Data transfers and register exchanges are facilitated by a direct connection between busses 2 and 4 and by the ability to disable the bus 2 input to the full add logic.*

* The PB 440 bus structure is reminiscent of the busses of Kampe's arithmetic unit (reference 5) but gains significant advantages from placing manipulative logic between register output and input busses.

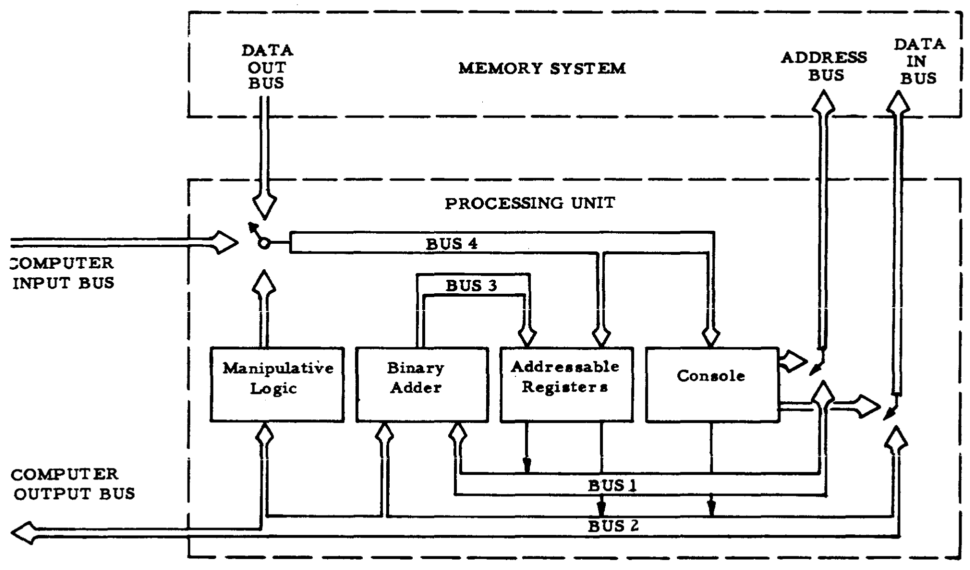


Figure 3. PB 440 Information Flow Diagram.

It should be noted on Fig. 3 that buses 1, 2, and 4 and their associated gating logic also provide the means for communication with the memory system, the input/output devices, and the operator's console. During the execution of a micro-order requiring memory operation, for example, the register designated for supplying the address will be gated onto bus 1 and bus 1 will be selected as the input to the memory address bus. For a memory "read" operation, the contents of the memory data output bus will be gated onto bus 4 and thence to the selected destination register. For a memory "write" operation, the data source register will be selected as an input to bus 2 which in turn will supply the memory data input bus.

During an input operation, bus 4 may be connected to the computer input bus which carries both data and status information from external devices. Similarly, data and control information may be gated from the register designated by an output micro-order to an external device via bus 2 and the computer output bus.

Access to the memory system and the addressable registers of the processing unit, from the operator's console, may be accomplished efficiently by making use of existing bus structure logic. Bus 4 may be monitored as a common point for displaying both memory data and register contents. Manual memory interrogations may be conducted from the console by supplying address data as an input to the memory address bus and by selecting the memory data out bus as an input to bus 4. For storage operations, data from a "register" of switches at the console may be gated onto bus 2 and thence to the memory data input bus. The contents of processing unit registers may be displayed by use of bus 2 and the direct input to bus 4. The same logical path may be used to alter the contents of a specified register by selecting the console data switches as an input to bus 2, and bus 4 as an input to the designated register.

The logical details of the processing unit bus structure may now be considered. Illustrated from left to right in Fig. 4 are:

(1) the X-register and its associated logic for decoding the micro-order type and the register codes (R1, R2) of the micro-order which is to be executed at the next logic clock;

(2) the general bit position, i, of each of the processing unit registers which may be gated onto bus 1 and/or bus 2 as dictated by control signals derived from the micro-order type and register codes;

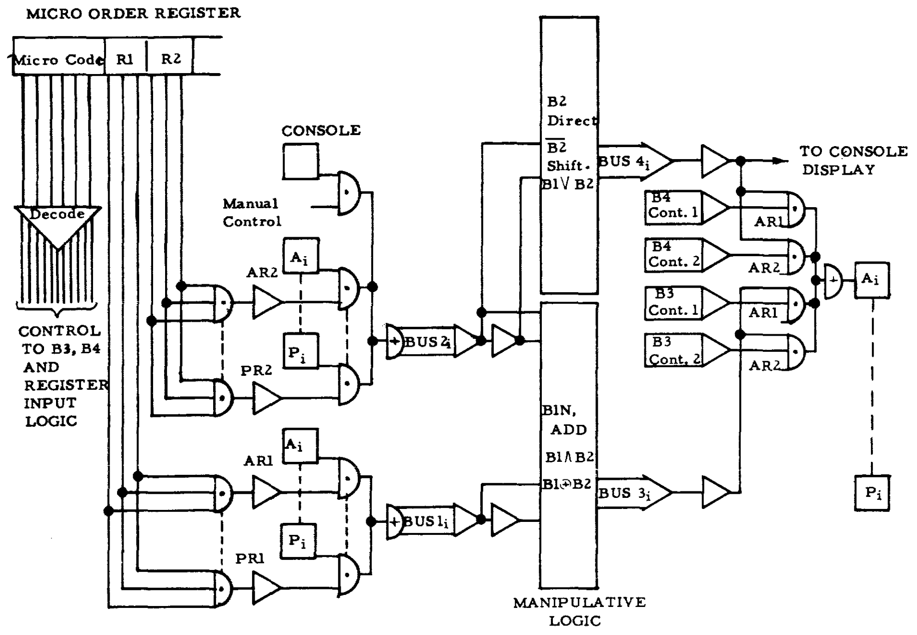


Figure 4. Processing Unit Bus Logic.

(3) the principal types of arithmetic and manipulative logic which may be applied selectively to the register output busses; and

(4) the register input busses (3 and 4), which may be selectively gated into the general bit position, i, of each of the processing unit registers; again, under the control of the current micro-order type and register code.

Not indicated on Fig. 4, but important to an understanding of computer operation, is the logic for several other processing unit functions. These include:

(1) a variation on the binary addition logic which permits data on bus 1 to be incremented or decremented;

(2) logic to increment the contents of the P-register (program counter);

(3) logic to decrement the contents of the N-register (repeat counter);

(4) double length shifting logic on registers A and B; and

(5) the logic for applying a variety of tests to the data being transmitted over the general bus structure.

Apparent on Fig. 4 are several features which are significant in the mechanization of the processing unit logic described above. It will be noted, for example, that the flip-flops of the 7 programmable registers need to supply only normal (or true) outputs to the register selection logic at the inputs to busses 1 and 2. The complement form of the selected register's contents is made available to the manipulative logic by inverting amplifiers on the busses. The bus 2 amplifiers supply not only the logic associated with the full adder, but also the variety of functions which are mechanized at the input to bus 4.

An equally important feature is the mechanization of the flip-flop input logic which is permitted by the use of single input, DELAY flip-flops. The logical properties of the "D" flip-flop require a "one" input at the beginning of each clock interval during which a "one" output is desired, and cause a return to the zero state if no input is present. This eliminates the need for constructing the "reset" portion of conventional set/reset logic, and further eliminates the requirement that register input busses 3 and 4 be implemented to carry complement information.

Finally, as an indication of required logic circuit performance, Fig. 4 identifies the delay producing elements in both data and control paths. The accumulated effect of these individual time delays is to establish the upper limit of the logic clock frequency. More specifically, a 12-bit micro-order will be inserted into the left half of the X-register (left side of Fig. 4) at one logic clock time, and the voltage levels produced by the register input logic (right side of Fig. 4) will establish the new states of the register flip-flops at the next logic clock time. In a typical operation (ADD, e.g.), the delays encountered are, in order of occurrence:

(1) decoding and driver delay for micro-order type and register selection controls;
(2) logic circuit delay at bus 1 and 2 input gating;
(3) bus 1 and 2 driver delays;
(4) carry propagation time through full add logic;
(5) bus 3 driver delay; and
(6) logic circuit delay at register input gating.

The delay times experienced with PB 440 production circuitry show an average delay of 25–40 nanoseconds through AND-OR-AND-OR diode logic and a bus amplifier, and 300 nanoseconds maximum carry propagation time in the full adder. A logic clock frequency of 1 megacycle was found to provide for worst case decoding and signal paths with adequate margin and still allow a required 200 nanoseconds settling time for flip-flop input circuitry.

As a comment on the basic nature of the micro-order repertoire, all non-repetitive micro-orders require a single clock time for their execution. That is, a binary addition of two 24-bit registers is accomplished in one microsecond with the result replacing one of the operands.

MEMORY SYSTEM

The use of memory input, output and address busses in processing unit communications with the memory system was introduced in Fig. 3 above. The technique used in connecting individual memory modules to these busses is illustrated in Fig. 5, and will be described further following some general remarks concerning this method of memory communication.

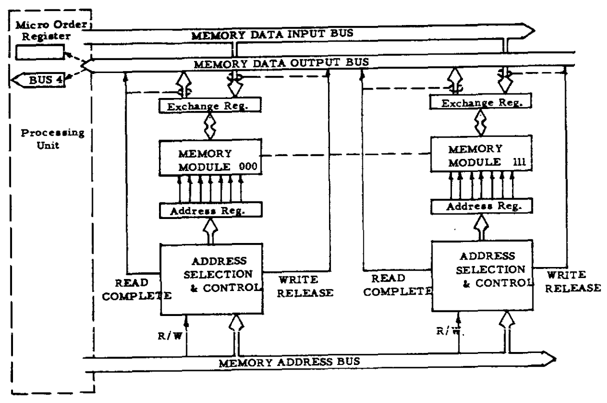


Figure 5. Memory Systєm.

One of the most significant features of the bus type memory communication is that it facilitates the modular expansion of the memory system. As indicated in Fig. 5, up to eight modules of 4096 words each may be accommodated without altering the logic of the basic computer design. Memory control logic, which treats each module as a separate asynchronous device, is a second important feature which has the following advantages:

(1) Memory modules of different access and cycle times may be accommodated in the system without complicating the control logic or introducing problems into the process of program storage allocation.
(2) The ability to initiate a memory read operation in one module while a rewrite (restore) operation is being conducted in a previously addressed module creates the opportunity to store program instructions or data strings alternately in the two modules and, as a consequence, to realize a significant reduction in program execution time.
(3) As memory modules of improved performance are developed they can be used to replace the slower original equipment, thus prolonging the competitive life of the computer.

Item (1) is particularly important to the dual-speed memory concept employed in the

PB 440. It permits the combined use of a fast, nondestructively read memory unit containing microprogrammed stored logic, and a larger capacity core memory containing macro-instructions, operands and constants. In addition to allowing overlapped operation of core modules, item (2) permits several accesses to the fast memory and the continued execution of the microprogram during the rewrite portion of each main memory cycle.

The connection of the Exchange Register and Address Register of the individual modules to the memory busses is enabled by address selection and control logic, as shown in Fig. 5. The memory address bus and Read/Write control lines appear there as inputs to each memory module. The three most significant bits of the 15-bit address designate one of eight possible modules and the remaining 12 bits specify a particular one of its 4096 words.

One code is assigned to Fast memory and the remaining codes are used in designating up to 7 magnetic core memory modules. Module selection requires that in addition to the presence of a Read or Write request and the appropriate module code, the indicated module must be in a "not busy" condition. In requesting a memory operation, the computer control logic maintains the address information on the bus until an acknowledgement is received which indicates acceptance. In the event that the designated module is currently conducting a write cycle, the busy indication prevents the initiation of the requested operation until the cycle is completed.

Referring to Fig. 5, the output of the module selection logic gates the address data into the module address register and, in the case of a write operation, also connects the memory data input bus to the module exchange register. At this point in the write cycle timing a "Write Release" signal is sent to the computer control logic which permits continued (or renewed) program execution. At the conclusion of a read cycle a "Read Complete" signal is generated which connects the module exchange register to the memory data output bus and indicates to the processing unit that data is ready.

In order to insure that the data is not sampled while in transition between logic levels, some form of synchronization with the logic clock is required. This interlock is achieved by a "Data Ready" flip-flop which responds to the completion signal sufficiently in advance of the next logic clock to guarantee that the accompanying memory data will be settled at clock time. The Data Ready flip-flop also provides an echo to the selected module and a release signal to the computer control logic. The computer is thus not permitted to initiate a memory read operation before receiving the data ready signal terminating a previous read operation. At the left of Fig. 5, two possible paths are shown for memory output data. The direct path to the micro-order decoding register is enabled at the time in each computer cycle when the next micro-order pair is presented to the control circuitry. The second path, which uses bus 4, is typically employed to gate the results of an operand access into the register specified by the micro-order currently being executed.

The PB 440 standard memory module (4096 words of 25 bits) is designed to be connected in a tandem arrangement to the memory busses of a single computer. Certain classes of applications however require that one or more modules which comprise the memory system of the computer be shared with other devices. These applications can be accommodated by employing a Memory Interchange Unit to sample Read/Write requests from the several devices and to connect the device's address and data busses to those of the memory module(s) being shared for the duration of a single operation. From 1–8 memory modules may thus be shared by a maximum of 4 devices as shown in Fig. 8. In order to permit two or more PB 440's to share a common portion of their memory as described above, it is only necessary to connect the "Remote Memory" output jacks of each of the individual computers to a set of the Memory Interchange input jacks.

INPUT/OUTPUT SYSTEM

The input/output system and its relationship to the units already described is illustrated in the Input/Output System Block Diagram of Fig. 6.

Communication with the peripheral equipments is achieved by controlling their connec-

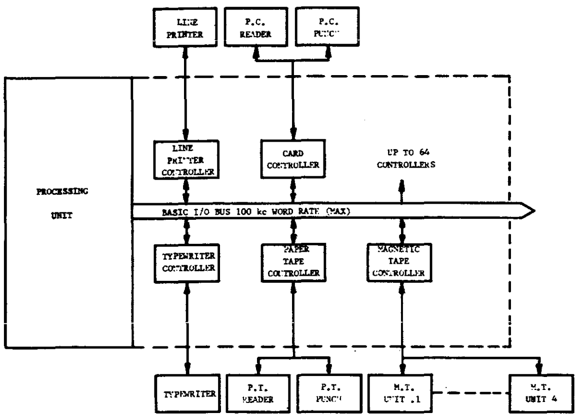


Figure 6. PB 440 Input/Output System Diagram.

tion to an input/output bus. The types of devices which may be so connected to the computer appear in Fig. 6 as individual blocks, distributed along the length of the input/output bus. The electronic connection of these equipments to the bus is accomplished by the use of a device controller which is unique for each equipment type and which permits the connection of up to 4 devices. A controller contains sufficient logic:

(1) to sample input/output bus control signals and determine when one of its devices is being "addressed" by the computer;
(2) to connect the device to the bus for the purpose of receiving commands or transfering data; and
(3) to maintain the device in a previously commanded mode of operation until either further commands are received or a predetermined condition occurs.

A controller also provides character or word buffering, appropriate to the device, and accomplishes any necessary error detection. The PB 440 input/output system is expandable to 64 controllers, each with 4 devices.

The basic control techniques employed in the logical mechanization of the input/output bus permit a variety of communication modes to be programmed. These include the ability to conduct up to four independent data block transfer operations, with assignable priorities, on a program interrupt basis. External devices with data rates on the order of 80,000 characters/

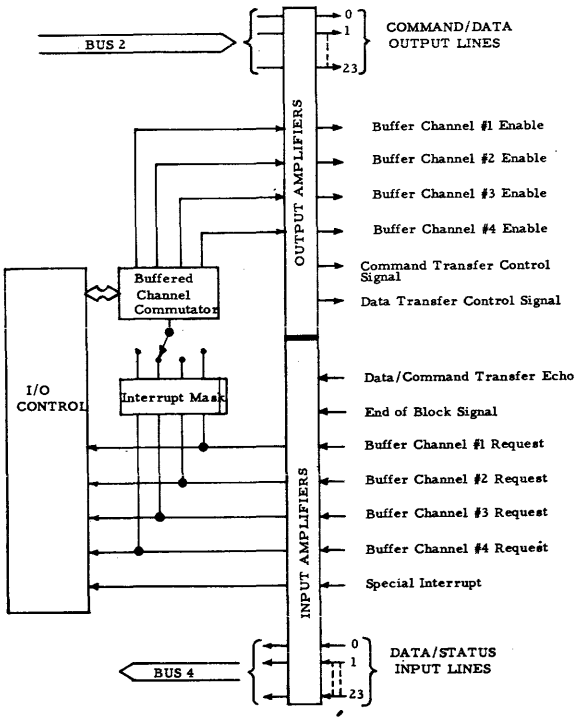

BUS 2
0
1
23
COMMAND/DATA OUTPUT LINES
OUTPUT AMPLIFIERS
Buffer Channel #1 Enable
Buffer Channel #2 Enable
Buffer Channel #3 Enable
Buffer Channel #4 Enable
Command Transfer Control Signal
Data Transfer Control Signal
Buffered Channel Commutator
I/O CONTROL
Interrupt Mask
Data/Command Transfer Echo
End of Block Signal
Buffer Channel #1 Request
Buffer Channel #2 Request
Buffer Channel #3 Request
Buffer Channel #4 Request
Special Interrupt
INPUT AMPLIFIERS
BUS 4
0
1
23
DATA/STATUS INPUT LINES

Figure 7. Input/Output Bus Details.

second can be accommodated by this means. The uninterrupted transfer of data to or from one program-selected device can be carried on at character rates up to 400 KC.

The details of implementing external device communications are presented in Fig. 7 which identifies the data and control lines that compose the input/output bus. Also, shown schematically, are the data connections to busses 2 and 4 and the logic required by the program interrupt feature. On the input side of the bus, the 24 data or status lines are gated on to bus 4, under the control of an input type micro-order, for distribution to the program selected register. The transfer of a command or data word is similarly conducted on the output side of the bus by gating the information from bus 2 under the control of an output type micro-order.

The four control lines, shown in the central portion of Fig. 7, serve to coordinate the transfer of information on the bus. The presence of a Command Transfer Control Signal identifies the information currently on the output lines as a command word. In addition to specifying a controller and a device number, a command word contains a buffered channel assignment to be used by the selected device in requesting subsequent data transfers. In responding to such a command, a selected device energizes the Data/Command Transfer Echo line and places 24 bits of status information on the input lines.

If, at the time a device is activated, the programmer elects to be interrupted when the device is ready for a transfer of data, processing continues until a signal is recognized on one of the buffered channel request lines. The buffered channel commutator halts automatically at this line, the program is interrupted, and control passes to a stored logic sequence which prepares to receive or transmit data into or from a preassigned memory location. Under stored logic control the locked position of the commutator is transmitted as a buffered channel enable signal to the interrupting device. A Data Transfer Control Signal simultaneously clocks data out of the computer for an output transfer, or in the case of an input transfer, indicates the computer's state of readiness. The transfer is terminated by the computer upon the recognition of a Data Transfer Echo, the commutator is unlocked (or reset) and control is returned to the interrupted program.

In contrast to the somewhat automatic features of the above described data transfer, at least two other modes of input/output communication are possible. The programmer may elect to devote the computer on a full time, uninterrupted basis to a data block transfer with a selected device. In this mode, the commutator is locked on the desired buffered channel under program control and a data transfer control signal is issued. The computer is thereby halted until the completion of the data transfer is indicated by the receipt of a data transfer echo. This process is repeated until the specified number of words has been transferred.

A third mode makes use of interrupt masking flip-flops, which may be set and tested under program control. They provide a means of masking buffered channel request signals at the input to the commutator but at the same time permit the request lines to be tested by the program for possible action. Thus, the readiness of the external device may be tested before executing a data transfer command and setting the input/output interlock.

A final input/output feature, denoted "special interrupt" in Fig. 7, permits the computer to respond to external events that were anticipated but not initiated by the program. This type of interrupt may correspond to the pressing of a typewriter key by the operator or the occurrence of some singular real time event. It may be tested periodically as a basis for transferring control to a program which will determine the cause of the alarm by interrogating each of the devices known to communicate over the special interrupt line. A data transfer can then be scheduled by assigning a buffered channel to the interrupting device in the manner described above.

These basic input/output capabilities can be extended by the use of an I/O adapter unit which permits direct buffering of data in and out of memory at character rates up to 800 KC. In Fig. 8, the address and data registers of such a direct channel are shown connected to a memory module which is shared with the computer. During system assembly this adapter is connected between the computer I/O bus and the controller for the selected device (a separate adapter is required for each device). The adapter transmits I/O commands from the computer to the controller and returns status response to the computer as required by the basic I/O communications procedures.

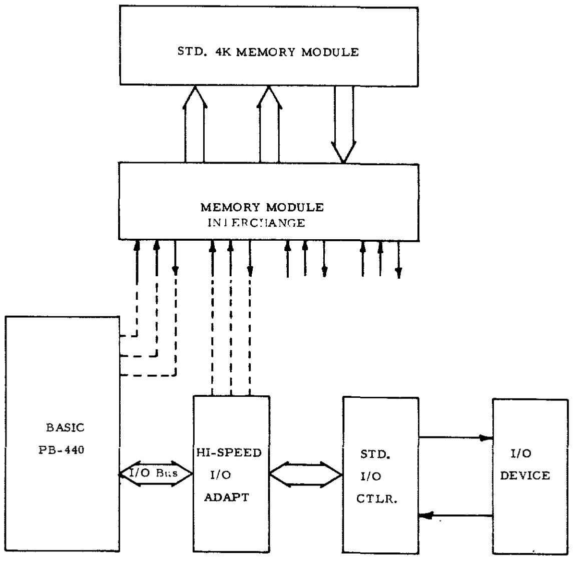


Figure 8. PB 440 Shared Memory and Hi-Speed I/O Options.

In addition, it contains a word counter and a current storage address counter which are filled by the programmer at the outset of a data block transfer between the device and the designated area of shared memory. The adapter thereafter functions to translate the controller's requests for inputs or outputs into Memory Read or Write signals, respectively, and conducts the corresponding data transmissions directly from or into memory. The computer may thus continue processing without interruption until such time as the desired number of words shall have been transferred and will then be directed to a micro-order sequence which will terminate the transmission.

The data transfer rates which can be accommodated if one memory module is used with the above described Memory Interchange and I/O Adapter options are determined by the memory module cycle time (5 microsconds) and the degree of competition among the devices connected to a Memory Interchange. The resulting data rate capabilities are tabulated below for the conditions of 1–4 active channels. In addition to a 24-bit word rate, a character rate is quoted which is applicable to those character oriented devices whose I/O controller is equipped with an assembly/disassembly register for 6-bit characters. Also, the 24-bit word rate is quoted as a bit rate to facilitate comparison with other systems.

SHARED MEMORY DATA RATES

No. Simultaneously Active Channels	24-Bit Word Rate	6-Bit Char. Rate	Bit Rate
1	200 KC	800 KC	4.8 MC
*2	100 KC	400 KC	2.4 MC
3	67 KC	268 KC	1.6 MC
4	50 KC	200 KC	1.2 MC
*If two memory modules are interlaced the following speeds apply:			
2	400 KC	1600 KC	9.6 MC

PROGRAMMING FEATURES

The microprogrammed character of the PB 440 has been developed above and its effect on the logical organization of the computer was discussed in some detail. A consideration of the programming features implied by this design approach will provide additional examples of microprogramming philosophies.

It has been emphasized that the PB 440 does not have a set of instructions in the same sense that conventional computers have. Instead, its basic design provides the programmer with the option of describing, in terms of elementary micro-orders, a sequence of steps which define an instruction. For clarity, we might refer to an instruction described in this way as a "macro-instruction". A complete instruction set consists of a "control" sequence, to determine which of the various macro-instructions is indicated, and a set of "function" sequences, each of which defines a macro-instruction. This instruction set is normally stored in fast memory although micro-orders can be executed from main memory if so desired.†

The design of an instruction set thus consists of three tasks. First, a macro-instruction format is chosen. This format need not be restricted to a word length of 24 bits, nor to single address instructions; it may, in fact, be a format used by another computer. Indexing, indirect and relative addressing, and other special functions may be included.

A control or interpretive sequence of micro-orders is next devised which fetches and identi-

† Grasseli's use of a Main Memory and a Control Memory in his "conventional microprogrammed control unit" (Ref. 6) resembles the PB 440 control concept if his fixed logic decoding of a macro-instruction is instead accomplished by a microprogrammed "interpretive sequence" which itself determines the starting address of a microprogrammed "function sequence."

fies successive macro-instruction, performs indicated address modifications and obtains operands as required.

Lastly, the function sequences, each defining an instruction, are written. Here the programmer-designer reduces an instruction to its basic logical ingredients and, thus, mechanizes each operation he wishes to include in the instruction set. The last micro-order of each function sequence returns control to the interpretive sequence.

It should be noted that the amount of time required to interpret a macro-instruction should be kept as short as possible since it must be added to the "execution" time of each macro-instruction. It may vary from less than 5 microseconds, for a relatively simple format, to more than 25 microseconds, where indirect addressing, indexing, and other mode indicators must all be satisfied.

To offset this interpretive "overhead", a distinct advantage appears in the mechanization of the relatively more complex functions which are usually offered as subroutines on conventional computers.‡ The ability to efficiently construct the desired function by listing an appropriate sequence of logical manipulations (micro-orders) should be contrasted with the preparation of a subroutine of conventional instructions whose inflexible execution usually includes time for steps not required in the task at hand. The need for register exchange and other housekeeping operations required by fixed logical structures is also obviated during microprogrammed manipulations of functionally unassigned registers. In the time required to perform ten conventional "short" instructions of 10 microseconds duration, a PB 440 sequence of 100 micro-orders can be executed. As a performance indication, a microprogrammed, Arctangent sequence requires approximately 160 microseconds.

A "library" of macro-operations, programmed in micro-order sequences, has been expressed in several useful macro-instruction formats. This permits the user, who is neither prepared to think in terms of micro-orders, nor inclined to place a compiler between himself and the computer, to use a more familiar instruction word format in preparing his programs.

‡ This conviction is shared with Glantz who presents a clear discussion on p. 78 of reference [2].

To the casual programmer who has occasion to use only those programming aids and instruction sets that have been devised and provided for him by others, the microprogrammable feature means a larger and more varied instruction repertoire and generally improved performance. To the more senior programmer, this feature represents both a benefit and a challenge. He benefits from the ability to modify and extend the instructions which have been microprogrammed previously, and he is challenged to optimize critical applications by creative programming at the basic micro-order level.

CONCLUSIONS

A particular approach to the design of a microprogrammable computer has been implemented and described above. Its advantages may be summarized by stating that:

(1) the ability to replace specialized logical circuitry by a sequence of stored micro-orders leads to efficiencies in the design and mechanization of the computer,

(2) the microprogrammed decoding or interpretation of pseudo-instructions allows variable instruction word formats but adds to execution times,

(3) the execution time of functions which are typically provided as subroutines are significantly improved by microprogramming implementation,

(4) the inherent flexbility of the approach permits the manufacturer to adapt the computer to individual customer requirements, and the customer to adapt the computer to a variety of tasks, and

(5) the micro-order control of basic logical operations facilitates the simulation of other computers making possible the use of existing software.

ACKNOWLEDGEMENTS

The authors wish to express their appreciation to Mr. Lowell Amdahl for his valuable advice and consulting assistance during the preliminary design phase of PB 440 development. Congratulations are in order to Mr. J. Roy Willson for his outstanding work on all aspects of computer circuit design and to Messrs. Bill Nickell and Scott Nelson in the perfection of the PB 440 memory units. Thanks are also expressed to Messrs. Stanley Groves and James Lusk for painstaking efforts expended during the course of detailed logical design and equipment checkout.

REFERENCES

1. WILKES, M. V., and STRINGER, J. B., "Micro-programming and the Design of the Control Circuits in an Electronic Digital Computer," *Proceedings, Cambridge Philosophical Society,* Cambridge, England, vol. 49, pt. 2, April 1953, pp. 230-238.
2. GLANTZ, H. T., "A Note on Microprogramming," *Journal of the Association for Computing Machinery,* New York, N.Y., vol. 3, April 1956, pp. 77-84.
3. WILKES, M. V., "Microprogramming," *Proceedings of the Eastern Joint Computer Conference,* December 1958, pp. 18-20.
4. MERCER, R. J., "Microprogramming," *Journal of the Association for Computing Machinery,* New York, N.Y., vol. 4, April 1957, pp. 157-171.
5. KAMPE, T. W., "The Design of a General-Purpose Microprogram-Controlled Computer with Elementary Structure," *IRE Transactions on Electronic Computers,* vol. EC-9, no. 2, June 1960, pp. 208-213.
6. GRASSELLI, A., "The Design of Program-Modifiable Micro-Programmed Control Units," *IRE Transactions on Electronic Computers,* vol. EC-11, no. 3, June 1962, pp. 336-339.

APPLICATION OF PUSHDOWN-STORE MACHINES

R. James Evey
International Business Machines Corporation
Cambridge, Massachusetts

1. INTRODUCTION

Scientific knowledge has always been advanced through the combined efforts of those who experiment and those who theorize. This is as true of the field in which the effective processing of languages by computers is considered as any other.

Some of the first successes in this field were compilers which translated from specially designed languages to computer code. Those first compilers were developed to make it easier to communicate with a given computer. Then, with the rapid increase in the number of different types of computers, it was thought that compilers could solve the problem of having to learn a new code with each new computer. But those hopes were not completely fulfilled because new progamming languages were developed almost as quickly as machines. With the proliferation of both languages and computers, the need arose for some unifying theory.

During that same period the first attempts were made to use a computer to translate from one natural language to another. Several word-for-word translators were devised, but the struggle to improve them significantly has been difficult. Experimentation along these lines has continued apace, but again the search was begun for helpful theories.

In the past few years, many theoretical steps have been taken[1-4 7, 15]. This paper is an attempt to take another. It provides a theoretical backgound for the discussion of certain problems in the generation, analysis, and translation of languages. Both artificial and natural languages will be considered. It is assumed that the reader is familiar with Turing machines[5], finite-state machines[16], and phrase-structure languages[3].

The paper consists of three further sections. A set of abstract machines will be described in Section 2. These machines can serve as models of different types of computers. Certain relations among the various machines will also be demonstrated in that section.

Applications of the theory are presented in the last two sections. Artificial languages are considered in Section 3 while natural languages are the subject of the last section. In Section 3, algorithms developed by Oettinger[15] and by Samelson and Bauer[17] are shown to be combinations of some of the machines described in Section 2. In Section 4, the relation of pushdown-store machines to the algorithms of Yngve[19] and Kuno and Oettinger[12] for the generation and analysis of natural languages will be demonstrated.

2. A SET OF ASBTRACT MACHINES

In this section an informal description of a set of abstract machines will be given by adding pushdown stores to finite-state machines. After the set of machines has been described, it will be partitioned into:

a. Accepters (machines with only inputs) and transducers (machines with both inputs and outputs); b. Deterministic and nondeterministic machines. A set of sequences of symbols will be called a language. A language accepted (generated) by a machine will be called its domain (range).

It will be shown that a language is the domain of a nondeterministic accepter of a given type if, and only if, it is the range of a deterministic transducer of the same type. In Section 4 this fact will be related to experimental work being done on natural languages.

2.1 Pushdown-Store Machines

Pushdown stores have been rigorously defined elsewhere in the literature[4], but a new notation will be given here. Five different infinite sets of metasymbols are needed, $\{\alpha_i\}$, $\{\beta_i\}$, $\{q_i\}$, $\{S_i\}$, $\{T_i\}$. Symbols from the sets S_i and T_i sometimes will be primed.

Consider an expression of the form $(\alpha_i)\ q_j \rightarrow q_k$. This will be interpreted to mean that a machine in state q_j, upon receiving an input symbol α_i, goes into state q_k. Similarly, $q_i \rightarrow q_j\ (\beta_k)$ will be taken to mean that a machine originally in state q_i outputs symbol β_k and goes into state q_j. With only expressions of these two forms any finite-state machine can be described.

Example 2.1. A Finite State Transducer, F_1.

$$
\begin{array}{rl}
(h) & q_1 \rightarrow q_2 \\
 & q_2 \rightarrow q_3 (h) \\
(h) & q_3 \rightarrow q_4 \\
(1) & q_3 \rightarrow q_5 \\
(0) & q_3 \rightarrow q_6 \\
 & q_4 \rightarrow q_1\ (h) \\
 & q_5 \rightarrow q_3\ (a) \\
 & q_6 \rightarrow q_3\ (b)
\end{array}
$$

This example illustrates some conventions that will govern all descriptions of machines to follow. Each machine will be considered as starting in state q_1, and a computation by the machine will not be considered complete unless the machine ends in state q_1. From this, it can be seen that each string of input or output symbols must begin and end with h. This fact also will be accepted as a convention which will be true for all machines that will be described. Now it is obvious that the simple machine F_1 does nothing more than output an 'a or b whenever it receives a 1 or 0, respectively.

To describe a pushdown-store machine, two more forms are needed: $q_i \rightarrow S_j\ q_k$ and $S_i\ q_j \rightarrow q_k$. These will mean, respectively: when the machine is in state q_i, it puts symbol S_j at the top of the pushdown and goes into state q_k; when the machine is in state q_j and finds symbol S_i at the top of the pushdown, it removes S_i and goes into state q_k.

Example 2.2. A Pushdown-Store Transducer, P_1.

The rules of P_1 are the same as those for F_1 down to and including q_4, so only the rules from q_5 on will be given.

$$
\begin{array}{rl}
 & q_5 \rightarrow A'q_7 \\
 & q_6 \rightarrow B'q_8 \\
 & q_7 \rightarrow q_9\ (a) \\
 & q_8 \rightarrow q_9\ (b) \\
(h) & q_9 \rightarrow q_{12} \\
(1) & q_9 \rightarrow q_{10} \\
(0) & q_9 \rightarrow q_{11} \\
 & q_{10} \rightarrow Aq_7 \\
 & q_{11} \rightarrow Bq_8 \\
 & A'q_{12} \rightarrow q_{13} \\
 & A\ q_{12} \rightarrow q_{15} \\
 & B'q_{12} \rightarrow q_{14} \\
 & B\ q_{12} \rightarrow q_{16} \\
 & q_{13} \rightarrow q_4\ (a) \\
 & q_{14} \rightarrow q_4\ (b) \\
 & q_{15} \rightarrow q_{12}\ (a) \\
 & q_{16} \rightarrow q_{12}\ (b)
\end{array}
$$

A second pushdown store can be added by using expressions of the forms $q_i \rightarrow q_j\ T_k$ and $q_i\ T_j \rightarrow q_k$. Machines with two pushdown stores can be seen to be equivalent to Turing machines.

Example 2.3. A Turing Machine, T_1.

The rules of T_1 are the same as those for P_1 down to and including q_{14}, so the rules from q_{15} on will be given.

$$
\begin{array}{rl}
 & q_{15} \rightarrow q_{17}A' \\
 & q_{16} \rightarrow q_{17}B' \\
A' & q_{17} \rightarrow q_{20} \\
A & q_{17} \rightarrow q_{18} \\
B' & q_{17} \rightarrow q_{21} \\
B & q_{17} \rightarrow q_{19} \\
 & q_{18} \rightarrow q_{17}A \\
 & q_{19} \rightarrow q_{17}B \\
 & q_{20} \rightarrow q_{22}(a) \\
 & q_{21} \rightarrow q_{22}(b)
\end{array}
$$

$$
\begin{aligned}
q_{22}A' &\rightarrow q_{13}\\
q_{22}B' &\rightarrow q_{14}\\
q_{22}B &\rightarrow q_{24}\\
q_{23} &\rightarrow q_{22}(a)\\
q_{24} &\rightarrow q_{22}(b)
\end{aligned}
$$

An example of a computation by P_1 will illustrate the computations of machines which are more powerful than finite-state machines. To represent computations of such machines four columns will be used, containing, respectively: (1) the input symbol; (2) the rule used; (3) the *instantaneous description* (i.e., the sequence of symbols in the pushdown stores together with the machine state); (4) the output symbol.

Example 2.4. A Computation of P_1.

Input	*Rule*	*Instantaneous Description*	*Output*
h		q_1	
	$(h)q_1 \rightarrow q_2$	q_2	
1	$q_2 \rightarrow q_3(h)$	q_3	h
	$(1)q_3 \rightarrow q_5$	q_5	
	$q_5 \rightarrow A'q_7$	$A'q_7$	
1	$q_7 \rightarrow q_9(a)$	$A'q_9$	a
	$(1)q_9 \rightarrow q_{10}$	$A'q_{10}$	
	$q_{10} \rightarrow Aq_7$	$A'Aq_7$	
0	$q_7 \rightarrow q_9(a)$	$A'Aq_9$	a
	$(0)q_9 \rightarrow q_{11}$	$A'Aq_{11}$	
	$q_{11} \rightarrow Bq_8$	$A'ABq_8$	
h	$q_8 \rightarrow q_9(b)$	$A'ABq_9$	b
	$(h)q_9 \rightarrow q_{12}$	$A'ABq_{12}$	
	$Bq_{12} \rightarrow q_{16}$	$A'Aq_{16}$	
	$q_{16} \rightarrow q_{12}(b)$	$A'Aq_{12}$	b
	$Aq_{12} \rightarrow q_{15}$	$A'q_{15}$	
	$q_{15} \rightarrow q_{12}(a)$	$A'q_{12}$	a
	$A'q_{12} \rightarrow q_{13}$	q_{13}	
	$q_{13} \rightarrow q_4(a)$	q_4	a
	$q_4 \rightarrow q_1(h)$	q_1	h

In this computation the use of primed symbols is made clear; they indicate the lowest symbol in a pushdown store.

So far, three machines have been described: a. finite-state machines, b. pushdown-store machines, and c. Turing machines. Three more machines can be described using only the notation so far presented, since a counter can be thought of as a pushdown store which can accept only one type of symbol (except, of course, for the primed end symbols). So counters can be added to machines to give three new machines: a. a finite-state machine with one counter, b. a finite-state machine with two counters, and c. a pushdown-store machine with one counter. It develops that the last two machines are exactly equivalent to Turing machines[14].

2.2. Nondeterministic Machines

The string of symbols, exclusive of h in the input (output) column of a particular computation will be called the *input* (*output*) of the machine for that computation. A set of such strings of symbols will be called a *language*. The set of inputs (outputs) of all computations of a particular machine will be called its *input language* (*output language*) or *domain* (*range*).

A machine with no input language seems rather nonsensical (though they could be given a meaning, as will be seen) so all machines will be assumed to possess a non-empty domain. But machines with no range are common and will be called *accepters*. All other machines (that is, those having both a range and a domain) will be called *transducers*. Any machine will be said to *accept* its domain and a transducer will be said to *generate* its range.

All machines now can be assigned to one of two classes, accepters or transducers. A second manner of partitioning the set of all machines will be considered now. In all the examples of machines given so far, two facts can be observed:

(1) For each i, if a rule of one of the forms

$q_i \rightarrow q_j\ (\beta_k)$, $q_i \rightarrow S_j\, q_k$, or $q_i \rightarrow q_j\, T_k$

occurs, then no other rule occurs with q_i to the left of the arrow.

(2) For each j, if q_j occurs to the left of the arrow in a rule of one of the forms

$(\alpha_i)\, q_j \rightarrow q_k$, $S_i\, q_j \rightarrow q_k$, or $q_j\, T_i \rightarrow q_k$

then:

q_j occurs to the left of the arrow only in rules of that same form, and no other rule of that same form occurs with subscripts i and j.

Machines whose descriptions obey these two conventions will be called *deterministic,* since the next instantaneous description in any computation of the machine can be completely determined. All other machines will be called *nondeterministic.*

One of the easiest ways to construct a nondeterministic machine is to construct, first, a deterministic machine and, then, reverse the rules according to the following conventions:

Rules of the forms

$q_i \rightarrow S_j q_k$, $S_i q_j \rightarrow q_k$, $q_i \rightarrow q_j T_k$, and $q_i T_j \rightarrow q_k$

shall be written as

$S_j q_k \rightarrow q_i$, $q_k \rightarrow S_i q_j$, $q_j T_k \rightarrow q_i$, and $q_k \rightarrow q_i T_j$,

respectively.

Rules of the forms

$(\alpha_i) q_j \rightarrow q_k$ and $q_i \rightarrow q_j (\beta_k)$

shall be written as

$q_k \rightarrow q_j (\alpha_i)$ and $(\beta_k) q_j \rightarrow q_i$, respectively.

A machine so described will be called the *inverse* machine of the original. However, inverse machines will not be restricted to deterministic machines; the inverse of any machine, deterministic or nondeterministic, can be constructed.

Example 2.5. The inverse machine P_1' of P_1.

$$
\begin{array}{rl}
(h)q_1 & \rightarrow q_4 \\
q_2 & \rightarrow q_1(h) \\
(h)q_3 & \rightarrow q_2 \\
q_4 & \rightarrow q_3(h) \\
(a)q_4 & \rightarrow q_{13} \\
(b)q_4 & \rightarrow q_{14} \\
q_5 & \rightarrow q_3(1) \\
q_6 & \rightarrow q_3(0) \\
A'q_7 & \rightarrow q_5 \\
A\,q_7 & \rightarrow q_{10} \\
B'q_8 & \rightarrow q_6 \\
B\,q_8 & \rightarrow q_{11} \\
(a)q_9 & \rightarrow q_7 \\
(b)q_9 & \rightarrow q_8 \\
q_{10} & \rightarrow q_9(1) \\
q_{11} & \rightarrow q_9(0) \\
q_{12} & \rightarrow q_9(h) \\
(a)q_{12} & \rightarrow q_{15} \\
(b)q_{12} & \rightarrow q_{16} \\
q_{13} & \rightarrow A'q_{12} \\
q_{14} & \rightarrow B'q_{12} \\
q_{15} & \rightarrow A\,q_{12} \\
q_{16} & \rightarrow B\,q_{12}
\end{array}
$$

Example 2.6. A computation of P_1'.

Input	*Rule*	*Instantaneous Description*	*Output*
h		q_1	
a	$(h)q_1 \rightarrow q_4$	q_4	
	$(a)q_4 \rightarrow q_{13}$	q_{13}	
a	$q_{13} \rightarrow A'q_{12}$	$A'q_{12}$	
	$(a)q_{12} \rightarrow q_{15}$	$A'q_{15}$	
b	$q_{15} \rightarrow Aq_{12}$	$A'Aq_{12}$	
	$(b)q_{12} \rightarrow q_{16}$	$A'Aq_{16}$	
	$q_{16} \rightarrow Bq_{12}$	$A'ABq_{12}$	
b	$q_{12} \rightarrow q_9(h)$	$A'ABq_9$	h
	$(b)q_9 \rightarrow q_8$	$A'ABq_8$	
	$(B)q_8 \rightarrow q_{11}$	$A'Aq_{11}$	
a	$q_{11} \rightarrow q_9(0)$	$A'Aq_9$	0
	$(a)q_9 \rightarrow q_7$	$A'Aq_7$	
	$Aq_7 \rightarrow q_{10}$	$A'q_{10}$	
	$q_{10} \rightarrow q_9(1)$	$A'q_9$	1
a	$(a)q_9 \rightarrow q_7$	$A'q_7$	
	$A'q_7 \rightarrow q_5$	q_5	
h	$q_5 \rightarrow q_3(1)$	q_3	1
	$(h)q_3 \rightarrow q_2$	q_2	
	$q_2 \rightarrow q_1(h)$	q_1	(h)

Note that this computation can be obtained as follows: Simply turn the computation of P_1 (Example 2.4) upside down so that the output column becomes the input column and vice versa; relabel the second and third columns so that they read "Instantaneous Description" and "Rule", respectively; and move each item in the third column (the new "Rule" column) down one row.

The two examples just given show that for any computation of a transducer there is a corresponding computation of its inverse transducer. So the following theorem has been established:

Theorem 1. Given any transducer M (deterministic or nondeterministic) there is another transducer M′ (probably nondeterministic) such that the domain and range of M are the range and domain, respectively, of M′.

Now consider a deterministic transducer M, and construct the inverse machine M′ (which will probably be nondeterministic). Remove the outputs from M′ (either by rewriting its rules or by having it output the empty symbol) to form a new machine, the accepter M″. Then the domain of M″ is the range of M.

Conversely, given an accepter N, a "mechanism" N′ consisting of the inverse of all the rules of N can be constructed. Since N′ will have no inputs (N has no range), it does not fit the description of the machine which has been agreed upon here. This inverse "mechanism" must be nondeterministic since, even with no inputs, it can generate the complete domain of N. It can then be made to fit our description of a machine and at the same time be made deterministic by constructing a new machine N″, which is like N′ except that for each state in which there is a choice of action possible for N′, machine N″ will read a symbol from its input string. There will be one symbol for each possible choice that N′ could make. In this way the range of N″ will be made to coincide with that of N′ but N″ will have a domain and be deterministic.

So we have established Theorem 2: A language is the domain of a nondeterministic accepter of a given type if, and only if, it is the range of a deterministic transducer of the same type.

Chomsky[4] has proved Theorem 3: A language is a context-free phrase-structure language if, and only if, it is the domain of a nondeterministic pushdown-store accepter.

Theorems 2 and 3 give Theorem 4:* A language is a context-free phrase-structure language if, and only if, it is the range of a deterministic pushdown-store transducer.

3. TRANSLATION OF ARTIFICIAL LANGUAGES

In this section, some useful algorithms which have been developed to translate from one artificial language to another will be examined. In particular the algorithms of Oettinger[15] will be related to the one given by Samelson and Bauer.[17] It will be shown that all these algorithms follow rather directly from some of the theoretical considerations of Section 2. The papers of Oettinger and Samelson-Bauer will be assumed as known and the languages called L, P_2, and P_3, by Oettinger, will be referred to without further comment.

3.1. Translations for Parenthesis-free Languages

Theorem 4 tell us that for every context-free language there is a deterministic pushdown-store transducer that generates it. There is a very straightforward way of constructing such a transducer given such a language. The method will be shown by example.

Example 3.1. A context-free phrase-structure grammar for P_3.

Axiom: A

Productions:

—.	$A \rightarrow (— A)$
+.	$A \rightarrow (A + A)$
*.	$A \rightarrow (A * A)$
x_i.	$A \rightarrow x_i$

Construct a pushdown-store transducer that can read the following symbols from its input string, h, —, +, *, and x_i. For each symbol read, it performs the following actions:

1. If h is read at the start of a computation, h is put into the empty pushdown store. If h is read during a computation, the machine checks the pushdown store. If anything other than h is discovered there, the machine gives an error indication and stops. If h is discovered, it is removed, emptying the pushdown store, and the machine halts.

2. If —, +, *, or x_i is read, the machine puts the string to the right of the arrow in the corresponding production of the grammar of P_3 into the pushdown store, starting from the left of the string and proceeding to the right. Then the machine removes the topmost character of the pushdown store. If the character is not A, it is written in the output string and the machine removes the next character from the top of the pushdown. This process is repeated until the machine removes A from the top of the pushdown. This symbol is not written in the output string; instead, the machine reads the next input symbol.

A study of Oettinger's algorithm for translating from L to P_3 shows that it is simply

* All theorems stated here—whose proofs have merely been illustrated—are proved rigorously in the author's doctoral dissertation[7]. Furthermore, Theorem 4 is proved directly there without benefit of Theorem 3. That proof has the advantage that not only does it give an independent proof of Theorem 3, but also two theorems (one by Ginsburg and Rose[9] and the other by Bar-Hillel et al.[1], used in Chomsky's proof, fall out as corollaries.

another way of stating what we have described. But the method just given is applicable to any context-free grammar. Furthermore, there are a great many context-free languages which can be generated with a parenthesis-free language as the input language[7].

Translating in the reverse direction is not so simple, however. Theorem 1 shows that it can always be done if one is satisfied with a non-deterministic transducer, but it is not true in general that the inverse machine can be made deterministic. However, it can be done for the inverse machine of the one we have just described, and when it is constructed, it is seen to be precisely Oettinger's algorithm for translating from P_3 to L[7].

3.2. The Samelson and Bauer Algorithm

Consider two context-free languages which can be generated using pushdown-store transducers which have the same input language (e.g., P_2 and P_3 can both be generated using L as the input language). If one of these languages can be translated in the reverse direction using only a deterministic transducer, then that language can be translated to the other language using two applications of deterministic pushdown-store transducers. In essence, this is what Samelson and Bauer do in their algorithm, except that the final language they generate is not a context-free language, so a more powerful machine must be used for the second step. The first step is precisely as described here, however, except that they translate from a more general language than P_3 (we will call it P_4). This language is simply the familiar algebraic language in which most of the parentheses are omitted. We will now consider the second step of their algorithm.

Consider a very simple stored-program, single-address, random-access computer C. Only two arithmetic (floating-point) operations, addition and multiplication, are possible on this machine. All operations are carried out using one register called an accumulator. A symbolic assembler AP has been written for C which accepts programs written using:

1. CLA (Clear and add to the accumulator.)
2. ADD (Add the contents of the accumulator to . . .)
3. MPY (Multiply the contents of the accumulator by . . .)
4. STO (Store the contents of the accumulator in . . .)
5. Addresses which may be either a string of (capital) letters or a string of letters followed by one positive integer.

For example, here is a program for C which evaluates the formula $((x^* y) * (y + z))$:

```
CLA    X
MPY    Y
STO    A1
CLA    Y
ADD    Z
MPY    A1
STO    A.
```

Let the language L (C) be the set of all programs of C which evaluate single formulas of P_3, which contain no occurrences of — . Then what would constitute a non-well-formed formula of L (C)? No attempt will be made here to be completely precise, but some criteria can be given:

1. Each well-formed program must begin with CLA and end with STO. CLA is essential at the beginning of a program to make sure that the accumulator is first cleared and then the proper number is entered. STO at the end is arbitrary since the final answer could be left in the accumulator.
2. No CLA may occur inside a program, except following STO. The next instruction after STO need not be CLA, but CLA cannot occur in a well-formed program except as the first instruction or immediately following STO. This criterion rules out such sequences as:

   ```
   CLA   X
   ADD   Y
   CLA   Z
   ```

   which are clearly useless. However, STO, not followed immediately by CLA, serves the purpose of retaining useful partial results if Criterion 4 also is adhered to.
3. STO can only follow ADD or MPY. Since the computer is random access and only single formulas are being evaluated, no

useful purpose is served by sequences such as:

```
        CLA    X
        STO    Y
or
        STO    A
        STO    B.
```

4. If α is the address of STO, not the last instruction in a program, then α must also be the address of some instruction other than STO. Furthermore, an instruction with α as the address, which is not STO, must occur following the given STO and occur before any other STO which has α as an address. In other words, it is useless to store partial results and then never use them again.

How can programs of C which satisfy these four criteria be generated? The following simple procedure is proposed. Let α, β, γ, etc., be variables ranging over the acceptable addresses of the assembler AP. A STO list will be kept, with all addresses used in STO instructions. At the beginning of the generation of a program, the STO list is, of course, empty. As each instruction which is not STO is added to the program, its address α is compared with the elements of the STO list. If α is not in the STO list, the instruction with α as address is simply added to the program. If α is in the STO list, it is removed from that list before the instruction is added to the program. When a STO is proposed for addition to the program, the STO list is checked again. If its address α is not in the STO list, the instruction STO α may be added to the program. If α is in the STO list, another address must be proposed for the STO instruction. This address also must be checked against the STO list. This procedure is repeated until an acceptable address is found for the STO instruction.

Use of the STO list will cause Criterion 4 to be satisfied. The other criteria are simple to satisfy. Only the last instruction which has been added need be remembered. Unless it is ADD or MPY, the next instruction cannot be STO; unless it is STO, the next instruction cannot be CLA. If these rules are followed, the first instruction is CLA and the process ends by providing STO whose address is the only element in the STO list, and the program will be accepted as well-formed.

If these four criteria are accepted as specifying L (C), it does not appear possible to design any simpler mechanism to generate L. But any transducer which is used to carry out this process must be equipped with two pushdown stores to handle the STO list. So, even this simple mechanism is more powerful than a pushdown-store transducer; therefore, by Theorem 4, L (C) cannot be a phrase-structure language.

Most computers have basic codes ("machine language") which are similar to the one outlined here, but more complicated. If the well-formed programs of even this simple computer do not constitute a phrase-structure language, then the well-formed programs of most actual computers probably do not. This could be a reason why humans find machine languages so distasteful.

If the attempt to generate all of L (C) is abandoned, a pushdown-store transducer with one counter can be constructed which generates a subset of L (C) which still includes well-formed programs for evaluating all the formulas of P_3 not containing any occurrences of —. Furthermore, this machine will have a particularly useful input language.

Let L′ be the subset of the language L which does not include any formulas containing occurrences of —. Let M_1 be a pushdown-store transducer with one counter which accepts L′ as input language. The output language of M_1 will be a subset of L (C) in which occur only addresses of the forms Xn and Am, where n and m are positive integers. The pushdown-store vocabulary of M_1 will consist of Xn, which will be called program variables; Am, which will be called STO addresses; and a symbol * to designate the accumulator. As M_1 reads an input formula of L′, symbol by symbol, it performs one of the following actions depending on the symbol read and the state of its pushdown store:

1. If the input symbol is a variable Xn, M_1 checks the top two characters in its pushdown store.

   Case 1. The top character is a program variable Xm but the next character down is *. Then M_1 writes STO Ap, where p is the current content of the counter, on the output tape; adds one to the counter; replaces * by Ap, puts Xm back on top of Ap; adds Xn on top of Xm; and reads new input symbol.

Case 2. Any other configuration than that given by Case 1. Put Xn on top of pushdown store and read new input symbol.

2. If the input symbol is + or*, M_1 checks two top characters of the pushdown store.

Case 1. Both are program variables, Xn_1 on top and Xn_2 next below. Then M_1 writes on its output tape CLA Xn_2, ADD Xn_1 or MPY Xn_1, as the case may be; replaces Xn_1 and Xn_2 by *; and reads next input symbol.

Case 2. The top character is a program variable Xn and next character down is *. M_1 writes ADD Xn or MPY Xn, as the case may be; removes Xn leaving * as topmost character of pushdown store; and reads next input symbol.

Case 3. The top character is * and just below is Xn. M_1 writes STO Am, CLA Xn, ADD Am, or MPY Am, as the case may be, where m is the number in the counter; removes Xn leaving * as the top-most character of the pushdown store; and reads next input character.

Case 4. The top character is * and the next one below is An. The number in the counter at this point must be n + 1, so M_1 writes on its output tape STO CLA An, ADD or MPY AN+1, as the case may be; subtracts one from the counter; removes An leaving * as the top-most character in the pushdown store; and reads the next input symbol.

Case 5. No other cases should occur; if they do, it is a machine error and M_1 stops.

3. If the input symbol is h. If M_1 is in its starting state q_1, it writes h on the output tape, enters a state other than q_1, and reads the next input symbol. If M_1 is in a state other than q_1, it checks the pushdown store. If the pushdown store contains anything other than *, M_1 enters the error state q_s. Under no other condition does M_1 enter state q_s. If the pushdown store contains only *, M_1 empties the pushdown store, writes STO A, h on the output tape, and enters state q_1. Under no other conditions does M_1 enter state q_1.

Example 3.2. A generation of a program with x_1 x_2 x_2 x_3 + · as input formula.

Input Symbol	*Pushdown Store*	*Counter*	*Output*
h	Λ	Λ	h
x_1	Λ	Λ	Λ
x_2	X1	Λ	Λ
·	X1, X2	Λ	CLA X1, MPY X2
x_2	*	Λ	Λ
x_3	*, X2	Λ	STO A
+	A, X2, X3	1	CLA X2, ADD X3
·	A, *	1	STO A1, CLA A, MPY A1
h	*	Λ	STO A, h
—	Λ	Λ	Λ

It is essentially just such an extension which was used by Samelson and Bauer. They used a pushdown-store transducer to translate to L from an extended language related to P_4. But, instead of writing the formula of L on an output tape, the formula is used, symbol by symbol, as input to another machine which translates to a computer program. The resulting machine can be thought of as consisting of one input tape, one output tape, two pushdown stores, and a counter. This machine, though more powerful than a pushdown-store transducer, preserves the one-pass feature of such machines.

4. TRANSLATION OF NATURAL LANGUAGES

No one has yet devised a translator that translates from one natural language to another and which performs significantly better than a simple word-for-word translator. However, on the assumption that natural languages are almost phrase-structure languages, two investigations currently in progress under the leadership of Yngve[19] and Oettinger[12], respectively, may be very useful in constructing a good translator. Not until the past year was it established, however, that, in each of these investigations, it is implicitly assumed that natural languages are approximately phrase-structure languages. These facts will be established in detail later in this section.

First, Yngve's work will be reviewed and, using Theorem 4, the assumptions of phrase structure will be obvious. Secondly, an example of the work of Oettinger will be presented. This example is constructed in the hope that it will provide a simple abstract model for his work and thus make the theoretical implications of his methods easier to see.

4.1. Yngve's Language Generating Machine

Yngve starts by assuming a phrase-structure grammar for English. He then constructs a pushdown-store transducer to generate English sentences using the phrase-structure grammar.

If this were all that he suggested, there would be no problem, of course. The assumption of phrase structure would be explicit. However, he introduces two innovations. The first is called the depth hypothesis. This means that a finite pushdown store rather than an infinite one is used. Clearly, this changes the pushdown-store transducer into a finite-state transducer which is structured in a way resembling pushdown storage.

The new feature of Yngve's work which will be examined now is his method of handling discontinuous constituents. An example given by Yngve of a sentence containing a discontinuous constituent is:

It is true that he went.

The subject clause, "It . . . that he went", is called discontinuous. Yngve proposes that such constructions be handled by allowing a new type of rule in his grammar.

If the standard manner of presenting a nonterminal production of a normal phrase-structure grammar[4] is accepted as:

$$A \rightarrow BC,$$

Yngve proposes that rules of the form

$$A \rightarrow B \ldots\ldots C$$

be accepted also. The use of such a production follows. Suppose ϕDAz is a theorem of some phrase-structure grammar, where ϕ is a possibly empty formula over the complete vocabulary, z is a possibly empty formula composed of terminal symbols, and D and A are nonterminal symbols. The consequence of this theorem upon applying the new production would be $\phi BDCz$. The nonterminal symbol D (when it becomes the first available symbol) may be capable of further expansion. That is, it may be that there exists a formula ω such that there are more than two symbols in ω and $D \rightarrow \omega$. In this case, constituents B and C or their descendants would be separated by ω. Such rules allow for the generation of sentences containing very widely separated discontinuous constituents.

Yngve himself, as well as other commentators,[10] has wondered whether the addition of rules of this type cause this system to generate something which is not a phrase-structure language. However, a production of this type can readily be fitted into a pushdown-store grammar (i.e., a grammar whose language is the range of a pushdown-store transducer). So without the depth hypothesis, Yngve's model is simply a pushdown-store transducer and all the methods and results developed in this paper apply.

4.2. Multiple-path Syntactic Analyzer

Only recently, Kuno[12], under the leadership of Oettinger, has developed a new method of predictive syntactic analysis. This method makes it possible for all the different acceptable analyses of a syntactically ambigous sentence to be obtained. It has already been pointed out that predictive syntactic analysis is related to nondeterministic pushdown-store accepters.[4, 10] Theorem 3 then shows that this method is implicitly based on an assumption of phrase structure.

No attempt will be made here to establish these facts in a formal way. Instead the principles of the multiple-path syntactic analyzer will be applied to a simple grammar G_1. This will provide a simple example of this method.

The grammar G_1 is as follows:
Axiom: A
Productions:
$$A \rightarrow (A + A)$$
$$A \rightarrow (A + A$$
$$A \rightarrow A + A)$$
$$A \rightarrow x$$

One of the prominent features of predictive analysis is the use of a pushdown store to store predictions of the remaining structure of the formula being read. As each symbol is read the topmost prediction is compared with this symbol. This pair (prediction, symbol) is used as the argument of a grammatical-matching function G which is specified by a grammar table. To each pair G assigns a (possibly empty) set of new predictions.

In the first versions of predictive analysis, only the most probable of these predictions was kept[18]. It replaced the topmost prediction of the pushdown store. A new symbol was read and a new prediction was found. If $G(\langle P_i, S_j \rangle) = \Lambda$ for some prediction P_i and S_j, the process was terminated without obtaining an analysis of the sentence. This corresponds to reaching a blocked instananeous description of a nondeterministic pushdown-store accepter. Only if the right choices were made, would the end of the sentence be reached with all predictions in the pushdown store having been met. In this case one (and only one, of course) analysis of a sentence was obtained.

Multiple-path syntactic analysis improves on this scheme by trying all the predictions in each set at once. The process starts using only one pushdown store, but as soon as a value of $G(\langle P_i, S_j \rangle)$ for some P_i and S_j is obtained, which contains more than one member, more pushdown stores are necessary. So this process is an example of a machine using multiple pushdown stores and the number of pushdown stores necessary can increase without bound. The machine is thus actually a Turing machine which is structured into a number of pushdown stores.

The grammar table for G_1 will be constructed using the following conventions:

1. The grammar table will be presented as a square array with each prediction defining a row of the table and each symbol a column.
2. Let F be a prediction which means that a complete well-formed formula may follow.
3. Let S mean that the symbol read satisfied the prediction. For example, if the prediction is + and the symbol read is +, the prediction is satisfied. In this case, the prediction is simply removed from the pushdown store and the next prediction below becomes the new prediction.
4. Let # mean that the symbol read and the prediction at the top of the pushdown store are incompatible. In this case this particular analysis of the formula is discarded as unacceptable.

Grammar Table for G_1

$P_i \setminus S_j$	(	x	+	)
F	{ F+ F, F+ F +FR, F + FB, F + FB + FR}	{ + FR, S }	#	#
R	#	#	#	S
B	#	#	#	S
+	#	#	S	#

A Turing machine T_2 will now be constructed to use this grammar table to give multiple analyses of formulas of L (G_4). T_2 starts in state q_1 and reads h from the input tape. At this point, no pushdown stores have been established. But now T_2 establishes one pushdown store, puts F in it, enters a state other than q_1, and reads a new symbol. From this point, T_2 has a choice of actions depending on the symbol read and the predictions at the top of each pushdown store.

1. If the symbol read is (, T_2 checks each pushdown store. Those pushdown stores which do not have F at the top are emptied and dismantled. But for each of the pushdown stores which do have F at the top, T_2 establishes three new pushdown stores. Then T_2 copies the contents of each of the previously existing pushdown stores into three new pushdown stores. Then the F at the top of each of a set of four is replaced by a different member of G ($\langle$ F, $\rangle$), and T_2 reads a new input symbol. If a pushdown store is emptied by simply removing an F (for the prediction S of input symbol x), an h is placed in that pushdown store.
2. If the symbol just read is x, the action is similar to Case 1, except that the number of pushdown stores is only doubled since there are only two elements in G ($\langle$ F, x$\rangle$).
3. If the symbol read is +, all pushdown stores which do not have + at the top are dismantled. T_2 removes the + at the top of each of the remaining pushdown stores and reads a new input symbol.
4. If the next symbol is) , all pushdown stores which do not have B or R at the top are dismantled. T_2 removes the top symbol from the remaining pushdown stores and reads a new symbol. An h is placed in any pushdown stores which are completely emptied.
5. If the symbol read is h, T_2 checks to see if any pushdown stores contain an h. If so, T_2 empties all pushdown stores and enters state q_1. Under no other circumstances does T_2 enter stage q_1. If no pushdown store contains an h, T_2 empties all pushdown stores and enters state q_s. Under no other circumstances does T_2 enter state q_s.

A sample analysis by T_2 will now be displayed. This computation consists of two columns. In the first column, the input symbol will be given. In the second column, each line starting with the line on which the current input symbol is written and ending with the line just above the next input symbol will contain the contents of one pushdown store. The bottom of each pushdown will be to the right with the top at the left. The pushdown stores associated with an input symbol will show the contents when the symbol is read.

Input Symbol	*Pushdown Stores*
h	Λ
(	F
x	F + F
	F + F + FR
	F + FB
	F + FB + FR
	+ F
+	+ FR + F
	+ F + FR
	+ FR + F + FR
	+ FB
	+ FR + FB
	+ FB + FR
	+ FR + FB + FR
x	F
	FR + F
	F + FR
	FR + F + FR
	FB
	FR + FB
	FB + FR
	FR + FB + FR
)	h
	+ FR
	R + F
	+ FRR + F
	+ FR
	+ FR + FR
	R + F + FR
	+ FRR + F + FR
	B
	+ FRB
	R + FB
	+ FRR + FB
	B + FR

Input Symbol	*Pushdown Stores*
	+ FRB + FR
	R + FB + FR
	+ FRR + FB + FR
+	+ F
	+ F + FR
	h
	+ FB
	+ FR
	+ FB + FR
x	F
	F + FR
	FB
	FR
	FB + FR
)	h
	+ FR
	+ FR
	+ FR + FR
	B
	+ FRB
	R
	+ FRR
	B + FR
	+ FRB + FR
h	h
	h
	+FR
—	Λ

The fact that two pushdown stores end containing only h shows that there are two different analyses of the one input formula . . . (x + x) + x). The successive contents of the two pushdown stores which end containing h are:

Input Symbol	*Pushdown Store #1*	*Pushdown Store #2*
h	Λ	Λ
(	F	F
x	F + FB	F + FB + FR
+	+ FR + FB	+ FB + FR
x	FR + FB	FB + FR
)	R + FB	B + FR
+	+ FB	+ FR
x	FB	FR
)	B	R
h	h	h
—	Λ	Λ

It is a simple exercise to construct the pushdown-store transducer P_5 which generates L (G_4) where:

L designates the first production, A $\rightarrow$ (A + A.

R designates the second production, A $\rightarrow$ A + A).

B designates the third production, A $\rightarrow$ (A + A).

x designates the terminal production, A $\rightarrow$ x.

From this transducer, the inverse machine P'_5 can be constructed. Then the input formula used above will give as output either BxRxx or RxBxx (called generating formulas).

Generating formulas are not obtained by T_2 but they can be if the operations of T_2 are extended. Perhaps the simplest way is for T_2 to save the contents of the pushdown stores which end containing only h. One way to do this is for T_2 to add an extra pushdown store for each one established to store predictions. That is, call the pushdown stores, in which predictions are remembered, prediction pools, and associate with each prediction pool another pushdown store called the analysis pool. Whenever a prediction pool is changed, T_2 writes first * and then the entire contents of each prediction pool in each corresponding analysis pool. When a prediction pool is dismantled, the corresponding analysis pool is also emptied and dismantled. At the end of the input formula, T_2 could either simply copy on an output tape the contents of analysis pools which correspond to prediction pools which end containing h or these analysis pools could be analyzed further to obtain the generating formulas.

It should be emphasized that T_2 is a Turing machine while P'_5 is not. However, T_2 will give all analyses of an input formula whereas P'_5 provides, at most, one generating formula per input formula (P'_5 relates directly with the first methods of predictive analysis). If it is desired to use P'_5 to obtain all generating formulas, a supervisory or monitoring machine must be used in conjunction with it, and this total machine (i.e., the supervisory machine plus P'_5) also will constitute a Turing machine.

Finally, note that a grammar of the language to be analyzed is used explicitly in the predictive-analysis process but only implicitly in the inverse machine. That is, the assumed grammar is displayed explicitly in the predictive-analysis algorithm and can be determined by inspection. The grammar underlying a nondeterministic machine is only implied by the structure of the machine (or to a lesser extent by the generating formulas, in the case of a transducer) and can only be determined by an analysis of that structure (or of the generating formulas). In either case, however, the grammar must be known before a machine can be constructed. In the case of natural languages, one of the main problems is that the structure of the language to be analyzed is not known. Even if it is assumed to be phrase structure, as it must be if predictive analysis in a pure form is to work, the discovery of a practical grammar is not an easy task.

5. BIBLIOGRAPHY

1. BAR-HILLEL, Y., PERLES, M., and SHAMIR, E., "On formal properties of simple phrase structure grammars," *Zeitschrift für Phonetik, Sprachwissenschaft und Kommunicationsforschung,* Band 14, Heft 2 (1961), pp. 143-172.
2. BURKS, A. W., WARREN, P. W., and WRIGHT, J. B., "An analysis of a logical machine using parenthesis-free notation," *Math. Tables and Other Aids to Computation,* Vol. VIII, No. 46 (1954), pp. 53-57.
3. CHOMSKY, N., "On certain formal properties of grammars," *Information and Control,* Vol. 2, No. 2 (1959), pp. 137-167.
4. CHOMSKY, N., "Formal properties of grammars." In R. R. Bush, E. H. Galanger, and R. D. Luce (Eds.), *Handbook of Mathematical Psychology,* Vol. 2, New York: Wiley (forthcoming).
5. DAVIS, M., *Computability and Unsolvability,* New York: McGraw-Hill (1958).
6. EVEY, R. J., "A fail-safe predictive translation algorithm for the incomplete parenthetic language," *Math. Linguistic Seminar Papers,* Harvard, Vol. VI (1960).
7. EVEY, R. J., "The Theory and Applications of Pushdown Store Machines," Doctoral Thesis, Harvard University (1963).
8. FISCHER, PATRICK C., "On Computability by Certain Classes of Restricted Turing Machines," Paper presented at Conference on Switching Circuits and Automata Theory, Chicago, Oct., 1963.
9. GINSBURG, S., and ROSE, G. F., "Operations which preserve definability in languages," *ACM Journal,* April, 1963, pp. 175.
10. GROSS, M., "On the equivalence of models of language used in the fields of machine translation and information retrieval," unpublished report (1962).
11. HUZINO, S., "On some applications of the pushdown store technique," Memoirs of the Faculty of Science, Kyushu University, Vol. XV, No. 1 (1961).
12. KUNO, S., and OETTINGER, A. G., "Multiple path syntactic analyzer," Proc. of IFIP Congress, Munich, Germany (1962).
13. KUNO, S., and OETTINGER, A. G., "The Syntatic Ambiguity of English," Proceedings of FJCC, Nov., 1963.
14. MINSKY, M. L., "Recursive unsolvability of Post's problem of tag," *Annals of Math.,* Vol. 74 (1961), pp. 437-455.
15. OETTINGER, A. G., "Automatic syntactic analysis and the pushdown store," *Structure of Language and its Mathematical Aspects,* Proc. of Symposia in Applied Math., Vol. 12, Amer. Math. Soc., Providence, R. I. (1961).
16. RABIN, M. O., and SCOTT, D., "Finite automata and their decision problems," *IBM Journal of Research and Development,* Vol. 3, No. 2 (1959), pp. 114-125.
17. SAMELSON, K., and BAUER, F. L., "Sequential formula translation," *Comm. of the ACM,* Vol. 3, No. 2 (1960), pp. 76–83.
18. SHERRY, M. E., "Syntactic Analysis in Automatic Translation," Doctoral Thesis, Harvard University (1960).
19. YNGVE, V., "A model and an hypothesis for language structure," Proc. of Amer. Philosophical Soc., No. 104 (1961), pp. 444-446.

AN INTERRUPT CONTROL FOR THE B5000 DATA PROCESSOR SYSTEM

R. V. Bock
Burroughs Corporation
ElectroData Division
Pasadena, California

INTRODUCTION

Today it is an accepted fact that for a data processing system to operate efficiently, a comprehensive executive program is required to control dynamically the use of the system. The purpose of such a program is to increase throughput by reducing lost system time. The effectiveness of the executive routine will, to a very large extent, be determined by the special hardware features providing the interface between the executive program and object program.

Clearly the requirements of this interface are determined by what functions the executive routine will have to perform. These functions can be enumerated as follows:

1. Control all Input/Output Operations.
2. Control all scheduling, selecting and loading of programs.
3. Provide for utilization of multiple processors.
4. Control all storage allocation.
5. Handle exceptional conditions occurring in the running of object programs.
6. Log system utilization.
7. Respond to changes in system configuration.
8. Determine and notify the operator of system malfunction.
9. Control service routines.

Each of these functions requires the establishment, detection and selection of conditions throughout the system. In addition, provisions must be made for the executive routine to seize control from or transfer control to a given object program.[1]

This paper, using these requirements as the criteria for an interrupt control, will describe how the B5000 System embodies such an automatic interrupt control.

The description of the interrupt control is divided into three major sections. 1. The B5000 organization, which provides the environment in which the interrupt control will exist. 2. An analysis of the generalized interrupt operation which describes how the interrupt control forms the link between the object program and the executive routine. 3. A description of the interrupt control as implemented in the B5000 System.

B5000 ORGANIZATION

The B5000 System is modular in organization, consisting of Processors, I/O Channels, Memories, Central Control and various peripheral equipment. Each unit is modular in operation as well as construction. This modularity results in all units operating in an asynchronous manner. The interrupt control acts as controller for the system, sensing signals indicating the completion of operations or causing other units to commence operation.

Information and Control Structure

The basic information or control word handled in the B5000 is 48-bits in length. In Figure 5, the seven basic types of words used in the B5000 System are shown. Each type of word fulfills a specific function and these functions are: Operands—standard computational unit; Data Descriptors—define the base address of data areas in memory; Program Descriptors—define the base address of a program segment; Control Words—used to control subroutines, nesting operations, interrupt exists and returns; and Program Words which contain four syllables (operators) per word.

Processor State

There are two states: Normal State and Control State. All object programs are executed in Normal State. Control State is reserved for the use of the Master Control Program (executive routine) for the B5000. In Control State all syllables that are allowed in Normal State are permissible, plus a special group of operators intended for use by the Master Control Program.

The first 512 words of core storage are accessible only when in Control State. This area of memory is referred to as Control State Memory.

Program Reference Table—Stack—Program Area

Associated with each program being run, object or Master Control, are three areas of memory: the Program Reference Table (PRT), the Stack and the Program Area (PA).

The Program Reference Table consists of up to 1,024 words. This table can be relocated throughout memory. The base address of the PRT is specified by the *r* register. The PRT will hold simple variables, Data Descriptors, Program Descriptors and Control Words.* [3]

The Stack is an area of memory used for temporary storage. When a program becomes active, an association is established between the stack for that program and the *a* and *b* registers in the Processor. The association is established by the *s* register, which is set to point at the top word in the memory stack. The *a* and *b* registers function then as the top two registers in the active stack.

As words are stored in the memory stack, the *s* register is incremented. The removal of words from the memory stack will cause the *s* register to be decremented. In this way, the *s* register always addresses the current top of the memory stack.

For all object programs, the stack is always located immediately preceding the PRT in memory. Therefore, the *r* register forms the upper bound for the stack. If at any time the *s* register equals the *r* register, a stack overflow interrupt will occur.

Associated with the *a* and *b* registers are two occupancy flip-flops, arof and brof, which determine if these registers contain valid information. Prior to the execution of a word mode syllable, the occupancy flip-flops are tested to determine if the *a* and *b* registers are empty or full, as required by the syllable to be executed. If the registers are not in the required state, the stack is adjusted; that is, the stack is pushed up or down, whichever is required.

The Program Area (PA) is an area of memory that holds one or more program segments. The *c* register addresses program words. As each program word is fetched from memory, the *c* register is increased by 1. The *p* register holds the current program word. The syllables are taken from the *p* register one at a time and placed into the *t* register for execution.

General Processor Operation

In Figure 3, a Processor is shown with some of its registers. Also shown is the system's core memory. Several separate programs are shown in the core memory. Each program consists of a program segment (s), a Program Reference Table, and a Stack.

The Processor is shown executing the i^{th} program. Program words are brought from the program area as addressed by *c* and are placed in the *p* register. From *p*, syllables are taken,

* Table 2 provides a list of registers and special flip-flops. A brief description of their functions is also provided.

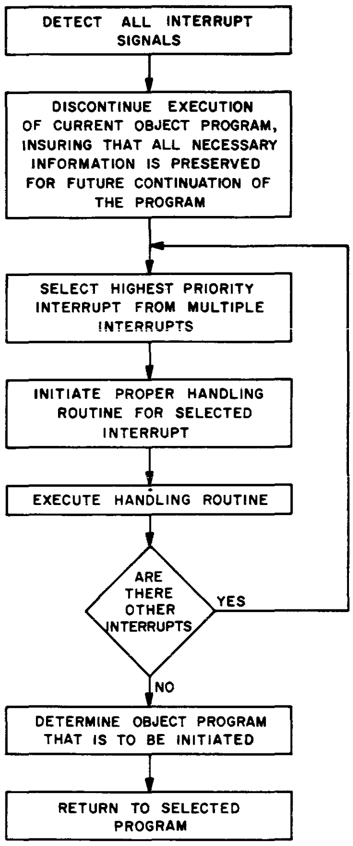

DETECT ALL INTERRUPT SIGNALS
DISCONTINUE EXECUTION OF CURRENT OBJECT PROGRAM, INSURING THAT ALL NECESSARY INFORMATION IS PRESERVED FOR FUTURE CONTINUATION OF THE PROGRAM
SELECT HIGHEST PRIORITY INTERRUPT FROM MULTIPLE INTERRUPTS
INITIATE PROPER HANDLING ROUTINE FOR SELECTED INTERRUPT
EXECUTE HANDLING ROUTINE
ARE THERE OTHER INTERRUPTS
YES
NO
DETERMINE OBJECT PROGRAM THAT IS TO BE INITIATED
RETURN TO SELECTED PROGRAM

Figure 1. Generalized Interrupt Procedure.

one at a time, and are placed into the *t* register under control of the *l* register. Syllables, as executed, will bring operands directly from the PRT, or from other areas in memory via descriptors found in the PRT, and place them in the stack. As results are formed in the stack they will be placed in the PRT or in data areas as defined by data descriptors found in the PRT. This, then, shows how the PRT, Stack and PA are used.

INTERRUPT ANALYSIS

Figure 1 shows the interrupt procedure, depicting the essential operations that occur when a condition arises that requires action by the executive routine. This procedure forms the basis for the interrupt control in the B5000 System.

The Sensing of Interrupt Conditions

Interrupt sensing has generally been accomplished by programmatically interrogating a number of separate control bits or the contents of an interrupt register. This method will, of course, add execution time to the object program, which is objectionable in high speed data processing systems. The added time can be eliminated by automatic hardware sensing of interrupt conditions.

Storing the State of the Processor

When an interrupt occurs, the Processor must be released to the executive routine. In order to return later, the present state of the Processor will have to be preserved. The present state will also be required for the analysis of interrupts caused by the object program. The automatic storing of the Processor's registers when an interrupt occurs can decrease the time spent in entering the executive routine.

Interrupt Priority Selection

Priority of interrupts may, in general, be determined by their effect on the system and the urgency of response to each interrupt. Once determined, the weighting of the interrupts remains invariant so priority resolution may effectively be mechanized with hardware.

Handling Routine Selection

Individual interrupts will be treated by individual subroutines within the executive routine. Each time that an interrupt occurs a given handling subroutine will be entered. Once entered, the path followed in resolving the interrupt will vary, but the entry will always be to the same subroutine. Since each interrupt is associated with a unique handling routine, automatic branching may be provided by the hardware.

The Handling Routines

Each handling routine is tailored to the related interrupt. The variations in these routines prohibit an economical trade off of software for special hardware operations. Therefore these routines are mechanized using standard operators in conjunction with several special operators that are necessary to allow the executive routine to use certain features of the hardware, such as, the Real Time Clock, I/O Channels or other Processors.

Multiple Interrupts

Here the scheme that seems easiest to control and yet is sufficiently complete is to process one interrupt at a time, not allowing interruption of an interrupt. Once the highest priority interrupt has been processed, but before returning to the object program, a test is made to see if other interrupts exist. If others do exist, then they are processed according to priority.

Return to Object Program

Once all interrupts have been handled, the Processor is returned to the object program of highest priority as quickly as possible. This can be accomplished by restoring the Processor to the exact state it was in, prior to the last time the program was interrupted. Here again, special hardware may be used.

Interrupt Summary

This examination of the interrupt procedure for the executive routine leads to the conclusion that a comprehensive hardware interrupt control should have means to: 1. sense various conditions requiring action by the executive routine, 2. cause entry into the executive routine while preserving the state of the processor prior to entry, 3. determine highest priority for multiple conditions, 4. initiate the proper action, and 5. return control to the highest priority object program.

INTERRUPT CONTROL IN THE B5000

Interrupt Types

The B5000 has provisions for sensing 48 individual interrupts. At the present time only 40 of these are being utilized. A list of the active interrupts in order of priority is given in Table 1. It will be noted that certain of the interrupts are referred to as being syllable dependent (i.e., operator dependent). The term "syllable dependent" is used to indicate that these interrupts can occur only because of some direct action on the part of the current syllable being executed in the corresponding Processor. These interrupts form two identical groups, one group occurring in each Processor. Within each of these groups the individual interrupts are mutually exclusive. This fact was used in the implementation to reduce the number of flip-flops required to store the interrupts within these groups. The remaining interrupts are referred to as syllable independent; each of these has exclusive control of a specific bit in the interrupt register.

An examination of Table 1 may leave the reader wondering where the control and error conditions for peripheral equipment are detected. These conditions are found in four words (one for each I/O Channel) located in control state memory. The words are referred to as Result Descriptors, and are placed in memory at the end of each I/O Operation by the associated I/O Channel. When an "I/O finished" interrupt occurs, the Master Control Program will examine the appropriate Result Descriptor to determine if some control or error condition exists.

The Interrupt Registers

The interrupt control receives signals from all over the system. These signals are received at local centers where they are stored until they are interpreted.

In the B5000 these signals are staticized in three interrupt registers. One is in each of the Processors and one is in Central Control. The registers in the two Processors are identical, as are their input signals. These two registers receive all Processor-dependent interrupts. The interrupt register in Central Control receives and stores all external interrupts. Figure 2

FIGURE 2
System Inter-
rupt Sensing
RV Bock

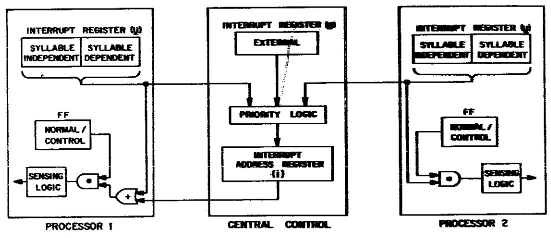


Figure 2. System Interrupt Sensing.

Table 1
B5000 INTERRUPTS

INTERRUPTS - TABLE 1

PRIORITY	NAME	FUNCTION	SYLLABLE DEPENDENCE	PROCESSOR DEPENDENCE
1	P1 Memory Parity Error	Memory malfunction.	Independent	Dependent
2	P1 Invalid Address	Used to determine system configuration or memory malfunction.	"	"
3	Timer Interval	Signal from Real Time Clock every 64/60 of a second. Used for record keeping.	"	Independent
4	I/O Busy	Used to determine system configuration or I/O malfunction.	"	"
5	Keyboard Request	Indicates operator has request to enter via keyboard.	"	"
6	Printer 1 Finished	End of Print cycle.	"	"
7	Printer 2 Finished	" " "	"	"
8	I1 Finished	I/O Data transfer complete - I/O Channel one is free.	"	"
9	I2 Finished	" - I/O Channel two is free.	"	"
10	I3 Finished	" - I/O Channel three is free.	"	"
11	I4 Finished	" - I/O Channel four is free.	"	"
12	P2 Busy	Used to determine system configuration or Processor 2 malfunct.	"	"
13	Inquiry Request	Indicates an Inquiry to process.	"	"
14				
15	Special Interrupts	-----------------	"	"
16				
17	P2 Memory Parity Error	Memory malfunction.	"	Dependent
18	P2 Invalid Address	System malfunction.	"	"
19	P2 Stack Overflow	Stack must be enlarged.	"	"
20	P2 Communication Operator	Means for object program to communicate with MCP.	Dependent	"
21	P2 Program Release	Indicates I/O area is ready to receive or transfer data.	"	"
22	P2 Continuity Bit	Indicates linked I/O memory areas.	"	"
23	P2 Presence Bit	Indicates need for information not in core-storage allocation.	"	"
24	P2 Flag Bit	Indicates end of data area.	"	"
25	P2 Invalid Index	Indicates attempt to index outside of data array.	"	"
26	P2 Exponent Underflow	-----------------	"	"
27	P2 Exponent Overflow	-----------------	"	"
28	P2 Integer Overflow	-----------------	"	"
29	P2 Divide by Zero	-----------------	"	"
30	P1 Stack Overflow	Stack must be enlarged.	Independent	"
31	P1 Communication Operator	Means for object program to communicate with MCP.	Dependent	"
32	P1 Program Release	Indicates I/O area is ready to receive or transfer data.	"	"
33	P1 Continuity Bit	Indicates linked I/O memory areas.	"	"
34	P1 Presence Bit	Indicates need for information not in core-storage allocation.	"	"
35	P1 Flag Bit	Indicates end of data area.	"	"
36	P1 Invalid Index	Indicates attempt to index outside data array.	"	"
37	P1 Exponent Underflow	-----------------	"	"
38	P1 Exponent Overflow	-----------------	"	"
39	P1 Integer Overflow	-----------------	"	"
40	P1 Divide by Zero	-----------------	"	"

shows the three interrupt registers and their interconnection.

The outputs of these registers are mixed at the input to the priority network. The highest priority interrupt is selected by the priority logic and is staticized in the Interrupt Address Register. At every clock pulse, the priority logic is sampled and the highest priority interrupt is placed in the *i* register. It will be noted that the name of this register is the Interrupt *Address* Register. The term address is significant and will be explained later in the paper. The interrupt sensing circuit sees only one interrupt at a time. As an interrupt is accepted by the sensing circuit for processing, the bit or bits in the corresponding local Interrupt Register are reset. The next clock pulse will place the next highest interrupt into the *i* register if one exists.

Control State Memory

The first 512 words of core memory are reserved for the exclusive use of the Master Control Program (MCP). This memory area is used to hold parts of the MCP that must always be in core memory, tables, the control state stack and miscellaneous control words. In addition to the above, 48 memory locations are set aside for linking an interrupt to the proper handling routine. One location is used for each of the possible 48 interrupts. A given interrupt is always associated with the same memory location. It will be recalled from above that the output interrupt register was referred to as the Interrupt Address Register and indeed the output from this register is an address relating each interrupt to a specific memory location. For example, if Processor One sets a "Flag Bit" interrupt the output of the interrupt

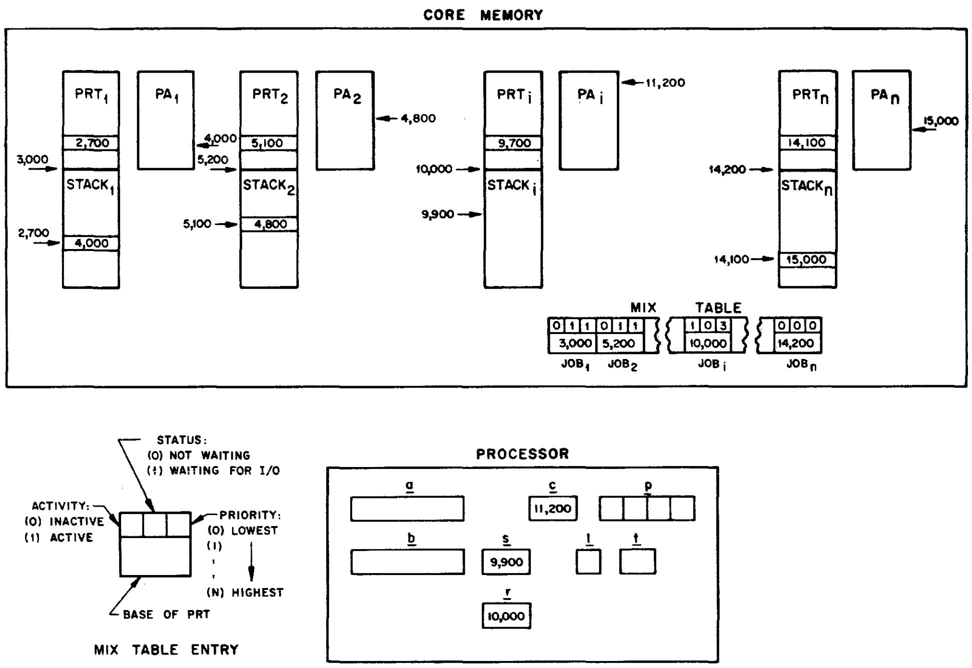


Figure 3. Simplified Processor Organization.

address register would be 56, addressing memory location 56. If a branch is made to this address, a four syllable linkage routine (one word) will be executed. The linkage routine will call the proper handling routine to process the interrupt.

Special Syllables for the Master Control Program

Four syllables are provided that function only in Control State. These syllables are treated as No-ops if encountered in Normal State.

The first of these syllables, Initiate I/O, will take a Data Descriptor found at the top of the stack and place this descriptor in a specific location in Control State Memory. Then a signal is sent to Central Control requesting that an I/O Channel be initiated. At this point the Processor is released to continue processing. Central Control will select a free I/O Channel and initiate this unit.

The Initiate Processor Two syllable functions in a very similar manner to the Initiate I/O. It too takes the top word in the stack (in this case it is an Initiate Control Word) and places it in the Control State Memory and sends a signal to Central Control to Initiate Processor Two. If Processor Two is not busy (if it is busy a malfunction exists) it will use the word stored in Control State Memory to establish its initial conditions. Processor One is free to proceed as soon as the initiate signal is sent to Central Control.

The third special syllable provided allows the Real Time Clock to be read. The syllable, Read Timer, will take the present value of the timer and place it at the top of the stack.

The last syllable, I/O Release, is used to mark a data area as "present". This indicates that an input area has been filled by some input device or that an output area has been emptied by an output device. This is accomplished by setting the presence bit in the descriptor associated with that data area.

A Description of the Dynamic Operation of the Interrupt Control

The following text will cover a system with one Processor. The results of adding a second Processor will be covered later. In addition to the above restriction, the discussion will be

Table 2
GLOSSARY OF REGISTERS AND FLIP-FLOPS USED IN THE B5000

NAME		DIMENSIONS		USAGE IN WORD MODE
Actual	Used in This Paper	ν	μ	
A	$\underline{a}$	48		Top position in stack
B	$\underline{b}$	48		Second level in stack
C	$\underline{c}$	15		Address of program word in $\underline{p}$
F	$\underline{f}$	15		Address used for sub-routine control
G	$\underline{g}$	3		Character address within $\underline{a}$
H	$\underline{h}$	3		Bit address within a character in $\underline{a}$
IOR	$\underline{u}$	7		Processor Interrupt Register
IAR	$\underline{i}$	6		Interrupt Address Register
K	$\underline{k}$	3		Character address within $\underline{b}$
L	$\underline{l}$	2		Syllable address within $\underline{p}$
M	$\underline{m}$	15		Address register used to bring words into the stack
N	$\underline{n}$	4		Counter
P	$\underline{p}$	48		Holds program word
R	$\underline{r}$	9		Base of PRT - upper limit of stack
S	$\underline{s}$	15		Address of top word in memory stack
T	$\underline{t}$	12		Holds syllable being executed
V	$\underline{v}$	3		Bit address within a character in $\underline{b}$
X	$\underline{x}$	39		Extension of $\underline{b}$
MEM	$\underline{M}$	49	2^{15}	Core Memory
MSFF	$\underset{\smile}{\text{msff}}$	1		Used in sub-routine control
NCSF	$\underset{\smile}{\text{ncsf}}$	1		State flip-flop - indicates whether processor is operating in Normal State or Control State
SALF	$\underset{\smile}{\text{salf}}$	1		Used in sub-routine control
CWMF	$\underset{\smile}{\text{cwmf}}$	1		Mode flip-flop - indicates whether processor is executing a word mode or character mode program
AROF	$\underset{\smile}{\text{arof}}$	1		$\underline{a}$ register occupancy flip-flop -indicates whether $\underline{a}$ is full or empty
BROF	$\underset{\smile}{\text{brof}}$	1		$\underline{b}$ register occupancy flip-flop -indicates whether $\underline{b}$ is full or empty

limited to a system where the Processor, at the outset, is operating in Normal State, executing a Word Mode Program. Figure 4 provides a synopsis of the following discussion. In addition to Figure 4, four microprograms are given. These programs are not essential to the understanding of the following discussion, but provide more detail as to the logical steps taken during interrupt. Table 3 provides a summary of the notation used in these microprograms.† [3]

Fetch and Interrupt Sensing—Microprogram 1

As the execution of each syllable is completed, an automatic test is made to see if an interrupt exists anywhere in the system. Three separate signals are sensed for this purpose. 1. The output of the *i* register, which will indicate an interrupt anywhere in the system, that occurred two or more clock pulses ago. 2. The output of the interrupt register (*u*) located in the Processor, which will indicate processor dependent interrupts that occurred one or more clock pulses ago. 3. The input to the Processor Interrupt Register, which will indicate interrupts that occur as the syllable is being completed. As the test for interrupt is made, the next syllable is fetched, i.e., placed in the *t* register. If an interrupt does not exist, then processing is continued. If, however, an interrupt is present, then the occupancy flip-flop (trof) for the syllable register (*t*) is turned off, invalidating this syllable. The next clock pulse finding trof off and an interrupt present, will set into the *t* register a syllable code, Store for Interrupt (SFI), that is used to continue the interrupt operations. At the same time, the occupancy flip-flop is turned back on.

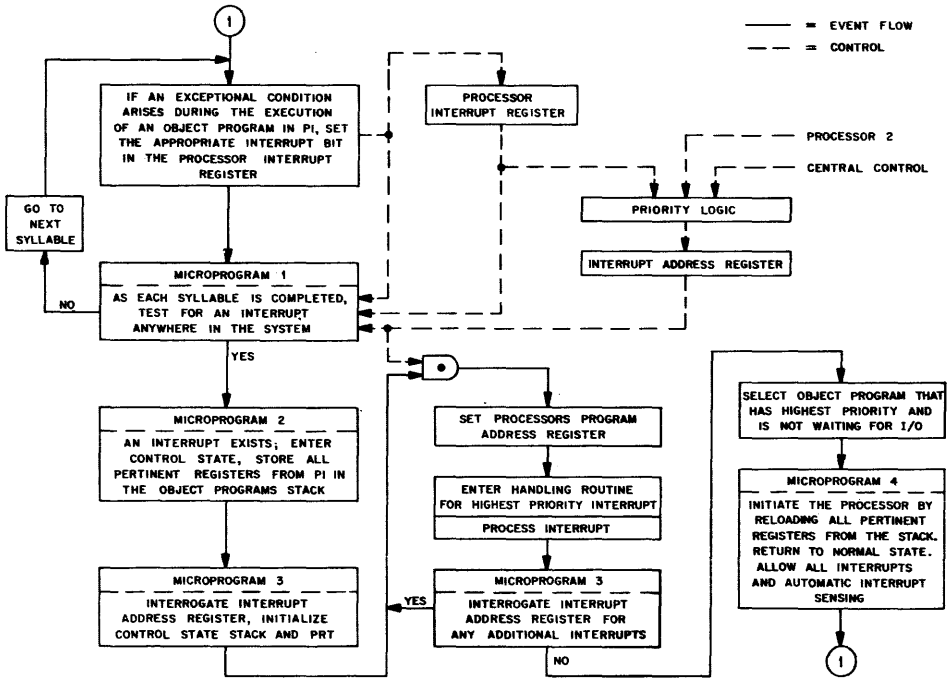
† These microprograms define only the logical steps necessary and do not give any indication of concurrence of actions or method of implementation use.

Store for Interrupt—Microprogram 2

The Store for Interrupt operator (SFI) will cause: 1. the present state of the machine to be stored in the stack, 2. the state of the Processor to be changed, and 3. the top of the stack address to be recorded in a control word placed in the PRT. This operator is not encountered in the normal program string, but is generated by the Processor when an interrupt occurs.

First, the state will be changed from "normal" to "control" and thus begin the transition from Object to Master Control Program.

The operation of freeing the Processor for use by the Master Control Program begins by



1
EVENT FLOW
CONTROL
IF AN EXCEPTIONAL CONDITION ARISES DURING THE EXECUTION OF AN OBJECT PROGRAM IN PI, SET THE APPROPRIATE INTERRUPT BIT IN THE PROCESSOR INTERRUPT REGISTER
PROCESSOR INTERRUPT REGISTER
PROCESSOR 2
CENTRAL CONTROL
PRIORITY LOGIC
INTERRUPT ADDRESS REGISTER
GO TO NEXT SYLLABLE
NO
MICROPROGRAM 1
AS EACH SYLLABLE IS COMPLETED, TEST FOR AN INTERRUPT ANYWHERE IN THE SYSTEM
YES
MICROPROGRAM 2
AN INTERRUPT EXISTS; ENTER CONTROL STATE, STORE ALL PERTINENT REGISTERS FROM PI IN THE OBJECT PROGRAMS STACK
SET PROCESSORS PROGRAM ADDRESS REGISTER
ENTER HANDLING ROUTINE FOR HIGHEST PRIORITY INTERRUPT
PROCESS INTERRUPT
SELECT OBJECT PROGRAM THAT HAS HIGHEST PRIORITY AND IS NOT WAITING FOR I/O
MICROPROGRAM 4
INITIATE THE PROCESSOR BY RELOADING ALL PERTINENT REGISTERS FROM THE STACK. RETURN TO NORMAL STATE. ALLOW ALL INTERRUPTS AND AUTOMATIC INTERRUPT SENSING
MICROPROGRAM 3
INTERROGATE INTERRUPT ADDRESS REGISTER, INITIALIZE CONTROL STATE STACK AND PRT
YES
MICROPROGRAM 3
INTERROGATE INTERRUPT ADDRESS REGISTER FOR ANY ADDITIONAL INTERRUPTS
NO
1

Figure 4. Dynamic Interrupt Operation for Processor One.

Table 3
SUMMARY OF MICROPROGRAM NOTATION

LOGICAL OPERANDS	NOTATION	DIMENSION	
Scalar - Flip-Flop	$\underset{\smile}{\text{arof}}$		
Vector - Register	$\underline{a}$	$\nu(\underline{a})$	Number of Bits in Register
Matrix - Memory	$\underline{\underline{M}}$	$\nu(\underline{\underline{M}})$ $\mu(\underline{\underline{M}})$	Number of Bits in Word Number of Words in Memory

ELEMENTARY OPERATIONS	NOTATION	DEFINITION
Sum	$a \leftarrow b+c$	a = Algebraic Sum of b and c
Negation	$u \leftarrow \bar{v}$	$u = 1$ iff $v = 0$
And	$u \leftarrow v \wedge w$	$u = 1$ iff $v = 1$ and $w = 1$
Or	$u \leftarrow v \vee w$	$u = 1$ iff $v = 1$ or $w = 1$
+ Reduction	$a \leftarrow +/\underline{v}$	$a = (\cdots((\underline{v}_1+\underline{v}_2) + \underline{v}_3)\cdots) + \underline{v}_\nu)$
Catenation	$\underline{b} \leftarrow \underline{f} \oplus \underline{s}$	$\underline{b} = (\underline{f}_1, \underline{f}_2, \cdots \underline{f}_\nu, \underline{s}_1, \underline{s}_2, \cdots \underline{s}_\nu)$
Base 2 Value $\underline{s}$	$a \leftarrow \perp \underline{s}$	a = Base 2 Value of the Vector $\underline{s}$
Residue Mod k	$j \leftarrow k \mid i$	$i = kg+j$; $1 \leq j < 1+k$; and g is integral
Left Rotation	$\underline{u} \leftarrow k \uparrow \underline{v}$	$\underline{u}_i = \underline{v}_j$ where $j = \nu\ (\underline{v}) \mid (i+k))$
Right Rotation	$\underline{u} \leftarrow k \downarrow \underline{v}$	$\underline{u}_i = \underline{v}_j$ where $j = \nu\ (\underline{v}) \mid (i-k)$
Binary Representation of $\underset{\smile}{\text{IIN}}$	$\underline{t} \leftarrow \rho(\underset{\smile}{\text{IIN}})$	
Compression	$\underline{s} \leftarrow \underline{u}/\underline{v}$	$\underline{s}$ is obtained from $\underline{v}$ by suppressing each $\underline{v}_i$ for which $\underline{u}_i = 0$.

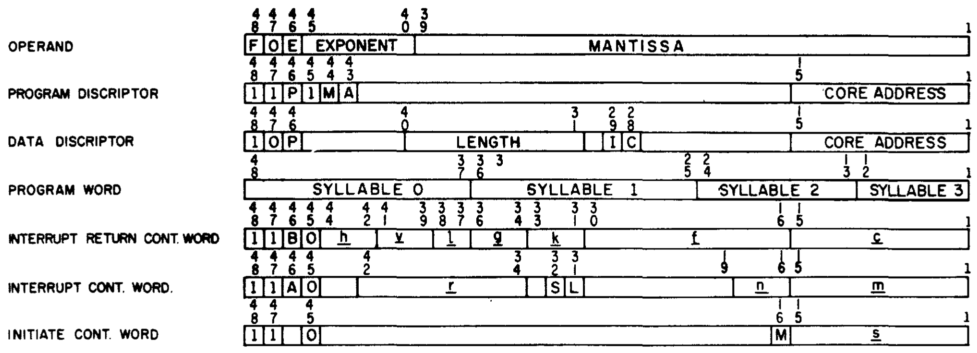
SPECIAL VECTORS	NOTATION	DEFINITION
Full Vector	$\underline{\epsilon}\ (n)$	Vector of ones, dimension n
Unit Vector	$\underline{\epsilon}^j(n)$	$\underline{\epsilon}j = 1$, $\underline{\epsilon}_i = 0$ for $i \neq j$
Prefix Vector	$\underline{\alpha}^j(n)$	$\underline{\alpha}_i = 1$ for $i \leq j$, $\underline{\alpha}_i = 0$ for $i>j$
Suffix Vector	$\underline{\omega}^j(n)$	$\underline{\omega}_i = 0$ for $i \leq n - j$, $\underline{\omega}_j = 1$ for $i<n-j$

BRANCHING CONVENTIONS

$$\downarrow a : b \qquad \overset{R}{\longmapsto}$$

The statement to which the arrow leads is executed next if the relationship (aRb) = 1; otherwise the next statement in sequence is executed. An unlabelled arrow is always followed.

NOTE: 1 - Origin Indexing is assumed throughout.



P	= PRESENCE BIT	F	= FLAG BIT
M	= MODE BIT	O	= OPERAND SIGN
A	= ARGUMENT BIT	E	= EXPONENT SIGN
I	= INTEGER BIT	B	= brof
C	= CONTINUITY BIT	A	= arof
S	= MARK STACK BIT		
L	= LEVEL BIT		

Figure 5. B5000 Word Format.

determining if the top two positions for the stack, *a* and *b* registers, are occupied. If either, or both, are occupied, their contents will be pushed down into the Core Memory Stack and the stack address counted up appropriately. Following the stack adjustment, two control words are constructed and placed in the stack. The two words are referred to as the Interrupt Control Word and the Interrupt Return Control Word. Figure 5 shows the organization of these words. These two control words store all registers and control flip-flops that will be needed to initiate the Processor after the interrupt has been processed.

With all the necessary control information now in the stack, the address of the top of the stack, the *s* register, must be preserved in a location that can be accessed by the Master Control Program. The MCP keeps a table (mix table) of all active PRT addresses. The top of the stack address may be stored in the PRT and located via the mix table. A specific word in every PRT is set aside for this purpose. This location is referred to by its relative address, $r \oplus \epsilon^3$ (6). It should be noted that the object program is now completely divorced from the Processor and is self-contained in the stack, PRT and program area.

The Processor is now free to call the proper portions of the Master Control Program to handle the interrupt. The *t* register is automatically changed from the Store for Interrupt syllable to the Interrogate Interrupt Syllable ($\underbrace{\text{INI}}$) to continue the interrupt process.

Interrogate Interrupt—Microprogram 3

The Interrogate Interrupt syllable is used to:

1. Determine if an interrupt condition exists.
2. Set up the Processor for the execution of the MCP.
3. Transfer control to the proper subroutine within the MCP.
4. Reset the offending interrupt bit(s).

The output of the *i* register is tested to ascertain if an interrupt condition is present in the system. If an interrupt is not found to exist, the syllable is terminated and the next syllable in sequence is executed. If, on the other hand, an interrupt is detected, the syllable is continued and the operations defined below will be performed. This test is included because the syllable is not only used when an interrupt has been detected in Normal State, but is also used by the Master Control Program to determine if other interrupts exist following the process-

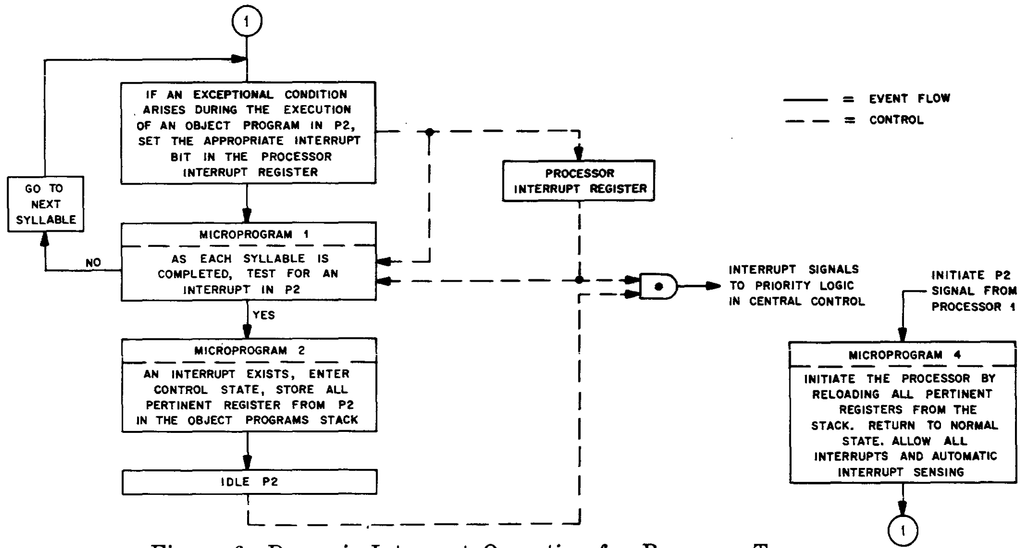

IF AN EXCEPTIONAL CONDITION ARISES DURING THE EXECUTION OF AN OBJECT PROGRAM IN P2, SET THE APPROPRIATE INTERRUPT BIT IN THE PROCESSOR INTERRUPT REGISTER
GO TO NEXT SYLLABLE
NO
MICROPROGRAM 1
AS EACH SYLLABLE IS COMPLETED, TEST FOR AN INTERRUPT IN P2
YES
MICROPROGRAM 2
AN INTERRUPT EXISTS, ENTER CONTROL STATE, STORE ALL PERTINENT REGISTER FROM P2 IN THE OBJECT PROGRAMS STACK
IDLE P2
PROCESSOR INTERRUPT REGISTER
INTERRUPT SIGNALS TO PRIORITY LOGIC IN CENTRAL CONTROL
= EVENT FLOW
= CONTROL
INITIATE P2 SIGNAL FROM PROCESSOR 1
MICROPROGRAM 4
INITIATE THE PROCESSOR BY RELOADING ALL PERTINENT REGISTERS FROM THE STACK. RETURN TO NORMAL STATE. ALLOW ALL INTERRUPTS AND AUTOMATIC INTERRUPT SENSING
1

Figure 6. Dynamic Interrupt Operation for Processor Two.

ing of the first interrupt. The use of this syllable by the Master Control Program eliminates the necessity of returning to normal state to detect other interrupts. It will be recalled that automatic interrupt testing is inhibited in control state.

Setting up the Processor for execution of the Master Control Program requires that: the *r* register be set to zero, which defines the base of the PRT for Control State, the *s* register is set to point at the bottom of the permanent Control State Stack and the *c* register is set from the contents of the *i* register.

The interrupt condition that initiates the MCP must now be reset. The interrupt bit(s) causing entry into the MCP is set to zero. Only this bit(s) need be reset since the *i* register will automatically remove the interrupt address with the input interrupt reset.

The Processor may now be released to the Master Control Program by fetching the link routine from the location defined by the Program Address Register (*c*).

Handling Routines

The Master Control Program will now proceed to process the interrupt. The interrupt may be due to some exceptional condition in the program that was interrupted or it may be totally unrelated to this program. The cause for the interrupt must be determined and the proper action taken to satisfy this condition.

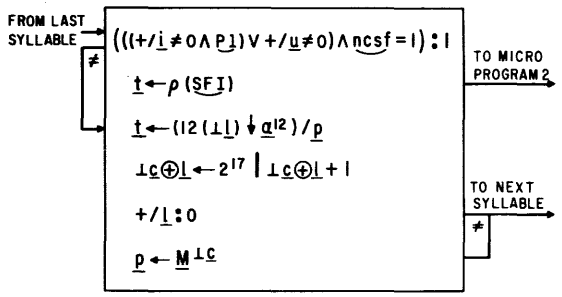
Having properly handled the interrupt, the Master Control Program must determine if other interrupts exist that require action by the MCP. If others do exist, entry into the proper subroutine is determined by the use of the $\underset{\smile}{\text{INI}}$ syllable defined above (see microprogram 3). If additional interrupts do not exist, then the executive routine must return control to a Normal State Program.

The return may be to any one of three types of programs: 1. the program just interrupted, 2. a program that was interrupted earlier, assuming the earlier program had a higher priority and had been waiting for input/output, or 3. a new program, assuming the interrupted program was finished, unable to proceed because of an exceptional condition, or in need of input/output.

Whichever program is to be initiated, the Master Control Program will select from the Mix Table the entry corresponding to that job (Figure 3). From this Mix Table entry, the PRT base address is procured. Using the base address of the PRT, the Initiate Control Word will be placed in the top of the Control State Stack and the Initiate Processor syllable will be executed.

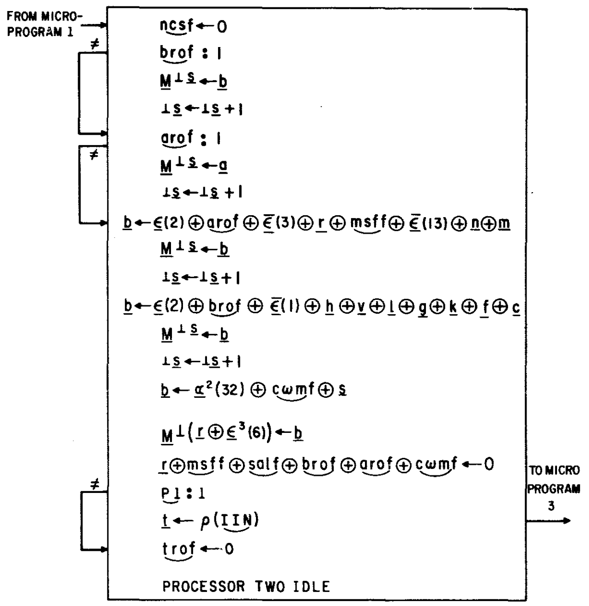
Initiate Processor—Microprogram 4

It will be remembered that when the Store for Interrupt was performed, the Initiate Control Word was formed (Figure 5). This word



P1=A SWITCH USED TO DETERMINE WHETHER PROCESSOR A OR B IS PROCESSOR ONE

Figure 7. Microprogram 1—Fetch.



P1= A SWITCH USED TO DETERMINE WHETHER PROCESSOR A OR B IS PROCESSOR ONE

Figure 8. Microprogram 2—Store for Interrupt

contained the top of the stack address plus the mode bit for the interrupted object program. The initiate syllable will place the stack address in the stack register (*s*) and the mode flip-flop will be set according to the mode bit in the control word.

The control words that are now at the top of the Object Program Stack are pushed up and their contents are distributed to the proper registers.

The state flip-flop is set to Normal State, which will allow the automatic sensing of interrupts and the setting of syllable dependent interrupts.

The Processor is now ready to proceed with the Object Program at the exact point where it was last interrupted.

Interrupts in a Two Processor System

In the B5000 System an optional second Processor is available. The second Processor is identical with the first in all respects.

The Processor that controls system operation is called Processor One. Either Processor may be assigned the role of Processor One by a mechanical switch located in Central Control. The Master Control Program runs only in Processor One and therefore, all interrupts including Processor Two's interrupts are handled by Processor One.

When an interrupt occurs in the second Processor, the appropriate bit will be set in its own interrupt register (Figure 2). The output from this register, as with Processor One, goes to the priority logic in Central Control. Processor Two senses automatically those interrupts that are generated in itself. All others are ignored. (The output of the *i* register is not sensed in Processor Two.) As in Processor One, interrupts are automatically sensed between syllables. An interrupt occurring will cause the same Store for Interrupt actions described for Processor One, except that at the completion of the Store for Interrupt the automatic Interrogate for Interrupt will not occur. Instead, the Processor will idle until such time as it is initiated again by Processor One. Figure 6 shows the sequence of events when an interrupt occurs in Processor Two.

An interesting problem occurs when considering interrupts for a second Processor. Handling Processor Two interrupts in the manner described above; a timing conflict can exist. Processor One could start to handle the interrupt from Processor Two before Processor Two has fully completed the storing of its registers. To prevent this, all Processor Two interrupts are gated by the Processor Two Not Busy Level (idle condition). In this way, the priority logic does not receive the interrupt until Processor

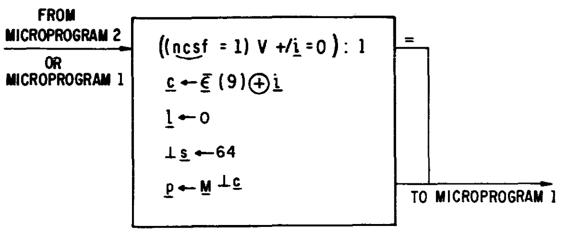

FROM MICROPROGRAM 2 OR MICROPROGRAM 1
((ncsf = 1) V +/i = 0) : 1
c ← ε (9) ⊕ i
i ← 0
⊥s ← 64
p ← M ⊥c
=
TO MICROPROGRAM 1

Figure 9. Microprogram 3—Interrogate Interrupt.

Two is idle, hence preventing Processor One from attempting to process the interrupt prematurely.

ABSTRACT

The design and operation of an interrupt control for the B5000 System is discussed. The factors considered when deciding which parts of the interrupt structure should be mechanized with hardware are given. The effect of multiprocessors on the interrupt control is discussed.

BIBLIOGRAPHY

1. BUCHHOLZ, WERNER: Planning a Computer System, McGraw Hill Book Company Inc., New York, New York, 1962.
2. Operational Characteristics of the Processors for the Burroughs B5000, Burroughs Corporation, Detroit 32, Michigan, 1961.
3. IVERSON, K. E.: A Programming Language, John Wiley & Sons, Inc., New York, New York, 1962.
4. BARTON, R. S.: A New Approach to the Functional Design of a Digital Computer, Proceedings of the Western Joint Computer Conference, Vol. 19, pgs. 393–396, May, 1961.

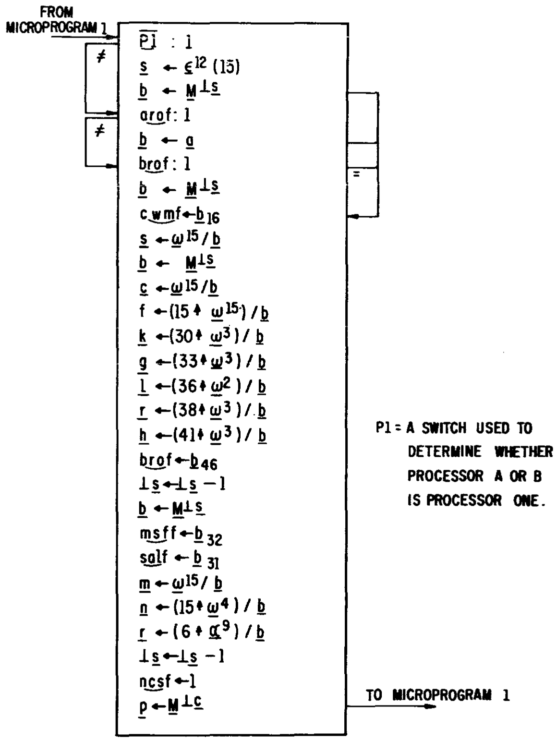


Figure 10. Microprogram 4—Initiate Processor.

5. Master Control Program Characteristics, Burroughs Corporation, Detroit 32, Michigan, 1961.

THE MECHANIZATION OF A PUSH-DOWN STACK

C. B. Carlson
Burroughs Corporation
ElectroData Division
Pasadena, California

INTRODUCTION

The present trend in programming General Purpose Computers is toward the use of Problem Oriented Languages such as ALGOL and COBOL. These languages provide convenient coupling between Man and Machine but impose additional requirements such as Automatic Compilers.

In the past, application of Problem Oriented Languages has been hampered because the Compiler has been obliged to work in machines that were designed for less sophisticated languages. The efficiency of both the Compiler and the Object Program can be greatly increased if the hardware provides nearly a "one for one" correspondence between the Source Language Operators and the Machine Instructions.

Early in the design of the Burroughs B5000 Processor, the decision was made to create an Operator Set and Logical Functions that provide, as nearly as possible, this "one for one" relationship.

The resulting machine has some interesting features and among them is an Automatic Push-Down Stack. This Stack has been built into the hardware and is now in operation.

It is the purpose of this paper to describe the organization, automatic control and application of this Stack.

GENERAL SYSTEM ORGANIZATION

The B5000 is a medium size general purpose data processing system designed for both scientific and commercial data processing. It is modular in design, permitting changes in the system configuration to fit the requirements of the user. Systems are made up from combinations of the following modules:

1 Central Control Unit
1 or 2 Processors
Up to 8 Memory Units or 4096 words each
1 or 2 Magnetic Drums of 32786 word capacity
1 to 4 Input/Output Control Units
1 to 16 Magnetic Tape Transports
1 or 2 High Speed Printers
1 or 2 Card Reader Units, either 200 or 800 CPM
1 Card Punch Units, either 100 or 300 CPM
1 or 2 Paper Tape Readers }
1 or 2 Paper Tape Punches } A total of 3 units
1 Control Console

The system operates at a one megacycle clock frequency, performing a 39-bit addition in one clock pulse. The processing speed is so much greater than the speed of the peripheral machines that full utilization of the processor becomes a problem. Much of the imbalance in speed between the processor and the I/O operations is absorbed by parallel operation of the Processor and I/O Units.

Parallel operation is facilitated by a Memory Exchange switching interlock whereby memory is time shared by, and accessible to, both the Processor and the I/O Control Units. There is no direct transfer of data between these units but data flow is effected by means of individual memory read or write operations originating with the Processor or the I/O Control Units. Thus a Processor may initiate an I/O operation that will be carried to completion by the I/O Control Unit. Once the I/O operation is in progress the Processor is free to execute another program. This ability to transfer the Processor from one program to another is called multiprocessing and will be discussed in further detail.

An executive routine called the Master Control Program (MCP) is required to maintain efficient control of these operations. The MCP schedules all jobs assigned to the system and controls such operating functions as assigning memory space, compiling object programs and directly I/O operations. An extensive automatic interrupt system is built into the hardware to provide the necessary controls for the MCP.

THE HARDWARE STACK

Into this system organization is incorporated a push-down stack that provides an automatic temporary store for parameters and control information relating to the program currently being executed. We will now consider the hardware that constitutes this stack.

Actually the amount of this hardware is small.

1. An assigned memory area of not more than 1024 words.
2. A 15-bit address register *s*.*
3. An identical register *f*.
4. Two 48-bit registers *a* and *b* in the Processor and their satellite flip-flops arof and brof.
5. The associated transfer paths consisting of cables and logical gates.

Memory Stack

The memory space assigned to the stack is specified by the memory allocation routine of the MCP. The size and location of this space may be changed from time to time by the MCP. However, to an object program, the top of the stack is specified as the absolute address con-

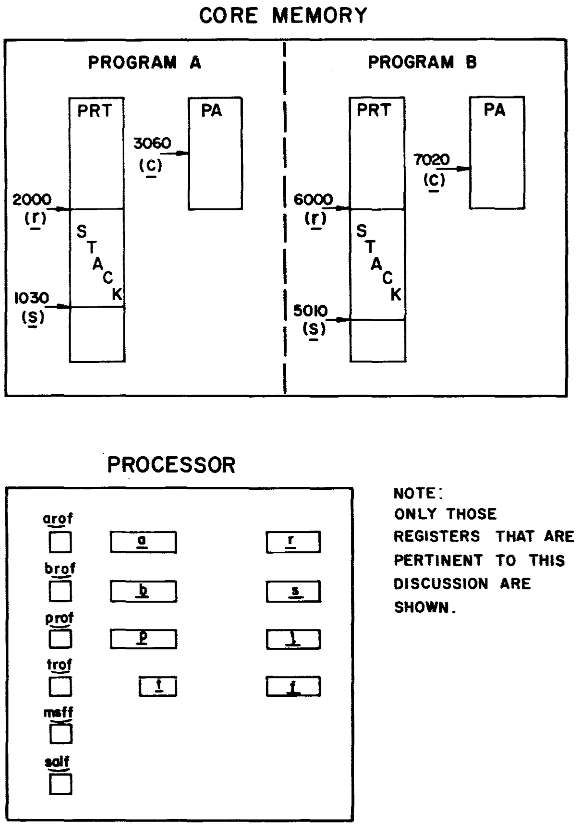


Figure 1. Block Diagram Processor and Core Memory.

tained in the register *s*. As words are stored in the stack, the *s* register is incremented and as words are read from the stack the *s* register is decremented such that *s* is always pointing to the top word in the memory stack.

Stack underflow, i.e., the possibility of the *s* register counting down below the assigned Stack area, is checked programmatically. Essentially, the object program never attempts to access data that was never placed in the Stack.

Detection of Stack overflow is checked by the hardware. When an object program is brought into the core memory, the Master Control Program assigns contiguous memory areas to the Stack and the Program Reference Table (PRT). The PRT is positioned above the Stack such that the base of the PRT is the upper limit of the Stack.† When a Processor is executing an object program the *r* register is set to the address of the base of the PRT and it remains set to this value so long as the Stack is active.

* Figure 6 lists the registers and symbols referred to in this paper.

† See Figure 1 for a diagram of these memory areas.

The test for overflow is simply a test for $s = r$ and is performed each time s is counted plus one. When $s = r$, an interrupt condition exists which requires that the state of the program be stored in the Stack and that control be turned over to the MCP. To provide a buffer area in which to store this overflow information the s register is allowed to count several locations into the PRT. Since the assignment of space in the PRT is controlled by program it is a simple matter to reserve this space.

The Active Top of the Stack

When a Processor is executing an object program, any word that is placed in the a register or in the b register is considered to be in the Stack. The a register is the top position of the Stack and the b register is the second position. Either, or both of these positions may be full or empty, thus if the a register is full and the b register empty there is a void in the Stack. Full control of this condition is provided by the two satellite flip-flops $\underset{\smile}{\text{arof}}$ and $\underset{\smile}{\text{brof}}$ which indicate the occupancy of the a register and the b register. When $\underset{\smile}{\text{arof}}$ is true the word in a is a valid member of the Stack. If $\underset{\smile}{\text{arof}}$ is false the a register is empty. Likewise, $\underset{\smile}{\text{brof}}$ indicates the status of the b register.

CONTROL OF THE STACK

The hardware just described provides the mechanism to store data into and extract data from the Stack. It also provides automatic control of addressing at the micro level.

A higher level of mechanization exists in the logic for each operator syllable. Some of this logic acts directly to provide automatic functions such as stack adjustment, the generation and use of Control Words‡ and subroutine linkage control. However, the full effect of the Stack under control of the operators is not apparent until the operators are used in program sequence. The balance of this paper is devoted to a description of some features of the machine language operators and the application of the Stack to the execution of a simple ALGOL statement.

‡ Control Words are 48-bit words which contain register settings, flip-flop states, address fields, etc. The format of some control words and data words are shown in Figure 5.

Automatic Stack Adjustment

Essentially all the operator syllables or instructions, that comprise the B5000 machine language require that operations be performed in the a or b registers. In order to accomplish this it is first necessary to see that space is available in these registers or that the requisite data is pushed up from the Stack. This operation is called stack adjustment and automatic control of this function is mechanized as follows.

There are four possible states that the two flip-flops $\underset{\smile}{\text{arof}}$ and $\underset{\smile}{\text{brof}}$ can assume, at least one of which reflects the initial data requirements of any operator syllable. Some operators are binary and require two operands. Others are unary and require only one word in the a or the b register before they can perform their function. An automatic stack adjustment operation which loads a and/or b as needed, is an integral part of each operator and logical gates are built into the hardware which activate the necessary push-up and push-down operations.

The function of these gates can be described as a series of tests and actions. Figure 2 is a set of four Microprograms which define the Stack adjustment operation required to establish each of the four states for $\underset{\smile}{\text{arof}}$ and $\underset{\smile}{\text{brof}}$.

In the first Microprogram it is required that any valid words remaining in the a or the b registers be placed in the Memory Stack (pushed down). This is the state where $\underset{\smile}{\text{arof}} = \underset{\smile}{\text{brof}} = 0$. Actually the logical steps defined by the Microprogram are quite simple. First the vector y is tested for the value zero. If this is true then the operation is already complete, otherwise the next test is made—(Is the second term in y ($\underset{\smile}{\text{brof}}$) equal to 1?). If true we count the s register $+1$ to address an empty cell in the Stack. Having counted $s + 1$, a test is made for stack overflow, $(s{:}r)$. If there is no stack overflow $(s \neq r)$ then the content of the b register is stored in the memory location addressed by the content of s. The b register is now empty and $\underset{\smile}{\text{brof}}$ is set to zero to indicate this fact.

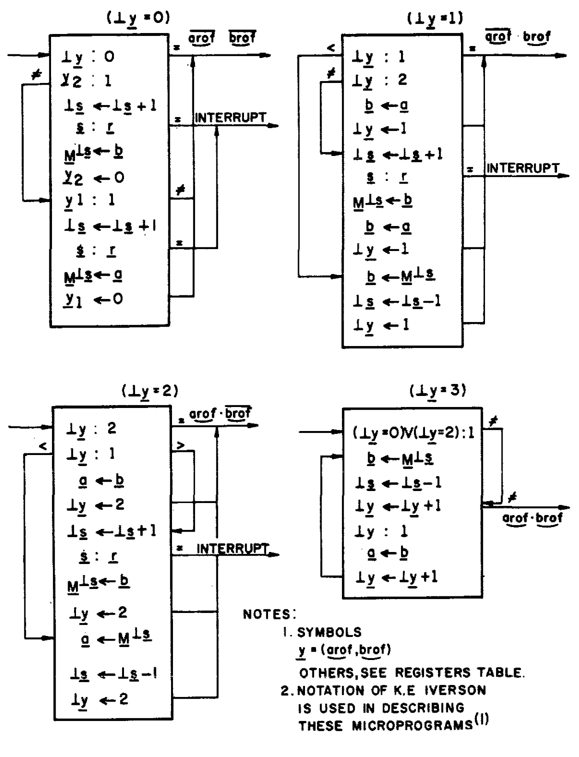

(⊥y = 0)
(⊥y = 1)
(⊥y = 2)
(⊥y = 3)
INTERRUPT
arof · brof
NOTES:
1. SYMBOLS
y = (arof, brof)
OTHERS, SEE REGISTERS TABLE.
2. NOTATION OF K.E IVERSON IS USED IN DESCRIBING THESE MICROPROGRAMS (1)

Figure 2. Iversion Microprograms.

The state of arof is next tested—(Is the first term in y (arof) equal to 1?). If it is not one, then both a and b registers are empty and the operation is finished. However, if arof is found equal to one, the contents of the a register are stored in memory in the same manner that b was stored and the operation is complete.

A study of these Microprograms makes it apparent that the hardware is automatically performing functions that would otherwise be done by program. The fact that these functions are common to almost all machine language operators justifies their hardware implementation.

Multiprocessing

During the execution of a program it is frequently necessary to stop the processing operations while a data area is filled or a new segment of program is brought into Core Memory. To allow the processor to stand idle during this I/O operation is wasteful and cannot be tolerated.

The programming concept of the B5000 provides that several programs may be in core at any time and that the Processor may switch from one to another under control of the MCP. Mechanization of this operation is accomplished in the following manner.

All the control information for a program is placed in three areas as it is brought into Core Memory. These areas are the Program Reference Table, the Stack and the Program Area (PA).

Figure 1 is a block diagram showing these areas. If the Processor is executing Program A and, before it has completed it, it is called upon to process Program B, the Processor will first store all the information it contains relative to Program A in the Stack. This includes the push-down of any data in the a register or the b register and the storage of control words that contain the essential state of the Processor. In this manner the integrity of Program A is preserved and the Processor is free to set its registers r, s and c to the addresses belonging to Program B. Subsequently, this Processor (or the second Processor in a two Processor System) may return to Program A, reset its r, s and c registers, restore itself from the information stored in the Stack and continue with the execution of Program A.

This complete independence and integrity of each program is the basis for efficient Multiprocessing. Programs can be written without regard for other programs that may be in the system at the same time.

Application of the Stack to an Object Program

The use of the Stack in conjunction with the PRT and the PA may be illustrated with a simple ALGOL Program:

```
BEGIN
REAL A,B,C,D;
A ← B+C×D;
END.
```

During compilation the Declaration "REAL" will reserve a location in the PRT for each of the four variables A, B, C, and D. Compilation also generates the following program string and stores it in the PA:
"OPDC_B, OPDC_C, OPDC_D, MUL, ADD, LITC_{0025}, STD"§

§ These are machine operator codes. More detail on their function may be found in 5.

Processing this object program proceeds as follows:

1. The first syllable $OPDC_B$ (an operand call on the word B) brings the Operand Word B into the Stack.
2. The second syllable places the Operand Word C in the Stack, positioned above B.
3. In like manner, D is placed in the Stack.

Since three words have been placed in the Stack through the *a* and *b* registers, the first word entered, B, has been automatically pushed down and is now in memory.

4. The next syllable encountered by the Processor is MUL (multiply) which operates on the top two words in the Stack, destroying C and D and leaving their product at the top of the Stack.
5. The fifth syllable, ADD (add), is also binary and at this point, a push-up is automatically initiated to obtain B. The addition is then performed in the *a* and *b* registers leaving at the top of the Stack the computed value for A.
6. The next operator is a Literal Call Syllable which brings the *r* relative address which is reserved for A, into the top of the Stack.
7. The final operator stores the value of A in the PRT location reserved for A, $(r + 25)$.

This program is comprised of Word Mode Operators, as are practically all operators that manipulate the Stack. In Character Mode the Stack remains intact and is used to pass parameters from and to Word Mode. However, operation of the Stack is basically a Word Mode Function.

In passing, it is interesting to note that there are seven terms in this original ALGOL statement and there are seven Program syllables in the compiled program. Although the correspondence is not always 100% as in this case, this illustrates the "one for one" relationship of operands, literals and operators that contributes to program and compiler efficiency.

|| Word Mode and Character Mode are two primary conditions under which the Processor operates. They are described in [5].

Application of the Stack to Subroutine Operation

Entering and leaving Procedures or Subroutines is similar to initiating and interrupting programs during Multiprocessing to the extent that both require the Processor to staticize one program while it addresses itself to another. In the case of Multiprocessing, this transfer of control is effected via the MCP whereas subroutine entry is performed by the object program. To provide the compiler and object programs with means to enter subroutines using a minimum amount of coding, a large part of the address logic has been built into hardware. Several powerful operators are provided which automatically generate or use Control Words which are stored in the Stack. The control of the Stack during a subroutine entry and return can best be illustrated by another ALGOL Program.

Assume a program calls upon a variable "X," as in the case shown in Figure 3. The object program will compile substantially as shown under "Main Program" and "Sub-

```
CALL ON PROCEDURE X:
                A←— X(Y,Z) + B--------;

PROCEDURE X:
            REAL PROCEDURE X(R,S);
                 VALUE (R,S);
                 X←—R x S
```

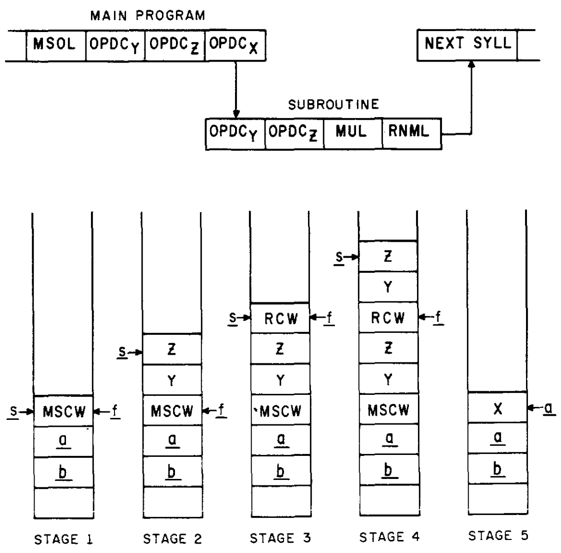


Figure 3. Call on Simple Procedure "X."

routine." At the bottom of the figure are shown five stages in the life of the Stack during the processing of this operator string.

Executing the Main Program syllables from left to right, the Processor encounters a Mark Stack Operator. This operator first pushes down into Memory Stack, any valid words in the *a* or *b* registers. It then builds a MSCW (Mark Stack Control Word) containing the *r* and *f* registers and msff and salf, and this MSCW word is pushed down into Memory Stack. The address of this MSCW is temporarily placed in the *f* register. Stage 1 shows the Stack with *b*, *a* and the MSCW in the Memory Stack.

The Parameters Y and Z are next placed in the Stack by two Operand Call syllables. Stage 2 depicts the addition of these parameters.

The next operator encountered in the Main Program is an Operand Call on a Program Descriptor. The ensuing operation builds a Return Control Word (RCW) from the pertinent registers in the Processor, including the *f* register, and pushes it into the Stack. The contents of the *f* register are now replaced with the address of this RCW. This completes the linkage back to the Main Program level. This Operand Call also addresses the *c* and *l* registers to the subroutine and places the Processor in Sublevel.# The Stack at stage 3 shows the addition of the RCW.

The Processor now encounters the syllables in the subroutine, the first three of which comprise a program to compute the value for X. Stage 4 shows the use of additional area in the Stack for this computation.

The subroutine is terminated with a Return Operator that accesses the RCW and MSCW and readdresses the Processor to the Main Program. Stage 5 shows the *s* register pointing to the original content of the top of the Stack. The result of the subroutine has been passed back to the Main Program and is now in the *a* register. The Processor registers (H, V, L, G, K, F, C, R) and flip-flops (msff, salf) have been restored from the content of the control words.

# Sublevel and Program Level are two conditions under which the Processor operates. They are described in [5].

```
CALL ON PROCEDURE FACT :
        X ← FACT (3) ;

PROCEDURE FACT :
        REAL PROCEDURE FACT  (N) ;
        IF  N = 1  THEN  FACT ← 1
     ELSE FACT ← N  X  FACT  (N−1) ;
```

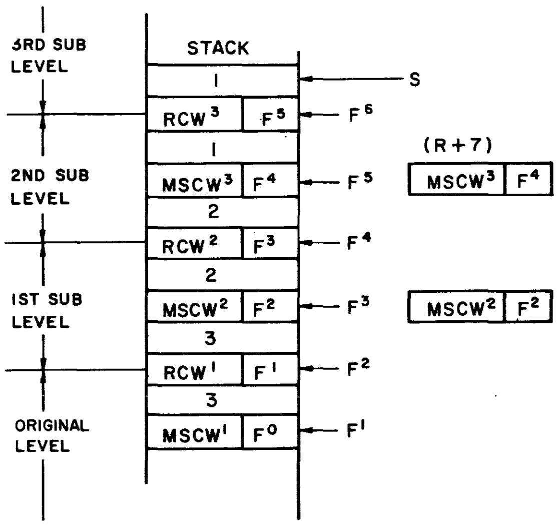


Figure 4. Call on Recursive Procedure "FACT."

Application of the Stack to Recursive Procedures

The ability to enter a subroutine is recursive and it is essential that the control linkage from one level to the next be preserved.

The following example demonstrates how the stack is used to retain this linkage for subroutines of unlimited depth.

Consider an ALGOL Program that calls on a Procedure "FACT":

X ← FACT (3);

Let that procedure call upon itself within the Procedure Body:

```
REAL PROCEDURE FACT (N);
IF N = 1 THEN FACT ← 1
ELSE FACT ← Nx FACT (N — 1);
```

Figure 4 shows the Stack at the limit of recursion with the control words and parameters that it contains. The *f* register is set to the address of each control word as it is placed in the Stack and the sequential values are indicated with a superscripted F^X. Stored in each control word is the content of the *f* register at the time the control word was built. This value of F^X is the address of the previous control word. Thus there is stored in the Stack a full linkage from the latest address in the *f* register back through the control words to the original program level.

Following each MSCW in this Stack, is a Formal Parameter that is to be passed to the subroutine. The Procedure requires this parameter twice during each recursion. First, it is required to derive the value N, at which time it is accessed with an *f* relative address (f — 1). The second time it is required to compute (N — 1). By this time the *f* register has been set to a new value and access to N via the Control Word Linkage could become complex. This search can be avoided if the settings of F^2 (and later f^4) are preserved in the PRT at (r + 7). Operator logic is provided to automatically access the "old" value of F in (r + 7) when these Global Parameters are required. This use of the content of (r + 7) as a base for relative addressing is simply an extension of *f* relative addressing.

The return to the original program is performed sequentially with a return to the previous level, multiply (N — 1) × N and return to the next higher level, etc., until the original level is reached.

This elementary program illustrates how the Control Words, which are built and used by the hardware, automatically provide address links for any depth of subroutine. Also, the ability to address words (f —) relative and (f +) relative provides access to data that is below the current top of the Stack, allowing parameters to be passed forward from one level to the next.

CONCLUSION

It has not been possible to describe all the ramifications of the Stack and its use by the software. The applications given serve to illustrate how the mechanization of a push-down stack was organized in conjunction with the software design to automate many recurring

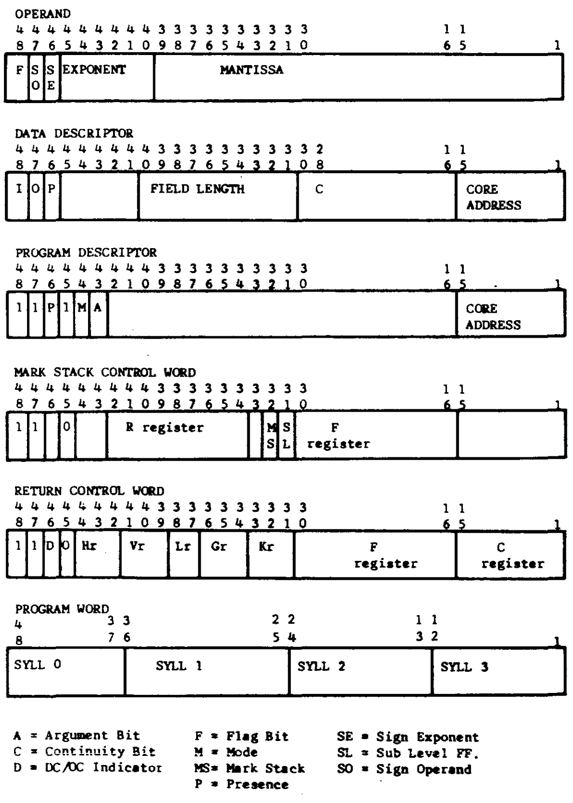


Figure 5. Word Formats.

Symbol	ν	μ	Description
$\underline{a}$	48		A register Top position of Stack
$\underline{b}$	48		B register Second position of Stack
$\underline{p}$	48		P register Holds Program word
$\underline{t}$	12		T register Holds Program syllable
$\underline{c}$	15		Program word address register
$\underline{l}$	2		L register Syllable pointer
$\underline{j}$	4		J register Sequence count
$\underline{r}$	9		R register Base of PRT
$\underline{s}$	15		S register Top of Memory Stack
$\underline{f}$	15		F register Subroutine Stack pointer
$\underset{\smile}{arof}$	1		A register Occupancy flip-flop
$\underset{\smile}{brof}$	1		B register Occupancy flip-flop
$\underset{\smile}{msff}$	1		Mark Stack flip-flop
$\underset{\smile}{salf}$	1		Sub-level flip-flop
$\underline{M}$	48	32768	Core Memory Matrix (Memory Unit)

NOTE: This list does not comprise all the registers in the Processor.

Figure 6. Register and Symbols.

operations. The resulting powerful machine language operators have reduced the coding required to produce efficient compilers and object programs.

There is an opportunity for progress in the design of general purpose computers if consideration is given to the balance between the hardware logic and the logic performed by program. In the machine described, the automatic hardware stack has relieved the software of several arduous tasks and the cost in added components has been relatively small.

BIBLIOGRAPHY

1. IVERSON, K. E.: A Programming Language, John Wiley & Sons, Inc., New York, New York, 1962.
2. HAMBLIN, C. L.: Translation to and from Polish Notation, The Computer Journal, October, 1962.
3. BARTON, R. S.: A New Approach to the Functional Design of a Digital Computer, Proceedings of the Western Joint Computer Conference, May, 1961, pgs. 393-396.
4. MCCRACKEN, DANIEL D.: A Guide to ALGOL Programming: John Wiley & Sons, Inc., New York, New York, 1962.
5. The Operational Characteristics of the Processor for the B5000, Burroughs Corporation, November, 1961.

EFFECTS OF DIGITAL EXECUTION TIME IN A HYBRID COMPUTER

Takeo Miura
280 Mitachi Central Research Lab.
Kokubunji, Tokyo, Japan
and
Junzo Iwata
Hitachi Eectronics Co.
Miyukicho Odaita, Tokyo, Japan

1. INTRODUCTION

Recently, for the purpose of improving the computing accuracy of an analog computer as well as assuring the stability of solution, hybrid computing techniques have been developed rapidly in the form of applying digital computing techniques to relatively low accuracy computing elements such as nonlinear element or in the form of combining the conventional analog computing equipment in the simulator, etc. with a digital computer with the aim of realizing low cost, simplified equipment and easy updating. Two problems pointed out on such a bybrid computer are computation assignment between digital and analog parts and effects of digital execution time. In the parallel hybrid computing system in which the above-mentioned digital part operates in parallel with the analog computer, essential time delay required for the digital computation is of prime importance and, in many cases, the computation assignment can not be determined without considering the effects of it. For instance, if the amount of computation on the digital part is increased in order to raise accuracy, execution time is prolonged and so, increased computing error may result.

This paper describes theoretical and experimental investigations on these problems. Effects of digital execution time are derived in the form of general formulas, their application to several examples including setup for generating sinusoidal wave are considered, and a compensating system for the effects is proposed with its usefulness clearly shown.

2. TRANSFER FUNCTION OF DIGITAL COMPUTING PART

In a parallel hybrid computing system, time varying output of the analog computer is sampled, A–D converted and applied to the digital computer, and the result of computation is furnished as output after D–A conversion. This output, when applied to an analog computer, is held usually until it is updated by the result of the next new digital computation. Therefore, the input to the analog computer has a step wave form varying in steps every sample period. This is shown in Fig. 1. In (a), sampling period T is infinitely small and digital computing time τ (including the time for A–D conversion and D–A conversion) is zero, representing an ideal case. In (b), sampling period T is infinitely small and only the time required for digital computation, τ, is considered, the wave form being delayed by τ with respect to the ideal case. In (c), the effect of sample hold is also considered, and the wave form varying every sampling period T is obtained. Let the contents of computation on the digital computer

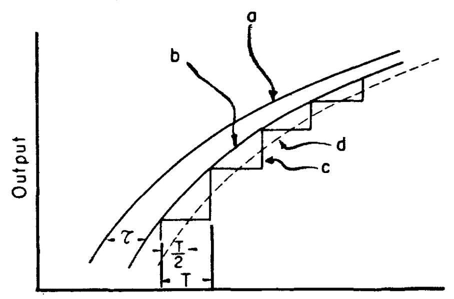


Figure 1

be denoted by A and obtain transfer function for three cases mentioned above. For (a), it is just A. For (b), it is $Ae^{-\tau p}$. For (c), it is obtained by first multiplying the wave form of (b) by the pulse train $(1 + e^{-TP} + e^{-2TP} + \dots)$ and then passing the product through the transfer function of the sample-hold circuit, $\frac{1 - e^{-TP}}{P}$. The calculation is facilitated by z transformation. That is, the transfer function (z transformation) of the circuits up to directly before the hold circuit is $Az^{-\frac{\tau}{T}}$, and the computing circuits subsequent to the digital circuit may be applied with Z transformation with the hold circuit added to the input.

Since the A denotes the contents of computation on the digital computer, computations involved are generally arithmetical operations, and when the representation of transfer function A is used, it is considered that any operation can be done instantaneously. If the past data is also used, a transfer function corresponding to it is obtained. For instance, in performing integration by the Euler's formula, product of sampling period and input is added to the present value every sampling. In this case,

$$A = \frac{TZ}{Z - 1} \tag{1}$$

When such a Z transformation is used in calculating the effect of hybrid computing system, the transformation should be done including analog computing circuits which are connected to the sample-hold circuit, and calculating on each computing circuit is complicated. However, if the computing circuit subsequent to the sample-hold is not sensitive to high frequency, the step wave form in the case of sample-hold shown in Fig. 1 (c) is equivalent to the smoothed wave form shown in (d). Usually, the sampling period is so selected as to prevent appearance of step wave form in the solution, the assumption given above is approximately true. This smoothed curve may be considered by delaying the original wave form (b) in the case of sample-hold by T/2 if the period of oscillation of input is sufficiently large compared with the sampling period as shown in the figure. Although reasoned intuitively from the figure, this can also be proven as follows and the error involved can be evaluated correctly.

The sample-hold circuit can be applied with the principle of superposition as to input and output. Thus, effect of sample-hold on any input can be known if the input is divided into frequency components and the effect of sample-hold is considered on them. Assume the input wave form to be $\sin(\alpha t + \beta)$ and denote the wave form resulting from its sample holding at period T as shown in Fig. 2 by F(t). Then, F(t) contains higher harmonics which are produced because of distortion of wave form by sample-hold operation, but, it can be calculated by developing the output wave form into Fourier series. (Refer to Appendix.) According to the result, frequencies of the higher harmonics appear only discretely and the lowest frequency is equal to the input frequency and, if denoted by $F_b(t)$, can be expressed as

$$F_b(t) = \frac{2}{\alpha T} \sin \frac{\alpha T}{2} \sin \left\{ \alpha \left(t - \frac{T}{2} \right) + \beta \right\} \tag{2}$$

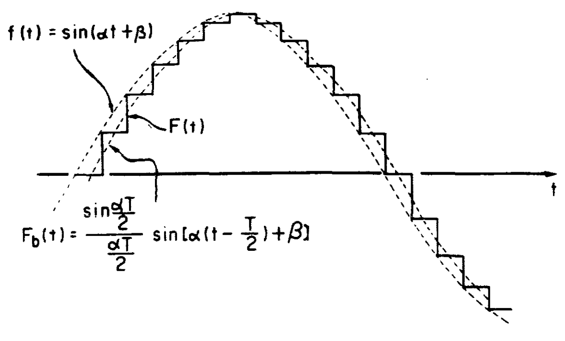

f(t) = sin(αt + β)
F(t)
t
$F_b(t) = \frac{\sin\frac{\alpha T}{2}}{\frac{\alpha T}{2}} \sin[\alpha(t - \frac{T}{2}) + \beta]$

Figure 2

TABLE 1. Amplitude of Higher Harmonics Components of Sample and Zero-order Hold Waveform. (m: number of sample per period)

m	4	10	15	20	30	50	100
$\dfrac{\sin \frac{\alpha T}{2}}{\frac{\alpha}{2} T}$	0.9003	0.9836	0.9927	0.9959	0.9982	0.9993	0.9998
$\dfrac{\sin \frac{\alpha}{2} T}{\pi - \frac{\alpha}{2} T}$	0.30	0.11	0.071	0.052	0.034	0.020	0.0095
$\dfrac{\sin \frac{\alpha}{2} T}{\pi + \frac{\alpha}{2} T}$	0.18	0.089	0.062	0.047	0.032	0.020	0.0093

The next higher harmonic has the period approximately equal to the sampling period T and, if denoted by $F_{S1}(t)$, can be expressed as

$$F_{S1}(t) = \Sigma \frac{2 \sin (\alpha T/2)}{\alpha T \pm 2\pi} \sin \left\{ \left(\alpha \pm \frac{2\pi}{T} \right) t - \frac{\alpha T}{2} + \beta \right\} \quad (3)$$

(Σ denotes summation with respect to each sign)

The subsequent components have the frequencies approximately equal to multiples of the above and are denoted by $F_{S2}(t)$, $F_{S3}(t)$, etc. (Refer to Appendix.) Amplitude of $F_b(t)$ is $\{\sin (\alpha T/2)\}/(\alpha T/2)$ of that of input and approaches unity as T is reduced. $\alpha T/2$ is proportional to the reciprocal of sample number per period, $m = 2\pi/\alpha T$, and the amplitude error is determined if the sample number per period is given. Table 1 shows the amplitude error. Difference in phase from input, which is $\alpha T/2$, indicates that time delay of T/2 is caused irrespective of frequency, proving the abovementioned reasoning.

Amplitude of the component having the frequency approximately equal to the sampling period, is $2\{\sin(\alpha T/2)\}(\alpha T \pm 2\pi)$ and approaches zero as T becomes small. Table 1 shows magnitude of it. In respect that the amplitude of this oscillation is small as compared with that of input and is considered further to be filtered in the subsequent computing circuits (for instance, an integrator as subsequent computing circuit reduces the gain down to approximately 1/m of that of oscillation of frequency α), any frequency other than input frequency will not be considered. Moreover, if T is small enough to assume $\{\sin(\alpha T/2)\}/(\alpha T/2)$ to be equal to unity, the transfer function of digital computer part, D(p), will be as follows.

$$D(p) = Ae^{-\left(\tau + \frac{T}{2}\right)p} \quad (4)$$

The error resulting from the consideration of such a transfer function is determined by the subsequent computing circuit and the magnitude is very small as shown in the example which appears later.

3. EFFECTS OF DIGITAL EXECUTION TIME ON COMPUTING ERROR

(3.1) *General Formula for Error*

Previously, the authors considered the computing error involved in solving linear differential equations or plural simultaneous differential equations by the use of an analog computer.(1) This technique can also be applied in estimating the effect of digital execution time when the same equations are solved by the hybrid computing system.

In general, a differential equation can be transformed into 1st order simultaneous differential equations. In the case of linear, equation is shown below

$$\left.\begin{aligned} &\frac{dy_1}{dt} + k_{11}y_1 + k_{12}y_2 + \cdots + k_{1n}y_n = k_{10} \\ &\frac{dy_2}{dt} + k_{21}y_1 + k_{22}y_2 + \cdots + k_{2n}y_n = k_{20} \\ &\cdots\cdots \\ &\frac{dy_n}{dt} + k_{n1}y_1 + k_{n2}y_2 + \cdots + k_{nn}y_n = k_{n0} \end{aligned}\right\} \quad (5)$$

The characteristic equation is as follows.

$$\begin{vmatrix} p + k_{11} & k_{12} & \cdots & k_{12} \\ k_{21} & p + k_{22} & \cdots & k_{2n} \\ \cdots\cdots & & & \\ k_{n1} & k_{n2} & \cdots & p + k_{nn} \end{vmatrix} = 0 \quad (6)$$

The computing circuit for Eq. (5) is shown in Fig. 3. With reference to the figure, denote by —I(p) and $K_{ij}(p)$ the transfer function of the integrator and that of the coefficient setting circuit for computation of coefficient k_{ij}, respectively. Then, the characteristic equation for the equations being solved in the circuit of Fig. 3 is

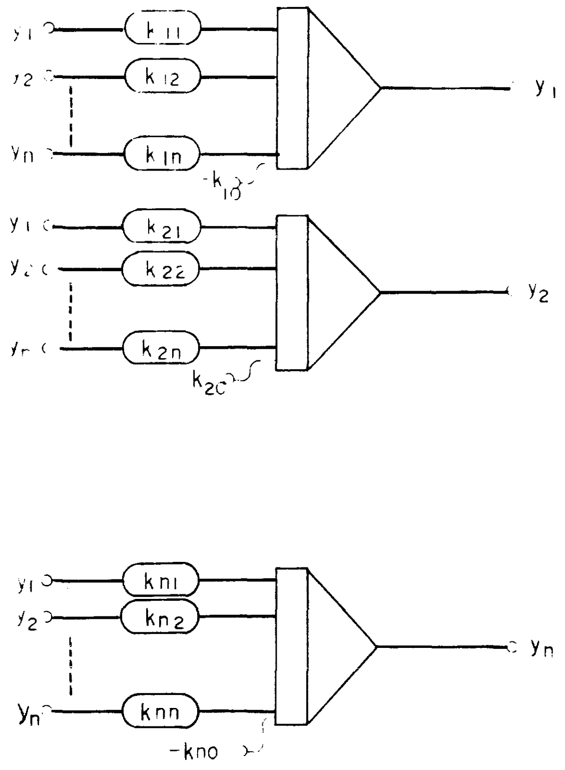


Figure 3

$$F(p) = \begin{vmatrix} I(p)^{-1} + K_{11}(p) & K_{12}(p) & \cdots K_{n1}(p) \\ K_{12}(p) & I(p)^{-1} + K_{22}(p) & \cdots K_{n2}(p) \\ \cdots\cdots & & \\ K_{1n}(p) & K_{2n}(p) & \cdots I(p)^{-1} + K_n(p) \end{vmatrix} = 0 \quad (7)$$

If there is no effect of digital execution time and $I(p)^{-1} = p$ and $K_{ij}(p) = k_{ij}$, Eq. (7) and Eq. (6) agree with each other.

Denote by p_a' one of characteristic roots of Eq. (7) and by P_a the corresponding root of Eq. (6). Then, error ϵ as the difference between the two roots is given, as already shown,(1) by

$$\epsilon = -F(p_a) \Big/ \left[\frac{\alpha F(p)}{\alpha p}\right]_{p = p_a} \quad (8)$$

Substituting Eq. (7) into Eq. (8) gives general formula for error. Let $I(p)^{-1} = p\,(1 + \epsilon_I(p))$ and $K_{ij}(p) = k_{ij}\,(1 + \epsilon_{ij}(p))$ where $\epsilon_I(p)$ and $\epsilon_{ij}(p)$ are error terms of the respective circuits.

By the first approximation we have

$$\epsilon \simeq \left[\underset{i=1}{\overset{n}{-\sum}} \left| \begin{array}{cccc} p+k_{11} & k_{12} & \cdots\cdot\, k_{1i}\epsilon_{1i}(p) & \cdots\cdot\, k_{1n} \\ k_{12} & p+k_{22} & \cdots\cdot\, k_{2i}\epsilon_{2i}(p) & \cdots\cdot\, k_{2n} \\ \cdots\cdot & \cdots\cdot & \cdots\cdot & \\ k_{i1} & k_{i2} & \cdots\cdot\, p\epsilon_I(p) + k_{ii}\,\epsilon_{ii}(p) & \cdots\cdot\, k_{in} \\ \cdots\cdot & \cdots\cdot\ \ \cdots\cdot & & \cdots\cdot \\ k_{n1} & k_{n2} & \cdots\cdot\, k_{ni}\epsilon_{ni}(p) & p+k_{nn} \end{array} \right| \right]_{p=P_a} \div$$

$$\left[\sum_{i=1}^{n} \left| \begin{array}{cccc} p+k_{11} & k_{12} & \cdots\cdot\, k_{1i}\dfrac{\alpha\epsilon_{1i}}{\alpha p} & \cdots\cdot\, k_{1n} \\ k_{21} & p+k_{22}\ k_{2i} & \cdots\cdot\, \dfrac{\alpha\epsilon_{2i}}{\alpha p} & \cdots\cdot\, k_{2n} \\ \cdots\cdot & \cdots\cdot & \cdots\cdot & \cdots\cdot \\ k_{i1} & k_{i2} & \cdots\cdot\, 1 + \dfrac{\alpha(p\epsilon_I\,(p) + k_{ii}\epsilon_{ii})}{\alpha p} & \cdots\cdot\, k_{in} \\ \cdots\cdot & \cdots\cdot & \cdots\cdot & \\ k_{n1} & k_{n2} & \cdots\cdot\, k_{ni}\dfrac{\alpha\epsilon_{ni}}{\alpha p} & \cdots\cdot\, p+k_{nn} \end{array} \right| \right]_{p=P_a} \qquad (9)$$

Eq. (9) is the general formula regarding the error. By using in $\epsilon_I(p)$ and $\epsilon_{ij}(p)$ the result obtained in Section 2 and substituting the characteristic root determined for the given differential equations, we can easily estimate the effect of digital execution time on the solution.

(3.2) *Error in Generation of Sinusoidal Wave (Circle Test)*

To verify the general formula in (3.1) experimentally, the following example of circle test is presented.

$$\ddot{y} + \omega^2 y = 0 \qquad (10)$$

The hybrid computing system cenceivable for this case is illustrated in Fig. 4. (a) and (d) show the main computing circuits and (b) and (c) the minor computing circuits, respectively inserted with digital computing part. In (b), (c) and (d), positive feedback path for digital part and negative feedback path for analog part, or conversely, negative feedback path for digital part and positive feedback path for analog part, are so arranged as to cancel each other. In the following, effects of digital computation for the cases shown in Fig. 4 are examined using Eq. (9). It is assumed that the digital computing part gives only effects of data sample and execution time τ. From the description given in Section 2, we have

$$\epsilon_{ij} = e^{-\left(\tau+\frac{T}{2}\right)p} \cong e_{-T_a p} - 1 \qquad (11)$$

$$\epsilon_I = 0$$

where T is the sampling period, τ the digital execution time and $T_d = \tau + T/2$ the equivalent dead time, and the error of computation in each case is as follows.

First, in the computing system of Fig. 4(a), placing $-\dot{y} = y_1$ and $\omega y = y_2$ in Eq. (10) gives $k_{11} = 0$, $k_{12} = -\omega$, $k_{21} = \omega$ and $k_{22} = 0$. It is obvious from Fig. 5(a) that $K_{12}(p) = -\omega e^{-T_a p}$. Thus, from Eq. (9),

$$\epsilon \simeq \left(\frac{-\begin{vmatrix} p & -\omega(-T_a p + T_a^2 p^2/2) \\ \omega & 0 \end{vmatrix}}{\begin{vmatrix} 1 & -\omega e^{-T_a p} \\ 0 & p \end{vmatrix} + \begin{vmatrix} p & \omega T_a e^{-T_a p} \\ \omega & 1 \end{vmatrix}} \right)_{p = jw} \tag{12}$$

$$\simeq \frac{\omega(2\omega T_a + \omega^3 T_a^3/2)}{4 + 5\omega^2 T_a^2} - j \frac{\omega(2\omega^2 T_a^2)}{4 + 5\omega^2 T_a^2}$$

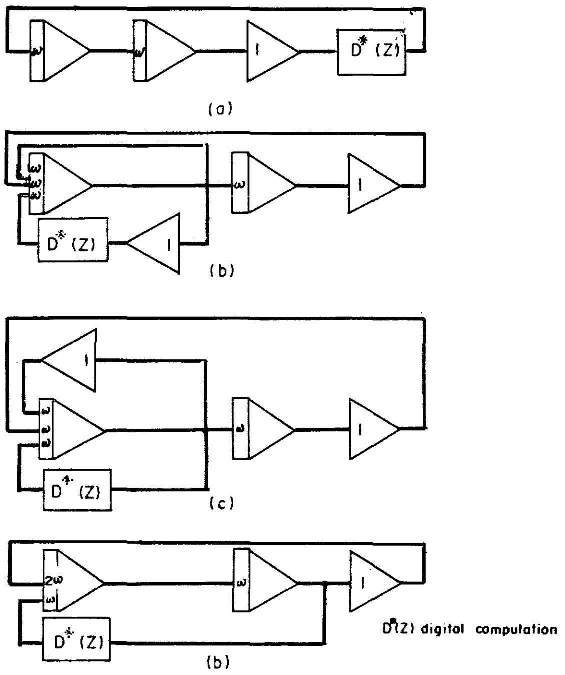


Figure 4

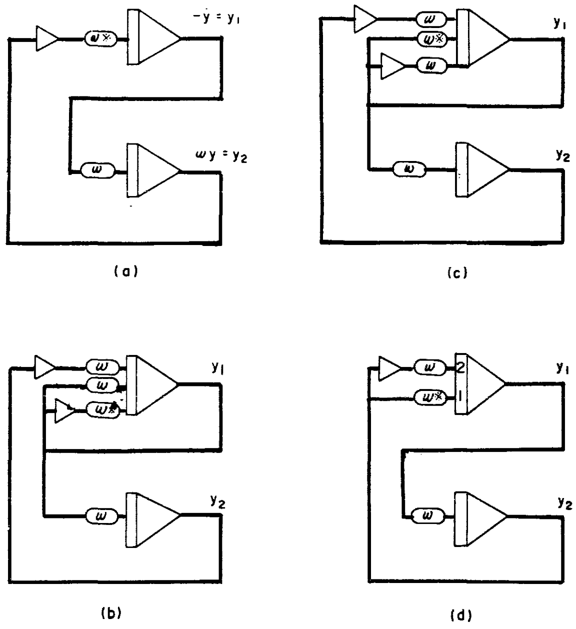


Figure 5

The first and second terms of Eq. (12) are amplitude and frequency error terms, respectively.

For the case of Fig. (b) and (c), reference to Eq. (9) and Fig. 5(b) and (c) gives

$$\epsilon \simeq \frac{\omega(\pm\omega^2 T_a^2 + \omega^3 T_a^3)}{4 + \omega^4 T_a^4} + j \frac{\omega(\pm\omega T_a - \omega^4 T_a^4/2)}{4 + \omega^4 T_a^4} \tag{13}$$

Similarly, for the case of Fig. 4(d), reference to Eq. (9) and Fig. 5(d) gives

$$\epsilon \simeq \frac{-\omega\left(2\omega T_a + \frac{11}{6}\omega^3 T_a^3\right)}{4 + 5\omega^3 T_a^2} - j \frac{\omega^2 T_a/3}{4 + 5\omega^3 T_a^2} \tag{14}$$

The amplitude error calculated from Eqs. (12) through (14) with the relation of Eq. (9) inserted in T_d is shown in Fig. 6. From this figure, it is obvious that the error for (a) and (d), with the digital part contained in the main computing circuit, and that for (b) and (c), with the digital part contained in the minor computing circuit, are approximately equal in magnitude, but, opposite in the polarity of divergence and convergence. The effect of T_d is more conspicuous where the digital part is contained in the main computing circuit, and it is significant to make assignment of computation in such a way, if possible, that in the main computing circuit analog computation is carried out by the use of, for instance, digital potentiometer or the like which is constructed to apply variable in place of standard voltage of D–A converter. It may be said generally that the minor computing circuit contains one or less integrator in its computation loop, while the main computing circuit contains two or

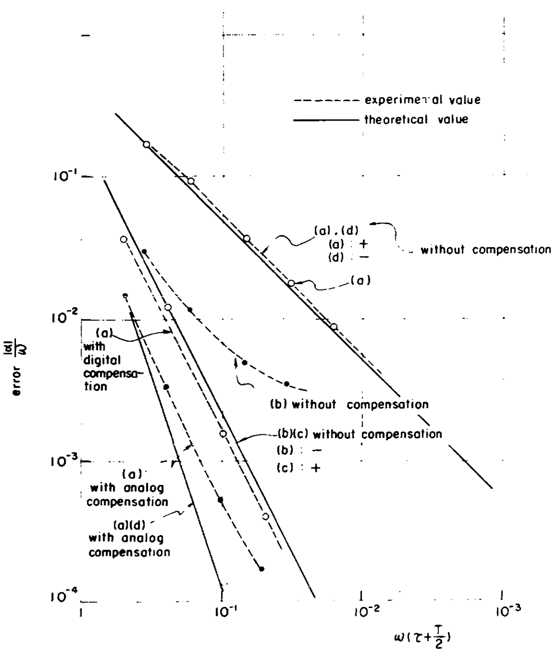


Figure 6

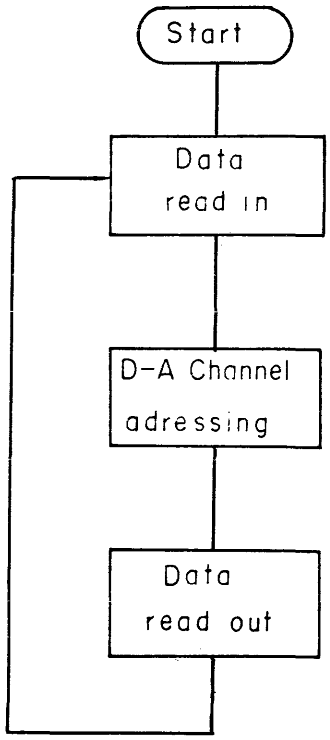


Figure 7

more integrators. In an inevitable case which requires digital part in the main computing circuit, it is advantageous to apply the compensating method which appears later.

Fig. 6 shows the experimental results obtained with the hybrid computer. The flow chart for the digital computer is shown in Fig. 7 and comprises memory of A–D converted data and repetition of D–A conversion only. In the circuits of Fig. 4(b) and (c), ϵ does not decrease with decreasing T, because its cancellation is impossible due to A–D conversion error, etc., even under the ideal signal conditions of positive and negative feedback circuits. In the experiment, τ was taken as approximately 90 ms. If the computing circuits subsequent to the sample-hold are determined, calculating of computation error may be carried out also by applying z transformation inclusive of the circuits as mentioned before. Calculation of ϵ using the z transformation is presented for the case of Fig. 4(a). The characteristic equation obtained from Fig. 4(a), as follows.

$$1 + z\frac{1}{p^2}H(p)e^{-p\tau} = 0 \qquad H(p) = (1 - e^{-pT})/p \tag{15}$$

$$\text{If } \tau = T \quad 2Z^3 - 4Z^2 + Z(2 + T^2) + T^2 = 0 \tag{16}$$

$$\epsilon \simeq \frac{3}{4}T\left(\frac{1 + \frac{11}{36}T^4}{1 + \frac{3}{2}T^2 + \frac{17}{48}T^4}\right) \simeq \frac{3}{4}T \tag{17}$$

On the other hand, placing $\tau = T$ in Eq. (12) gives $T_d = 3/2\ T$ and $\epsilon \simeq \frac{3}{4}T$. This shows complete agreement with Eq. (7).

4. COMPENSATION FOR THE ERROR DUE TO DIGITAL EXECUTION TIME

The execution time of digital part is, in some cases, a cause of large error in a hybrid computing system and the value of the error can be predicted from Eq. (9) as mentioned before. Thus, the sampling period required for the error to be held within permissible limits in accordance with Eq. (9) can be determined. However, the value thus found is not always satisfied easily by the digital computers in current use. The technique described in this chapter is to compensate for the delay of execution time due to digital computer and improve the computing accuracy markedly. Conversely, for a given degree of computing accuracy, the sampling period can be increased by this technique owing to the effect of compensation, effectively for the reducing of burden which is imposed on the digital computing speed.

(4.1) *Analog Compensation*

The delay, $\tau + T/2$, is expressed as

$$e^{-\left(\tau+\frac{T}{2}\right)p} = 1 - \left(\tau + \frac{T}{2}\right)p + \frac{1}{2!}\left(\tau+\frac{T}{2}\right)^2 p^2 \cdots \tag{18}$$

The second term and following produce the error in comparison with the case of no delay. To remove the effect of the second term, which is the largest error term, result of operation $(\tau + T/2)p$ is added to the above.

$$e^{-\left(\tau+\frac{T}{2}\right)p}\left\{1+\left(\tau+\frac{T}{2}\right)p\right\} = 1 - \left(1-\frac{1}{2!}\right)\left(\tau+\frac{T}{2}\right)^2 p^2 + \left(\frac{1}{2!}-\frac{1}{3!}\right)\left(\tau+\frac{T}{2}\right)^3 p^3 \cdots\cdots \tag{19}$$

Thus, the second term can be eliminated. In this technique, operation $(\tau + T/2)p$ is carried out on an analog computer and the result is added to digital computer output.

The operation $(\tau + T/2)p$ is a differentiating operation. Since the output of the digital computer is a step wave, it is not advisable to attempt its actual differentiation. However, if this output is input to the integrator as shown in Fig. 8(a), the delay can easily be compensated for by adding the result of digital computation multiplied by $K(\tau + T/2)$ to the integrator output as shown in Fig. 8(b). This is obvious from the fact that the integration following a differentiation results in the original signal. The (b) can be simplified as shown in Fig. 8(c). In this figure, operational amplifier of integrator is used also as that of sign changer in (b), and it is only necessary to insert capacitor in parallel with input resistance as shown in the figure. If the capacitor used has the capacity of $K(\tau + T/2)C$, potentiometer need not be used. Fig. 9 is an elementary illustration which shows the compensation for dead time by this method. It clearly shows that, by adding the input multiplied by $k(\tau + T/2)$ to the integrator output, time advance of $\tau + T/2$ or so is produced. In general, the digital computing part performs nonlinear computation and the result is fed to the integrator through a linear computing part. Even if the linear computing part is applied

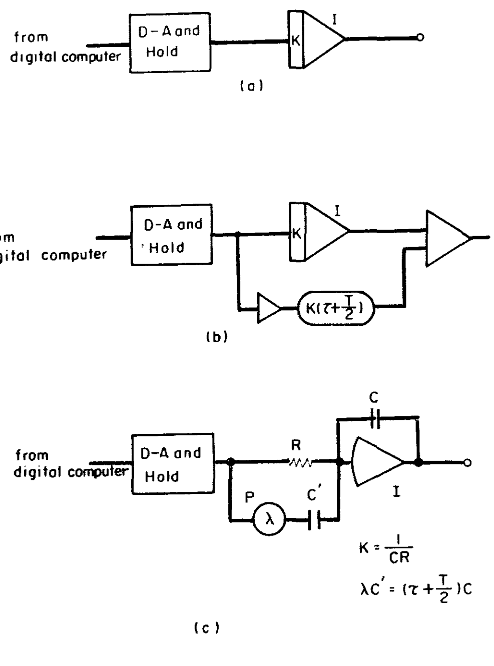


Figure 8

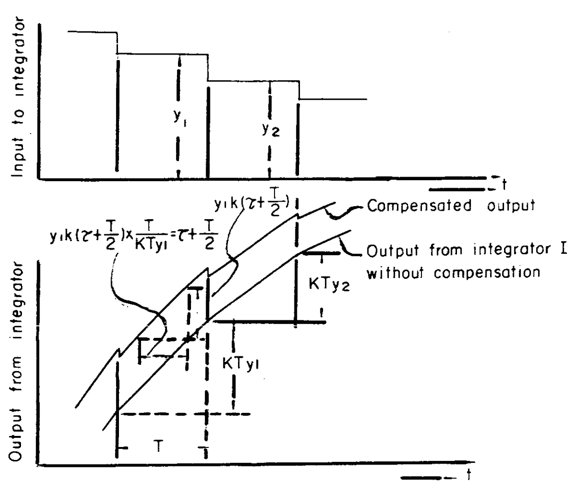
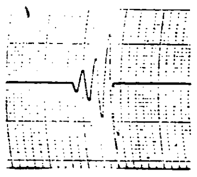
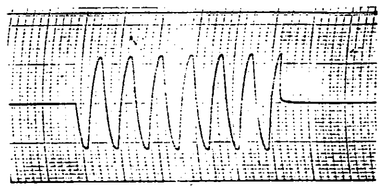


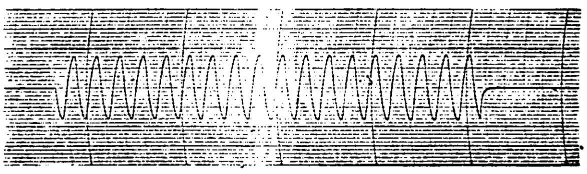
Figure 9



(a) Not Compensated.



(b) Compensated by Digital Computer.



(c) Compensated by Analog Computer.

Figure 10

with other input, that part is concerned being separated from because the linear computation, and the above-mentioned compensation is available.

Fig. 10 shows the experimental results. In the case of Fig. 4(a) digital computation is included in the main computing circuit for the circle test mentioned in the preceding chapter: (a) with no compensation and (c) with the said compensation. Both Fig. 8(b) and Fig. 8(c) give quite the same results.

Fig. 6 shows the calculated and measured error for the circle test with the compensation applied. With reference to Fig. 4(a), calculating the theoretical value corresponds to solving on the computing circuit shown in Fig. 11, and the associated error is given, from Eq. (9) as follows.

$$\epsilon \simeq \left(\frac{\begin{vmatrix} p & \omega(1-e^{-T\alpha p}) \\ \omega & \omega Te^{-T\alpha p} \end{vmatrix}}{\begin{vmatrix} 1 & -\omega e^{-T\alpha p} \\ 0 & p+\omega T\alpha e^{-T\alpha p} \end{vmatrix} + \begin{vmatrix} p & \omega T\alpha e^{-T\alpha p} \\ \omega & 1-\omega T\alpha^2 e^{-T\alpha p} \end{vmatrix}} \right)_{p=j\omega} \tag{20}$$

Error for the case of (b), (c) or (d) can be calculated similarly. From the figure, it is known that, in the cases of (a) and (d), where digital computation is contained in the main computing circuit, compensation is very effective for improving error of real part of the characteristic root, or, divergent or convergent error. In the cases of (b) and (c), compensation is not so effective for improvement of real part error, and the error is unchanged from that shown in the graph of Fig. 6 for no compensation.

(4.2) *Digital Compensation*

As mentioned previously, analog differentiation of D–A converted result does not give satisfactory result, but, it is relatively easy to obtain the differential on the digital computer. The differential of y is obtained approximately as $\frac{1}{T}(y_0 - y_{-1})$ where T is the sampling period, y_0 is the calculated value of y at the present time and y_{-1} is that at the previous time, and then, the differential $\frac{1}{T}(y_0 - y_{-1})$ multiplied by (τ + T/2) is added to y_0 to predict the value in advance by τ + T/2. That is, following the calculation of y_0, it is necessary to make the calculation shown below on the digital computer.

$$y = y_0 + \left(\tau + \frac{T}{2}\right)\frac{1}{T}\left(y_0 - y_1\right) = y_0 + \left(\frac{1}{2} + \frac{\tau}{T}\right)\left(y_0 - y_1\right) \tag{21}$$

If the value of y thus obtained is D–A converted and produced as output, the output will be in advance by π + T/2 as compared with mere output of y_0 and compensation in thereby accomplished. The experimental result for this digital compensation are as digital compensation of Fig. 10(b).

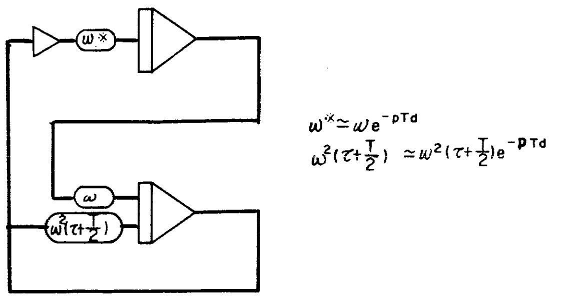

ω^{*}
ω
$\omega^2(\tau+\frac{T}{2})$
$\omega^{*} \simeq \omega e^{-pTd}$
$\omega^2(\tau+\frac{T}{2}) \simeq \omega^2(\tau+\frac{T}{2})e^{-pTd}$

Figure 11

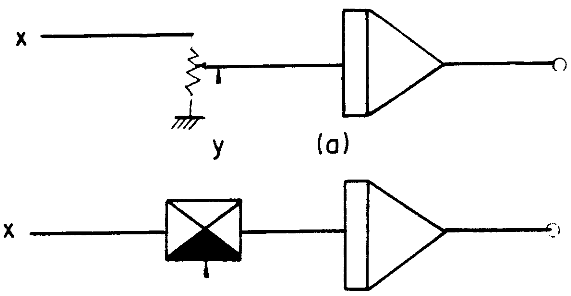


Figure 12

Application of this digital compensation are as follows. Sometimes, the result of digital computation, y, is used for the setting of potentiometer and it is multiplied by another input x as shown in Fig. 12(b) in the figure shows symbol of the potentiometer which is set by digital output. The advantage of such an arrangement is that, even if x varies so fast that the computation of xy carried out on the digital basis would require very fine sampling period, this arrangement, which carries out the multiplication on the analog basis, requires only such a magnitude of sample period that corresponds to the varying rate of y and makes the requirement for the digital computing speed moderate.

To compensate in this case for the first term of the delay time, it is necessary to add Tx $\cdot \frac{dy}{dt}$ However, since x is varying fast and its integral is not necessarily Txy, the method mentioned in the preceding section is not applicable, but, digital compensation should be used.

(4.3) *Compensation by Continuous Prediction Circuit*

In some cases, the step wave form of the digital computer output is an error source and changing it into a continuous wave form improves the accuracy or lengthens the sampling period advantageously.

Essentially, the digital computer output is intermittently obtainable: the sample and hold technique is only a means for converting it into analog input. A computing circuit which makes interpolation on the values intermittently obtained and gives a time advance of the execution time was previously reported as the prediction circuit. By the use of such a circuit, it is possible to make the digital computer output to approach the wave form of ideal case. However, too precise approximation results in unnecessarily complicated circuits or computing steps and may increase error at the time the computational solution changes in an unexpected manner. Thus, the prediction circuit of first approximation or so should be practical.

In this circuit, nearest two sampled values in the past are obtained and future value is predicted by a straight line which connects these two points. Fig. 13 shows an example of the

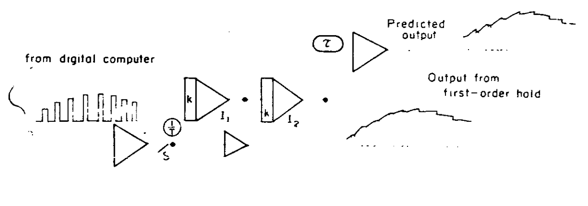


Figure 13

prediction circuit of first approximation.

When switch S is open, integrator I_1 receives no input and holds constant output, while integrator I_2 varies linearly with a gradient proportional to the hold output of I_1. If S is closed only during the period between t_2 and t_3 shown in Fig. 14, during which input to this channel

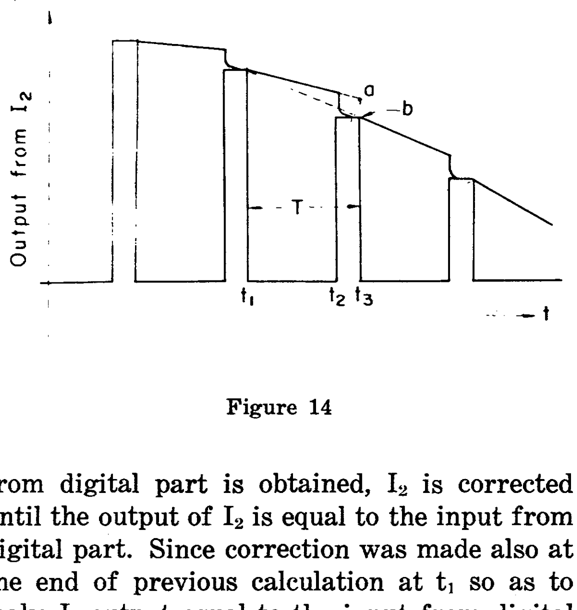


Figure 14

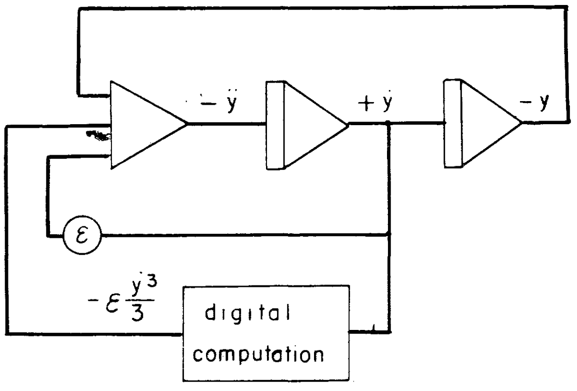
from digital part is obtained, I_2 is corrected until the output of I_2 is equal to the input from digital part. Since correction was made also at the end of previous calculation at t_1 so as to make I_2 output equal to the input from digital part, correction by $\overline{ab}$ of I_2 at present time is indicative that average gradient of change between the value of input from digital part at previous time and that at present time is different by $\overline{ab}/T$ from the hold output of I_1. Thus, if $1/T$ of I_2 input has been applied to I_1 as shown in the figure, hold output of I_1 is corrected by $\overline{ab}/T$ while I_2 is being corrected by ab and thereafter, output of I_2 varies with a gradient equal to average gradient of digital output between the previous and the present time. In this way, a wave form (1st-order hold wave form) connecting nearest two values of digital output and extending on the straight line is obtained as output of I_2. This first-order hold wave form is considered to be free from delay of $T/2$ due to zero-order hold, and in this case, compensation for digital execution time τ is sufficient. Since output of I_1 is differential of I_2, it is possible, by adding τ-multiple of it to I_2 output, to predict the value in advance by τ and compensation can be realized.

5. EXAMPLE OF ANALYSIS OF COMPUTATION OF NONLINEAR DIFFERENTIAL EQUATION

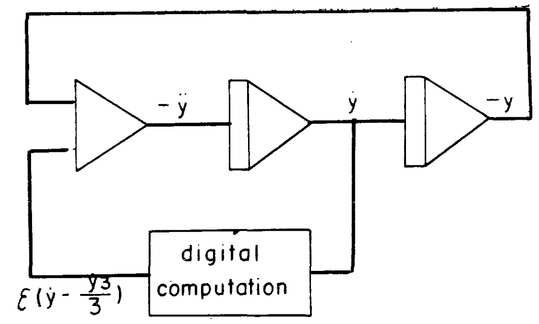
One of the principal purposes of the parallel hybrid computing system is in achieving such results as improved precision, reduction in cost, ease of handling, etc., by using the digital computer for nonlinear computation. As to nonlinear computation, however, the solution of computation itself can hardly be obtained theoretically, and the calculation of error term depending on the solution of computation is not easy. In this section, to clarify experimentally effects of the digital computation on nonlinear conputation, investigation is made for the case of computing the Van der Pol's equation

$$\ddot{y} - \epsilon\left(\dot{y} - \frac{\dot{y}^3}{3}\right) + y = 0 \qquad (22)$$

by the hybrid computing system. Generally, in solving (22) by the use of analog computer, the part which involves the lowest precision is the computing circuit which generates the 2nd term. Consequently, it is readily suggested in employing the hybrid system to make the digital part take charge of the generation of the 2nd term for the improvement of precision. In this case, the problem is that there are available two methods; in one method the digital part takes charge of only the 2nd term within the brackets of the 2nd term, and in the other, of the entire part within the brackets. Fig. 15 shows the respective computing circuits. The circuit in (a) is the former case; $\epsilon\dot{y}$ is realized by the analog circuit which corresponds to the



(a)



(b)

Figure 15

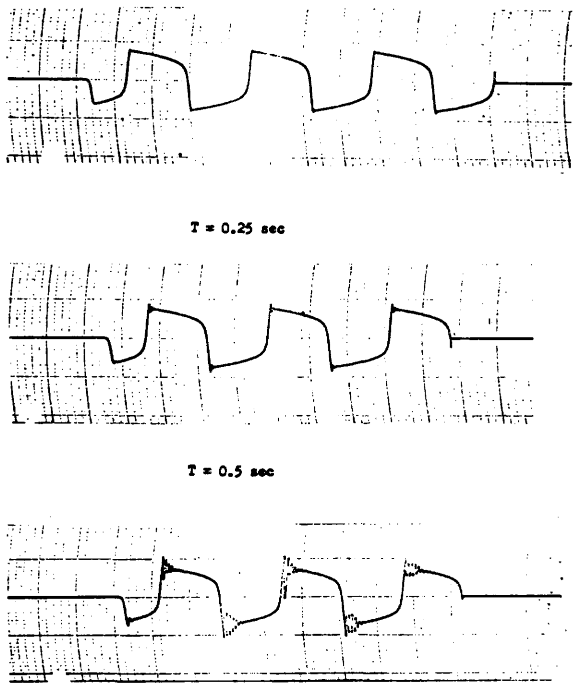


Figure 16A

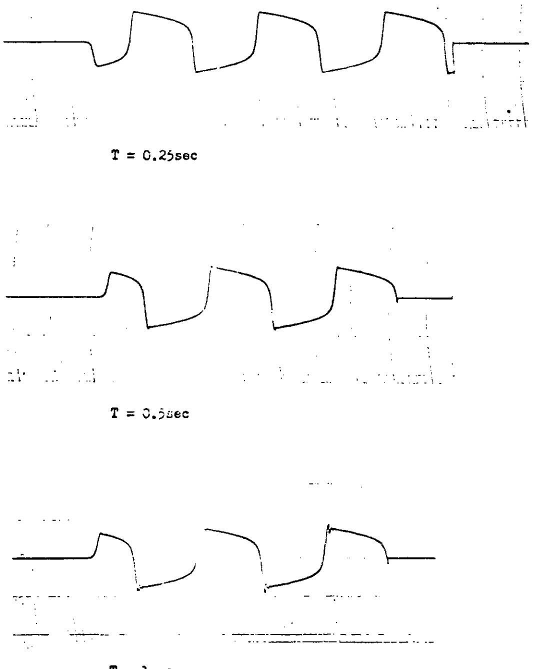


Figure 16B

connection in Fig. 4(c) in which positive feedback circuit is the analog circuit. While, (b) of the figure is the latter case; the entire 2nd term is taken charge of by the digital part and, since there is no positive feedback analog circuit compared with (a), more stable solution is expected.

Fig. 16(a) and (b) are solutions of computation obtained by the computing circuits of Fig. 15(a) and (b), respectively. In Fig. 16(a), an oscillation phenomenon having the period of vibration other than limit cycle is seen, and the degree becomes remarkable with increase of T. In Fig. 16(b), on the contrary, sample error becomes notable with increase of T, but the oscillation phenomenon mentioned above is not caused. It is quite a noteworthy phenomenon in that its oscillation frequency is appreciably lower than the sampling frequency. It has been proved that this phenomenon is caused when continuous dead time element is used, and is eventually due to equivalent delay in the digital computation. Consequently, it is shown that Fig. 15(a) is not proper as computing circuit of the *van der Pol's* equation. In other words, Fig. 15(a) shows that the effect of positive feedback analog path is appreciable, and such computing circuit should be avoided. To this end, it is necessary in determining charge of the digital part in the hybrid computing system, to select as far as possible such computing circuit which gives stable solution with the analog circuit part alone, exclusive of the digital part.

These results of computation agree very well to those obtained with analog simulation circuit for digital computation which is added with execution time and sample hold similar to the case when the contents of digital computation are simulated by analog computation as shown in Fig. 17 and hybrid computation is carried

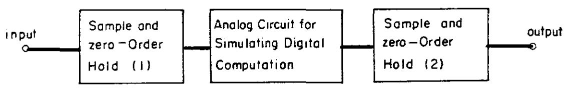

input
Sample and zero-Order Hold (1)
Analog Circuit for Simulating Digital Computation
Sample and zero-Order Hold (2)
output

Figure 17

out actually on it. This fact verifies the analog simulation circuit for digital computation. Analog simulation of digital computation just mentioned above is very effective for knowing effect on solution in hybrid computation by use of linkage and digital computer of such performance that is not yet available.

When the compensating method (analog compensation) mentioned in the preceding chapter is applied in solving the *van der Pol's* equation, the high frequency oscillation can be made very small. An example of solutions is shown in Fig. 18.

6. APPLICATION TO FLIGHT SIMULATORS

Estimation of error due to digital execution time, and the compensating method, described in the preceding chapters, were applied to the

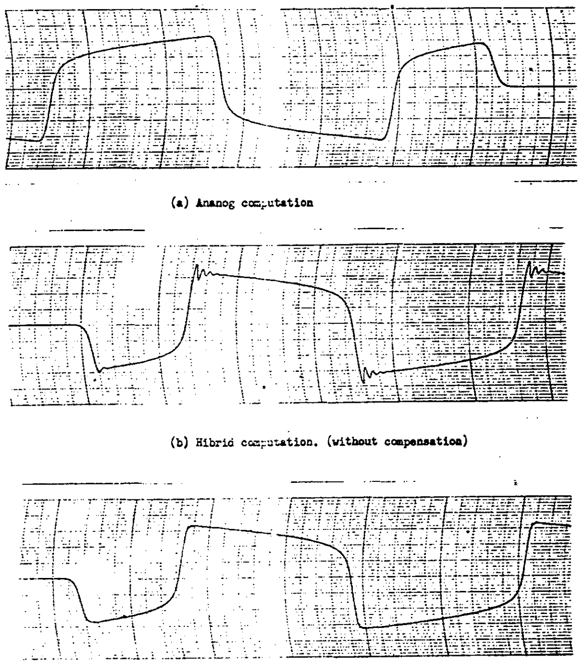


Figure 18

hybrid type flight simulator and the validity was confirmed. For a large flighter the following flight equations are established under some assumption; as to the pitching motion. Here is a discussion on the error mentioned before in solving Eq. (23) on the hybrid computing system.

$$\left.\begin{aligned} \frac{d\Delta V}{dt} &= k_{11}\Delta V + k_{12}\alpha + k_{14}\gamma \\ \frac{d\alpha}{dt} &= k_{21}\Delta V + k_{22}\alpha + k_{23}q \\ \frac{dq}{dt} &= k_{32}\alpha + k_{33}q + k_{34}\gamma \\ \frac{d\gamma}{dt} &= k_{41}\Delta V + k_{42}\alpha \end{aligned}\right\} \quad (23)$$

where $\triangle V$ = Variation of velocity from reference value, %

α = Attack angle, rad.

q = Angular velocity around Y axis, rad./sec.

γ = Flight path angle, rad.

$$k_{11} = -0.00976, \quad k_{12} = -0.04893, \quad k_{14} = -0.04543$$
$$k_{21} = -0.09423, \quad k_{22} = -0.4893, \quad k_{23} = 1$$
$$k_{32} = -0.46466, \quad k_{33} = -0.3626, \quad k_{34} = 0$$
$$k_{41} = -k_{21} \quad k_{42} = -k_{22}$$

Under the condition of vibration at natural frequency, terms including α and q are substantially negligible with respect to terms of $\triangle$V and γ, the equations are reduced as follows.

$$\frac{d\Delta V}{dt} = k_{11}\Delta V + k_{14}\gamma \qquad (24)$$
$$\frac{d\gamma}{dt} = k_{41}\Delta V$$

Coefficient k_{ij} is taken as constant in the above. However, it depends primarily upon the flight conditions, and in its digital computation, effect of execution time is involved. If the transfer function is denoted by $K_{ij}(p)$, the following is obtained as described before.

$$\left.\begin{aligned} K_{ij}(p) &= k_{ij}e^{-T\alpha p} \\ T\alpha &= \tau + \frac{T}{2} \\ \epsilon_{ij} &= e^{-T\alpha p} - 1 \end{aligned}\right\} \qquad (25)$$

Then, the error equation is

$$\epsilon = \left\{ \frac{-\left\{ \begin{vmatrix} -k_{11}(e^{-T\alpha p}-1) & -k_{14} \\ -k_{41}(e^{-T\alpha p}-1) & p \end{vmatrix} + \begin{vmatrix} p-k_{11} & -k_{14}(e^{-T\alpha p}-1) \\ -k_{41} & 0 \end{vmatrix} \right\}}{\begin{vmatrix} 1+k_{11}T\alpha e^{-T\alpha p} & -k_{14} \\ k_{41}T\alpha e^{-T\alpha p} & p \end{vmatrix} + \begin{vmatrix} p-k_{11} & k_{14}T\alpha e^{-T\alpha p} \\ -k_{41} & 1 \end{vmatrix}} \right\}_{p = p\alpha} \qquad (26)$$

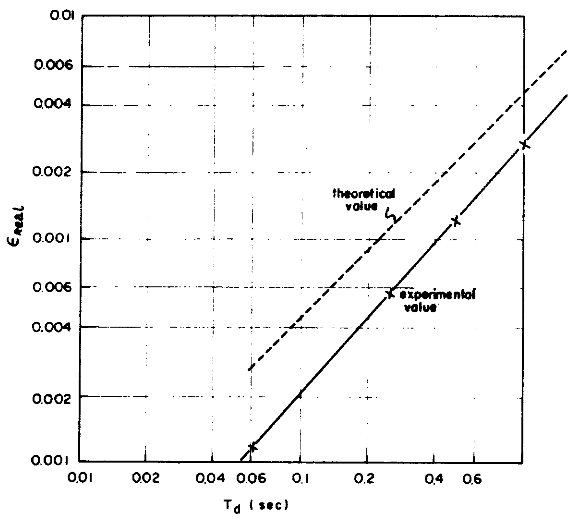


Figure 19

Fig. 19 shows the result of calculation using the coefficient values.

It is seen that, in order to limit below 0.001 or 0.1% in terms of divergence error per sec. (4% in terms of divergence error per cycle), it is necessary to limit the sampling period below 0.2 sec. or so. The corresponding frequency error due to delay is about 0.02% or less and insignificant. In the calculation of

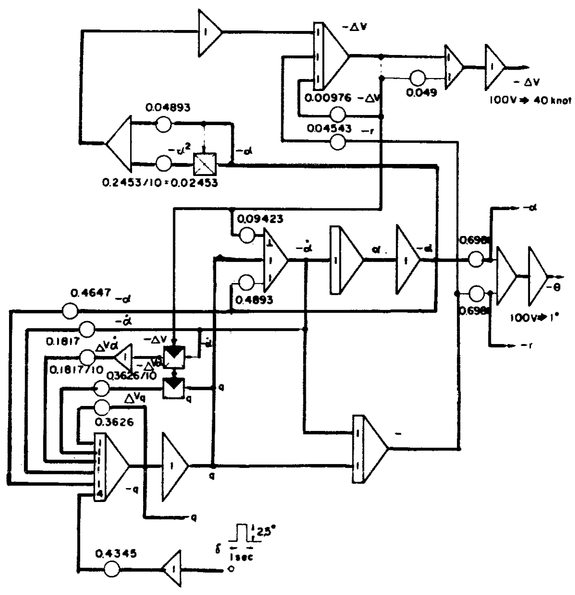
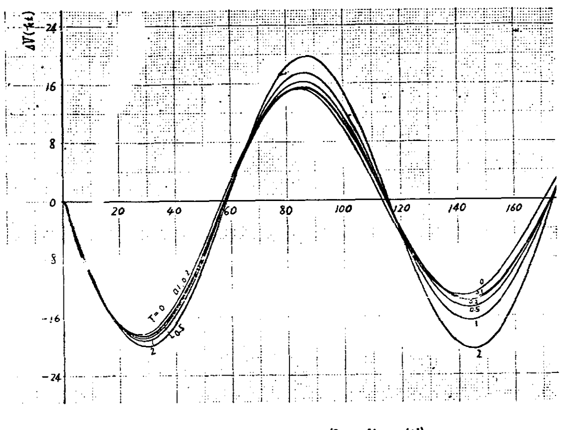


Figure 20

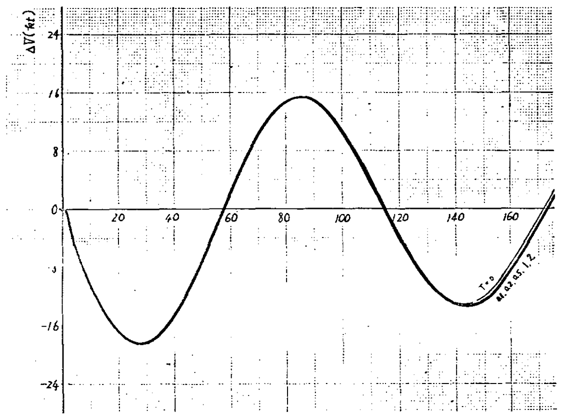
error, making linearization in this way and simplifying to the extent needed are justifiable.

Fig. 20 shows the computing circuit for Eq. (23) (exactly, for a rather complicated equation) determined by taking the execution time of digital part into account. That part enclosed by thick lines in the figure corresponds to the digital part. Fig. 21(a) is its solution. It can



(a) Waveforms of ΔV without compensation. (T: sampling period)

Figure 21a



(b) Waveforms of ΔV without compensation. (T: sampling period)

Figure 21b

be known by the diagram that computation error increases in accordance with the change of sampling interval T. When T is over 0.2 sec., the error becomes large, and when T is 2 sec. the solution diverses. In this system T should be less than 100m sec. The relation of ϵ — T_d is shown in Fig. 19. The dotted line shows theoretically calculated value and the line shows the result of experiments, and their results coincide well. Fig. 21(b) shows the case when its conditions are the same as Fig. 21(a) and the error is compensated. In this case, computation accuracy improves remarkably as is shown in the diagram. This result shows that computation with sufficiently high accuracy can be done even when T is 2 sec.

This means that, with application of this compensating method, computation of dynamic characteristics of high response is possible even on a digital computer of relatively low speed.

7. CONCLUSION

Effects of digital execution time in a hybrid computing system was considered, and on this problem, the authors obtained the following results.

(1) It was found theoretically as well as experimentally that the transfer function of digital computing part is given approximately by $Ae^{-(\tau + T/2)}$ where A denotes the contents of computation on digital computer, T the sampling period and τ the execution time of digital computer.

(2) General formula for the computation error due to digital computer execution time was obtained, and it was shown that proper assignment of computation and sampling period can be calculated.

(3) Three methods of compensating for aforementioned dead time were proposed together with theoretical and experimental analysis of their effects.

(4) In the application to the solution of the *van der Pol's* equation and to the flight simulators, practicability was confirmed.

Finally the authors gratefully acknowledge the kind support of Dr. Sudo, Manager of the Department.

REFERENCE

1. T. MIUTA and M. NAGATA: IRE Trans. EC-7, 4(1958).

APPENDIX: DEVELOPMENT OF A SAMPLE-HOLD WAVE FORM INTO FOURIER SERIES

In the sample-hold of a sinusoidal-wave as shown in Fig. 2, denote by a the angular fre-

CORRECTED INPUTS—A METHOD FOR IMPROVED HYBRID SIMULATION

Robert Gelman
General Electric Company
Re-Entry Systems Department
Philadelphia 1, Pennsylvania

I. INTRODUCTION

The purpose of this paper is to describe a programming procedure which is designed to minimize some of the difficulties often encountered in hybrid simulations. The method has been worked out for the hybrid system at General Electric's Re-Entry Systems Department, in Philadelphia. This system consists of general purpose analog and digital machines, connected by analog to digital (A-D) and digital to analog (D-A) converters, and associated logical elements. The procedure itself will be referred to as the method of corrected inputs. It is designed to minimize three major sources of error connected with data transfer. These are:

1. *D-A Data*
   Due to the nature of the digital computer, each variable which it sends to the analog is seen by the analog as a stepped function. This introduces errors because of the differences between this function and the actual one, and also because of the response of the analog to discontinuous inputs. Filtering this input can smooth the discontinuities somewhat, but introduces lags and distortions.
2. *A-D Data*
   The analog output is sampled and sent to the digital computer no more than once during each digital computation interval. If the digital time step is larger than, or even a significant portion of, the fundamental frequency of the analog output, then the A-D data could actually be a very poor representation of the analog outputs.
3. *Time Lags*
   The minimum time required for an output from a subsystem to affect the operation of the subsystem itself would be the sum of the A-D and D-A time lags. This could render hybrid operation very difficult, if not completely useless, in studies of the performance and stability of control systems, or other studies involving leads, lags, or phase relationships.

The method of corrected inputs provides a simple way of minimizing all these difficulties. The basic idea behind the method is to represent, on the analog, some of the functions which are also calculated on the digital computer. The outputs of the digital simulation would then be used, not as inputs to the main analog simulation, but as corrections to the comparable analog portion.

II. DESCRIPTION OF METHOD

The following is a brief summary of the system. The timing is arranged so that the digital computations lag the analog by the digital computation interval. Those inputs to the main analog program which are sensitive to phase, delays, frequency response, or similar factors, are obtained from subsections of the analog program. These inputs are sampled and sent to the digital computer, where they are stored until the digital simulation produces the corresponding quantities, presumably with more accuracy. The differences are then returned to the analog computer which generates corrections, in the form of ramp functions, to the analog inputs. The only requirement imposed on the analog subsystems involved is that the analog functions not diverge appreciably from the comparable digital functions in less than two digital computation intervals.

The analog inputs to the digital program are not sent directly, unless they are slowly varying functions. Otherwise, each one is averaged over the time period of one computation interval, and this average is sent to the digital. The fact that the digital calculations lag the analog enables the digital computations to be made using inputs from the analog which are the average values for the period of time covered by the calculations. Thus, the method minimizes the three major drawbacks to accuracy in hybrid simulations: data transfer time lags, discontinuous D–A signals, and unrepresentative A-D signals.

The manner in which this method can be used to meet the basic difficulties described above can best be understood if it is explained in more detail. This is done with the aid of figures 1, 2, and 3.

Figure 1 is a schematic representation of a hybrid simulation. The box F_1 represents, let us say, the low frequency portion of the prob-

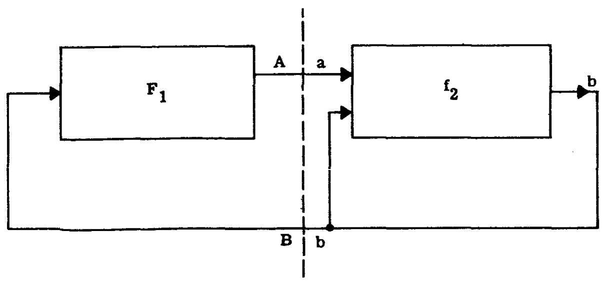


Figure 1. Schematic representation of a hybrid simulation.

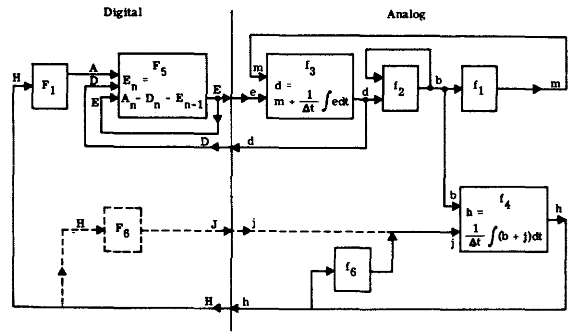

Digital
Analog
H
F_1
A
D
E
F_5
$E_n = A_n - D_n - E_{n-1}$
E
e
m
f_3
$d = m + \frac{1}{\Delta t}\int e\,dt$
d
f_2
b
f_1
m
D
d
b
f_4
$h = \frac{1}{\Delta t}\int (b + j)\,dt$
h
H
F_6
J
j
j
f_6
H
h

Figure 2. Hybrid simulation, modified for method of corrected inputs.

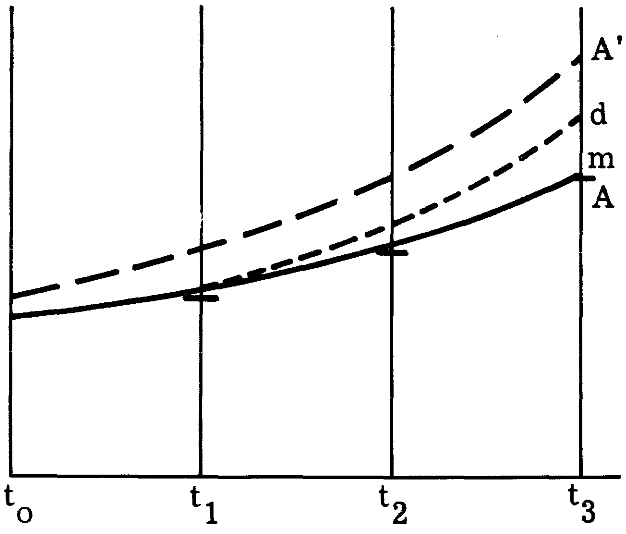


Figure 3-A. Variables during corrected D-A data transfer.

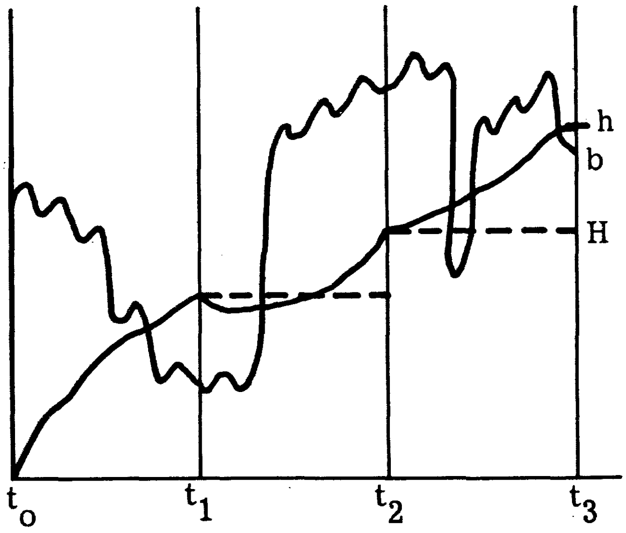


Figure 3-B. Variables during A-D data averaging procedure.

lem, solved on the digital computer, and box f_2 represents the high frequency portion, simulated on the analog. Output b is shown feeding back into f_2 in order to illustrate the possibility that the behavior of f_2 can depend on high frequency as well as low frequency inputs. Note that all digital functions are represented by capital F's, analog functions by small f's, digital variables by capital letters, and analog variables by small letters. When the same letter is both capitalized and lower case, it represents either a variable that is converted A-D or D-A, or a function which is simulated on both computers.

Figure 2 shows the simulation of figure 1, as modified for the method of corrected inputs. The modified circuit is not as much more complicated than the original one as these diagrams would make it appear. The analog function, f_3, consists of two amplifiers; f_4 consists of one. The digital operation F_5 consists of summing three numbers; F_6 consists of moving one number from an input location to an output. The analog function f_1 could be complex, but experience shows that it is usually possible to generate, by very simple means, a function which will be close enough to F_1 for our purposes. The function f_6, if used, is a sample and hold circuit.

The D-A data transfer will be explained first. The normal hybrid simulation would convert A directly to its analog equivalent, a, and feed it into f_2, as shown in figure 1. In the method of corrected inputs, f_2 receives its basic input, the quantity m, from f_1, as shown in figure 2. A correction is added to m before it is fed into f_2, so that f_2 receives the quantity d, which is a corrected m. The function f_3 is an integration and an addition. That is,

$$d = m + \int e\,dt \qquad (1)$$

The behavior of the variables is shown in figure 3-A. The dashed line, A′, shows what A would look like if it were calculated continuously and in phase with the analog. At time t_0, the quantity d is sampled as d_0, which is equal to m_0. It is converted to the digital number D_0, and stored in digital memory. At time t_1, the quantity A_0 has been computed, and the difference, E_0 ($=A_0 - D_0$), is converted to e_0, and fed into f_3. Thus, for the period from t_1 to t_2,

$$d = m + (A_0 - D_0)\frac{t - t_1}{t_2 - t_1} \qquad (2)$$

Also at t_1, the quantity d_1 ($=m_1$) is converted to D_1 and stored. Then, at t_2, the quantity $A_1 - D_1$ is computed. Here we have to consider one of the more subtle points of the process. The quantity D_1 does not reflect the correction, e_0, which has already been added to m.* For illustrative purposes, consider the special case, $m = A' - K$ is a constant. Then we would find that

$$A_0 - D_0 = K \qquad (3)$$
$$A_1 - D_1 = K \qquad (4)$$
$$A_2 - D_2 = 0 \qquad (5)$$

If we use, for E, the general formula

$$E_n = A_n - D_n, \qquad (6)$$

then we would obtain for consecutive values of d,

$$d_0 = A' - K \qquad (7)$$
$$d_1 = A' - K \qquad (8)$$
$$d_2 = A' \qquad (9)$$
$$d_3 = A' + K \qquad (10)$$
$$d_4 = A' + K \qquad (11)$$
$$d_5 = A' \qquad (12)$$
$$d_6 = A' - K \qquad (13)$$

etc.

If we use the formula

$$E_n = A_n - D_n - E_{n-1}, \qquad (14)$$

we would obtain for consecutive values of d,

$$d_0 = A' - K \qquad (15)$$
$$d_1 = A' - K \qquad (16)$$
$$d_2 = A' \qquad (17)$$
$$d_3 = A' \qquad (18)$$

etc.

Thus, the subtraction of the preceding correction term removes a phase lag oscillation from the D-A input. It is apparent from the foregoing that a condition is imposed on f_1, requiring that the amount that m diverges from A

* From the procedure, it follows that $d_n = m_n + \sum_{i=0}^{n-2} e_i$

in the time $t_{n+2} - t_n$, is no greater than the acceptable error in d. That is, from the time the variable d is sampled to the time the corresponding correction has been added to it, two time cycles have elapsed. Thus, as long as the inaccuracies in the generation of m do not cause it to drift more than an acceptable amount in two time cycles, it will be satisfactory.

The A-D data transfer is illustrated in figure 3-B. The variable, b, is shown as having high frequency components, and large excursions, so that samples taken at each digital time interval might not be representative. The solution consists of using b to construct another function, h, whose value at any sampling time, t_n, is equal to the average value of b during the interval $t_n - t_{n-1}$. The resulting value, h_n, is converted to H_n, and fed into F_1. The digital computer, at time t_n, is just starting the computations for the problem interval between t_{n-1} and t_n. All the variables will have values corresponding to time t_{n-1}, except those coming from the analog, which will have values equal to what their average will be during the time interval, $_{n-1}$ to t_n, for which the computations are to be made. It is to be noted that any high speed system responses are automatically reflected in b because of the closed analog loop through f_1, f_3, and f_2.

The manner in which h is produced from b is, in principle, very simple. In the first time interval, from t_0 to t_1, the only input to f_4 is b. Assuming that $t_n - t_{n-1}$ is a constant,† then it is a simple matter to adjust f_4 such that,

$$h_1 = \int_{t_0}^{t_1} bdt/(t_1 - t_0). \qquad (19)$$

The quantity h_1 is sampled, and fed back as soon as possible as j_1. Thus, at time t_2, we find,

$$h_2 = h_1 + \int_{t_1}^{t_2} bdt/(t_2 - t_1) - \int_{t_1}^{t_2} j_1dt/(t_2 - t_1). \qquad (20)$$

But, the first and third terms on the right side of this expression cancel out, leading to the general result,

† This assumption is not necessary, but the use of varying time increments would require the use of nonlinear analog equipment; specifically, a multiplier.

$$h_n = \int_{t_{n-1}}^{t_n} bdt/(t_n - t_{n-1}). \qquad (21)$$

The manner in which h is fed back to f_4 as j will affect the timing of the data transfer, as well as the accuracy of h. The output, h, can be sampled and fed directly back to f_4, by adding a sample and hold circuit, shown as f_6, to the analog simulation. Or, h can be sent to the digital computer as H, sent back immediately as J, and converted to j. I prefer the former method, as it leads to a neater data transfer routine. Therefore, the timing sequence which follows assumes that f_6 is in the circuit, and F_6 is out. The alternate sequence, based on F_6 in the circuit and f_6 out, is included as appendix A.

Figure 4-A illustrates the timing sequence involved in the operation of the method of corrected inputs, assuming the inclusion of f_6 in the analog simulation. The actual D-A data transfer will occur in a time interval τ seconds

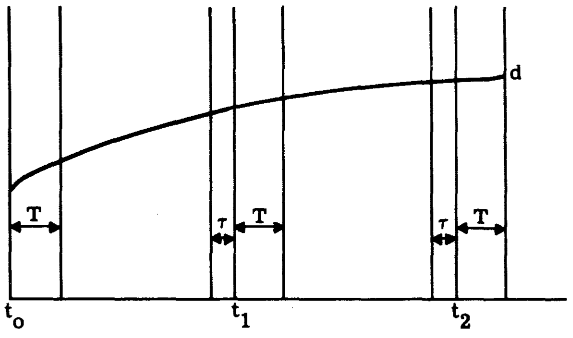


Figure 4-A. Timing sequence if analog averaging circuit is used.

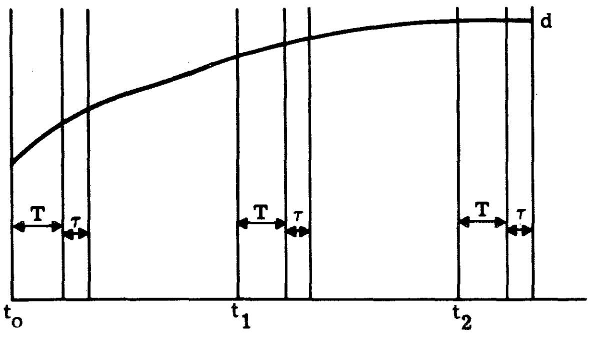


Figure 4-B. Timing sequence if hybrid averaging circuit is used.

long, and the A-D transfer in an immediately subsequent interval lasting T seconds. The sequence of operations follows:

$\underline{t_0}$

Analog variables sampled. (d_0 and h_0)

h_0 (=0) fed back to f_4

$\underline{t_0 \rightarrow (t_0 + T)}$ (A-D data transfer)

$d_0 \; (= m_0) \rightarrow D_0$

$h_0 \; (= \;) \rightarrow H_0$

$\underline{(t_0 + T) \rightarrow (t_1 - \tau)}$ (Digital computations)

$E_0 = A_0 - D_0$

$\underline{(t_1 - \tau \rightarrow t_1}$ (D-A data transfer)

$E_0 \rightarrow e_0$

$\underline{t_1}$

Analog variables sampled. (d_1 and h_1)

$$h_1 \left(= \int_{t_0}^{t_1} bdt/(t - t) \right) \text{ fed back to } f_4$$

$\underline{t_1 \rightarrow (t_1 + T)}$ (A-D data transfer)

$d_1 \; (= m_1) \rightarrow D_1$

$h_1 \rightarrow H_1$

$\underline{(t_1 + T) \rightarrow (t_2 - \tau)}$ (Digital computations)

$E_1 = A_1 - D_1 - E_0$

$\underline{(t_2 - \tau) \rightarrow t_2}$ (D-A data transfer)

$E_1 \rightarrow e_1$

$\underline{t_2}$

Analog variables sampled. (d_2 and h_2)

$$h_2 \left(= \int_{t_1}^{t_2} bdt/(t_2 - t_1) \right) \text{ fed back to } f_4$$

$\underline{t_2 \rightarrow (t_2 + T)}$ (A-D data transfer)

$d_2 \; (= m_2 + e_0) \rightarrow D_2$

$h_2 \rightarrow H_2$

III. RESULTS OF AN ANALOG DEMONSTRATION

A simplified three degree of freedom simulation of the flight and control of a missile was chosen to illustrate the method of corrected inputs. A description of the simulation is given in appendix B. This work was done on the analog, because the digital computer which comprises half of the hybrid facility is being replaced, a process which will not be completed until early in 1964. In order to simulate a hybrid operation on an analog computer, it was necessary to make simplifications, compromises, and omissions. Even so, it is felt that a valid and informative demonstration was obtained. The main differences between this analog demonstration and a full scale hybrid simulation are:

1. The data transfer took place at much lower rates. Most of the runs used transfer rates between one and ten cycles per second, instead of the 0.1 to 1 KC rates that are customarily used. This was not due to limitations of the equipment, but to the fact that it is much easier to demonstrate the method at the lower rates.
2. The digital equations were simulated on the same time base as the analog equations, with no lag. It is easy enough to store a value in the digital computer until the time comes to use it, but it was not felt to be worth the effort to accomplish the same thing on the analog.
3. The A-D averaging procedure was set up and run, but it was not used in the full simulation, as the fact that the digital lag was not included would have made it a hindrance, rather than a help. Figures 5 and 6 show examples of its operation at 1 and 10 cps respectively.

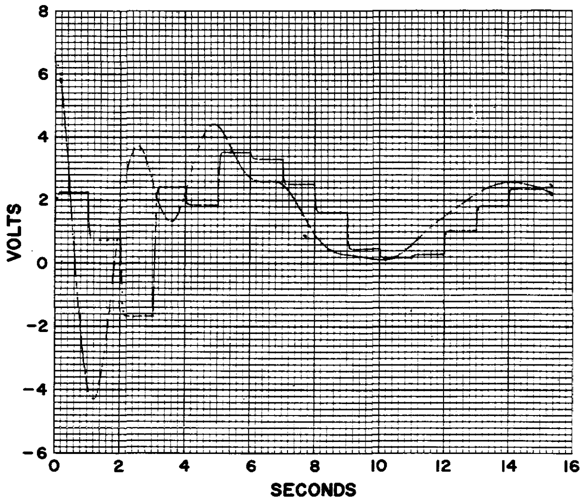


Figure 5. L, averaged and sampled at 1 cps.

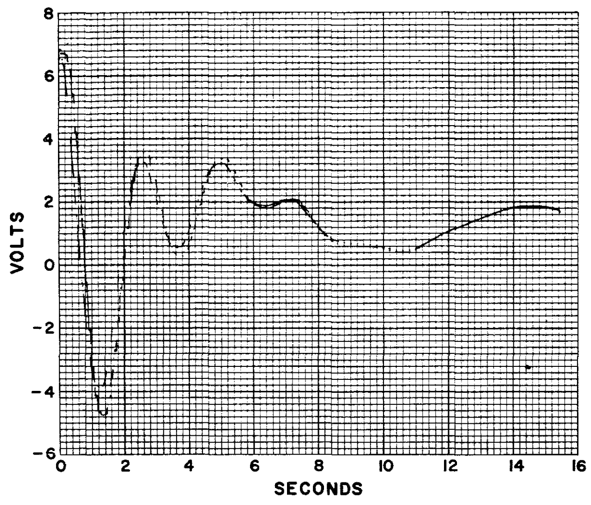


Figure 6. L, averaged and sampled at 10 cps.

Figure 7 is a schematic of the analog demonstration. The nominal case, against which the others can be compared, is set up when both switches are in position 1. Both switches are put in position 2 to represent a standard hybrid simulation. Putting switch A in position 4 makes the circuit represent an all analog simulation, and putting it in position 3 includes the D-A correction circuit.

Figure 8 shows the recorder outputs corresponding to switch B in position 1, and switch A in positions 1, 2, and 3, respectively. Sampling rate was 2 cps.

Figure 9 shows the variable, y, measured at points 1, 4, 3, and 2 of switch A (figure 7) with both switches in position 1. The sampling rate was 2 cps. Comparison of the third and fourth curves of this figure illustrates the difference between a sampled and a corrected input. Note that the approximate function differs appreciably from the controlling one.

Figures 5 and 6 show the operation of the A-D correction circuit. Figure 5, obtained with a one cps sampling rate, illustrates how each sampled value of the output is equal to the average value of the input function during the preceding sample and hold cycle.

In order to get some numerical indication of the effectiveness of each procedure, a function, Q, was generated, such that

$$Q = k \int y^2 \, dt$$

where the effectiveness is considered to be inversely proportional to Q. Table 1 shows the results of measuring Q at the termination of each run. All the runs stopped automatically, at $x = 9000$ ft. A comparison of the first three columns of this table is shown in figure 10. The data in these curves were taken without A-D sampling, so that they are a measure of the type of D-A transfer used, and of the sampling frequency. Curve A shows the results of D-A transfer, and represents an optimum result, obviously independent of sampling frequency. Curve B plots results with corrected D-A inputs, which can be seen to be far superior to the results obtained by sampling the D-A inputs, shown as curve C.

IV. SUMMARY

In summary, let us see how the method of corrected inputs minimizes the three sources of error listed at the beginning of this paper.

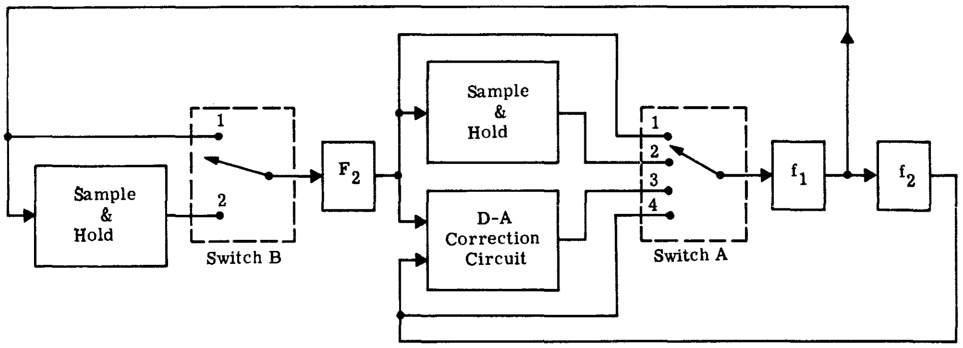


Figure 7. Schematic diagram of analog simulation of method of corrected inputs.

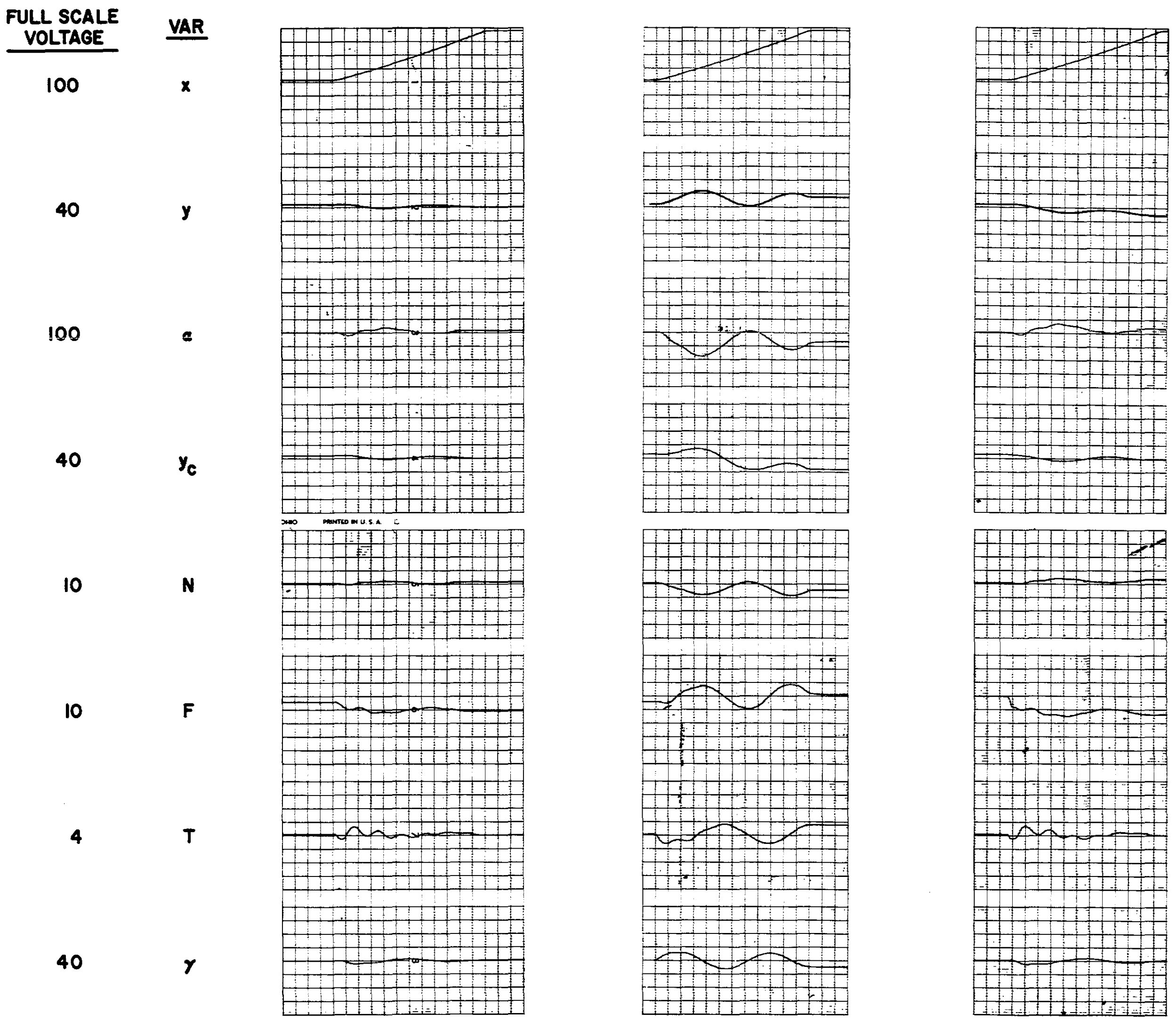


Figure 8. Recorder outputs for runs with direct D-A, sampled D-A, and corrected D-A, respectively (two cps).

1. *D-A Data Transfer Errors*
   The input to the main analog program is smooth instead of stepped. It is in phase with the simulation into which it is fed. It is constantly corrected; the correction is no more than two digital computation intervals behind the actual problem.
2. *A-D Data Transfer Errors*
   The digital computer receives the analog variable, already averaged over the interval for which the digital computations are to be made, instead of sampled at its value at the start of that interval. Any errors introduced by the averaging process will tend to cancel themselves out, rather than accumulate.
3. *Time Lags*
   The simulation can respond immediately to variations in any portion of the system. This ability is completely independent of either the A-D or the D-A transfer time requirements.

Thus, we have a system which ameliorates difficulties which were thought by many to be intrinsic to hybrid simulation.

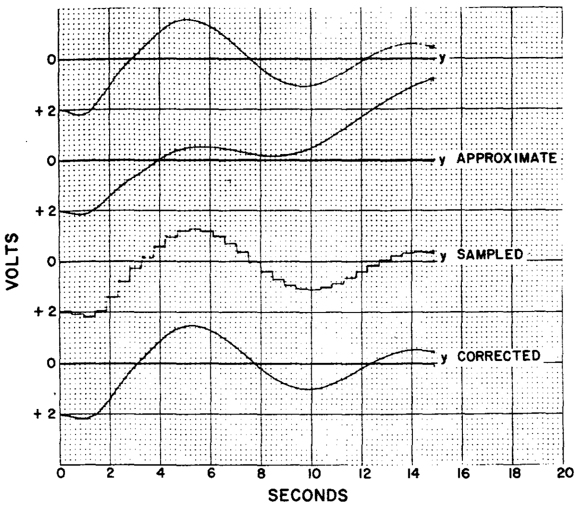


Figure 9. Variable, y, measured at various points during nominal run.

APPENDIX A

ALTERNATE TIMING SEQUENCE FOR METHOD OF CORRECTED INPUTS

The timing sequence shown in this appendix would be used in the event that the correction term j, which is fed to f_4 (see figure 2), originates in the digital computer, rather than the analog circuitry. The comments refer to figure 4-B. All A-D data transfers occur in the time intervals T, and all D-A data transfers occur in the immediately subsequent time intervals τ.

$\underline{t_0}$

Analog variables sampled. (d_0 and h_0)

$\underline{t_0 \rightarrow (t_0 + T)}$ (A-D data transfer)

$d_0 \ (= m_0) \rightarrow D_0$

$h_0 \ (=_0) \rightarrow H_0$

$\underline{(t_0 + T) \rightarrow (t_0 + T + \tau)}$ (D-A data transfer)

$J_0 \ (= H_0) \rightarrow j_0$

$\underline{(t_0 + T + \tau) \rightarrow t_1}$ (Digital computations)

$E_0 = A_0 - D_0$

$\underline{t_1}$

Analog variables sampled. (d_1 and h_1)

$\underline{t_1 \rightarrow (t_1 + T)}$ (A-D data transfer)

$d_1 \ (= m_1) \rightarrow D_1$

$$h_1 \left(= \int_{t_2}^{t_1} b dt / (t_1 - t_0) \right) \rightarrow H_1$$

$\underline{(t_1 + T) \rightarrow (t_1 + T + \tau)}$ (D-A data transfer)

$J_1 \ (= H_1) \rightarrow j_1$

$E_0 \rightarrow e_0$

$\underline{(t_1 + T + \tau)} \rightarrow$ (Digital computations)

$E_1 = A_1 - D_1 - E_0$

$\underline{t_2}$

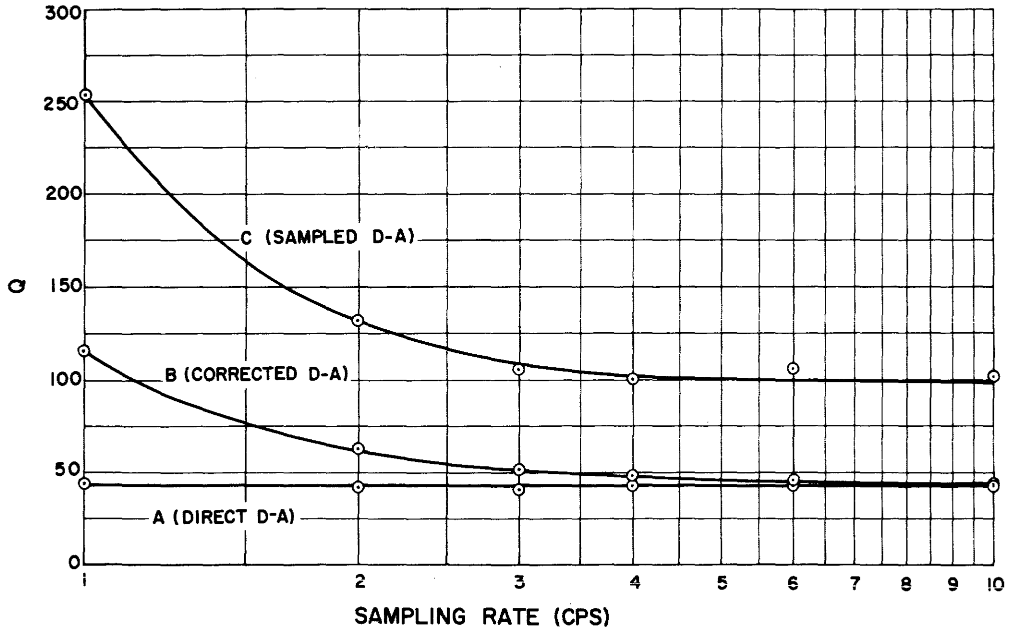


Figure 10. Q as a function of D-A sampling method and sampling rate.

Analog variables sampled. (d_2 and h_2)

$\underline{t_2 \rightarrow (t_2+T)}$ (A-D data transfer)

$d_2 \ (=m_2+e_0) \rightarrow D_2$

$$h_2 \left(= \int_{t_1}^{t_2} b dt/(t_2 - t_1) \right) \rightarrow H_2$$

$\underline{(t_2+T) \rightarrow (t_2+T+\tau)}$ (D-A data transfer)

$J_2 \ (=H_2) \rightarrow j_2$

$E_1 \rightarrow e_1$

APPENDIX B

ANALOG SIMULATION TO DEMONSTRATE METHOD OF CORRECTED INPUTS

This all analog simulation was split into so-called "digital" and "analog" portions, to represent those parts of the problem which would normally be assigned to each computer. The equations for calculating the acceleration, velocity, and position vectors were considered digital, and the equations involving the control system, applied forces and torques, and rotational dynamics were considered analog.

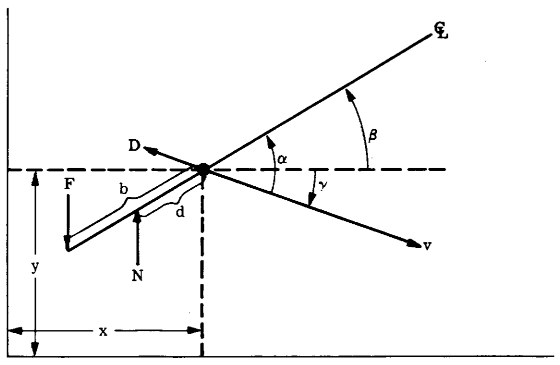


Figure 11. Diagram showing symbols used for dimensions and variables in the sample problem.

Referring to figure 11, the analog equations are:

$$\ddot{\beta} = -bF/I - dN/I - T/I \qquad \text{(B-1)}$$

$$\dot{\beta} = \dot{\beta}_0 + \int \ddot{\beta}\, dt \qquad \text{(B-2)}$$

$$\beta = \beta_0 + \int \dot{\beta}\, dt \qquad \text{(B-3)}$$

$$a = \beta - \gamma^{‡} \qquad \text{(B-4)}$$

$$N = k_1\, a\, \dot{x}^2 \qquad \text{(B-5)}$$

$$F = k_3 y^{‡} + k_4 \dot{y}^{‡} + k_5 \beta + k_6 \dot{\beta} \qquad \text{(B-6)}$$

$$T = k_2 \dot{\beta}\dot{x}^{‡} \qquad \text{(B-7)}$$

$$L = N + F \qquad \text{(B-8)}$$

and the digital equations are:

$$\ddot{x} = -D/M + k_7 \qquad \text{(B-9)}$$

$$\dot{x} = \dot{x}_0 + \int \ddot{x}\, dt \qquad \text{(B-10)}$$

$$x = x_0 + \int \dot{x}\, dt \qquad \text{(B-11)}$$

$$D = k_8 x^2 \qquad \text{(B-12)}$$

$$y = y_0 + \int \dot{y}\, dt \qquad \text{(B-13)}$$

$$y = \dot{x}\gamma \qquad \text{(B-14)}$$

$$\dot{\gamma} = L^{**}/m\dot{x} \qquad \text{(B-15)}$$

$$\gamma = \gamma_0 + \int \dot{\gamma}\, dt \qquad \text{(B-16)}$$

The quantities y, $\dot{y}$, and γ are approximated by a section of the simulation which replaces equations B-13, B-14, B-15, and B-16 with the equations:

$$y = y_0 + \int \dot{y}\, dt \qquad \text{(B-17)}$$

$$\dot{y} = \dot{y}_0 + \frac{1}{m} \int L\, dt \qquad \text{(B-18)}$$

$$\gamma = \dot{y}/\dot{x} \qquad \text{(B-19)}$$

For this simulation, the values of the constants were:

$m = 1.555 \times 10$	pound-sec²/ft
$I = 6.22 \times 10$	pound-ft-sec²
$b = 5$	feet
$d = 2$	feet
$k_1 = 3 \times 10^{-4}$	pound-sec²/ft²
$k_2 = 1.73 \times 10^{-3}$	pound-sec²
$k_3 = 1.06 \times 10$	pounds/ft
$k_4 = 5.8$	pound-sec/ft
$k_5 = 7.16 \times 10$	pounds

‡ The quantities y, $\dot{y}$, γ, and $\dot{x}$ are received from the digital portion of the simulation.

§ The quantity L is received from the analog portion of the simulation.

$k_6 = 1.83 \times 10$		pound-sec
$k_7 = 3 \times 10$		feet/sec^2
$k_8 = 1.25 \times 10^{-4}$		pound-sec^2/ft^2

In order to simulate the operation of the corrected D-A inputs, the variables y, $\overset{\circ}{y}$, and γ were sent through the circuit of figure 12-A (the D-A correction circuit box of figure 7). In this figure,

v = the desired function
v' = the approximate function
v_s = the output of the sample and hold circuit
v_f = the correcting and feedback voltage
v_c = the final corrected output

A1, A2, and A4 are summing amplifiers, and A3 is an integrator. In operation, the potentiometer, P, is set so that

$$\overset{\circ}{v}_f . = v_s/T_c,$$

where T_c is the period of the sample and hold circuit. The result is that v_f is a ramp voltage whose value at the end of each hold period is equal to the difference between $\dot{v}$ and v′ at the time of sampling. The problem was run using this correction circuit, both with and without sampling the variable L.

The A-D correction circuit (not the sampling circuit) is shown in figure 12-B. The output, v_c, is equal, during the hold period, to the average value of the input, v, during the preceding sample and hold period. This circuit, while operative (see figures 5 and 6), was not used in conjunction with the entire simulation.

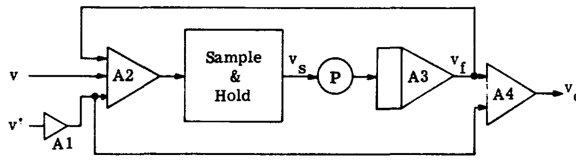


Figure 12-A. Analog simulation of D-A correction circuit.

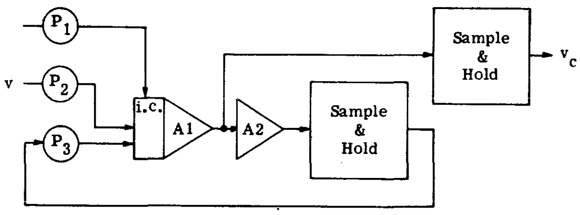


Figure 12-B. Analog simulation of A-D correction circuit.

TABLE 1—VALUES OF Q

Type of data transfer		Sampling Rate, cps					
D-A	A-D	1	2	3	4	6	10
Direct	Direct	44	42	41	43	44	42
Sampled	Direct	254	132	105	100	106	102
Corrected	Direct	116	63	51	48	46	41
Approx.	Direct	810	726	736	800	707	892
Direct	Sampled	272	126	100	96	94	94
Sampled	Sampled	—*	—*	266	207	171	164
Corrected	Sampled	890	334	196	161	135	123
Approx.	Sampled	—*	—*	—*	—*	—*	—*

* Unstable, went into overload.

A HYBRID ANALOG-DIGITAL DIFFERENTIAL ANALYZER SYSTEM

John V. Wait
Department of Electrical Engineering
University of Arizona, Tucson, Arizona

I. DESCRIPTION OF HYBRID SYSTEM

In recent years, considerable effort has been expended toward combining the capabilities of digital and analog elements in specialized computing systems. The concept of blending analog and digital elements can be extended to a system of true hybrid computing elements.

Consider the possibilities of a differential analyzer representing the value of each variable (and parameter) by a combination of a coarse digital word together with a continuous analog interpolation voltage. In theory, at least, it would appear possible that the accuracy of the analog channel would be improved by roughly ½n, where n is the number of digital bits used in the digital representation.

A little thought will reveal the obvious restriction that one must trade speed (whether viewed in terms of frequency, rise time, or slewing rate) for accuracy. This same limitation, of course, applies to DDA's, and to some extent, to analog computing systems. The hybrid differential analyzer may be regarded as a relatively inexpensive parallel DDA whose truncation and round-off errors are essentially eliminated through interpolation with analog computing elements. This feature, rather than accuracy or speed as such, is considered to be the salient advantage of such a system. Considerable work has already been done in this area by Skramstad, Schmid, Korn, and others (Refs. 2, 5, 11, and 13). The purpose of this study was to explore in detail the practical aspects of hybrid differential analyzers, and to verify experimentally the capabilities and limitations of a typical system.

The major portion of the experimental work associated with this paper is devoted to the performance of hybrid integrators. Later sections will discuss the requirements of hybrid elements for performing all of the above operations. Figure 1 shows abbreviated block diagrams of typical hybrid computing elements; details of the design of these elements appear in subsequent chapters. Reference 18, upon which this paper is based, provides further information.

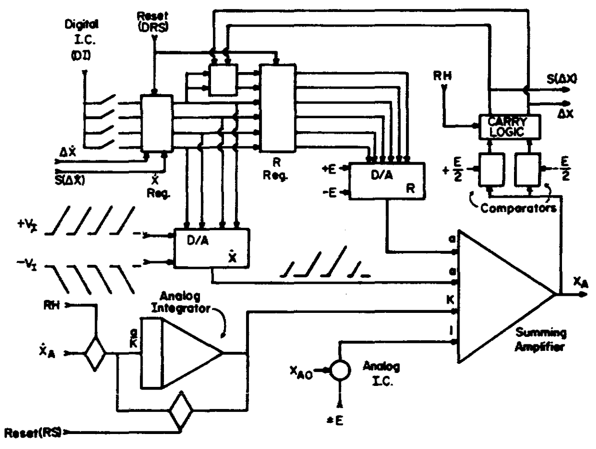


Figure 1(a). Hybrid Integrator.

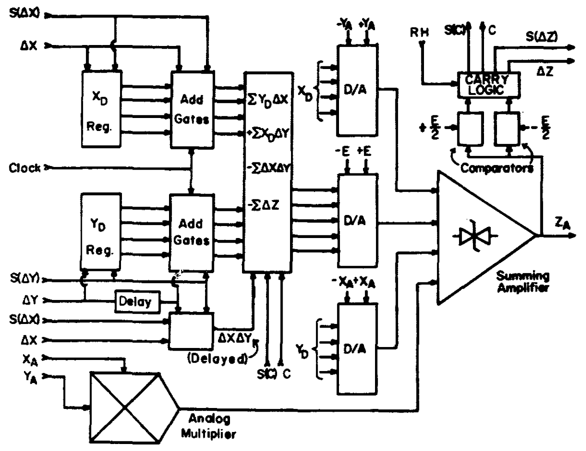


Figure 1(b). Hybrid Multiplier.

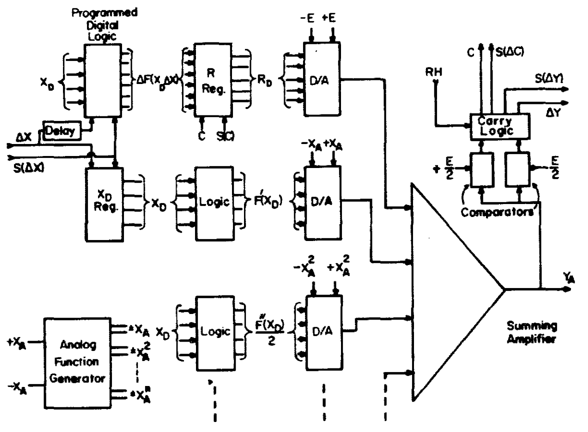


Figure 1(c). Hybrid Function Generator.

*System Notation**

Consider a system where each problem variable x is represented by a machine variable $X = a_x x$, appearing in the form

$$X = X_D + X_A$$

where X_D is a digital word with n binary digits plus sign bit; X_A is an analog voltage between —E and +E volts. Either 1 binary digit or E volts represents 1 *machine unit* (m.u.). We note that X_A is an interpolating voltage representing the fractional part of X (Fig. 2).

In this study, $n = 3$, $(2^n = 8)$ and $E = 10$ volts.

Assuming analog-computer accuracy within p per cent of E, this representation yields $100 \frac{2^n}{p}$ distinguishable increments of X between 0 and 2^n, or a half-scale accuracy of $2^{-n}p$ per cent. In this study p was estimated to be one per cent; thus, the estimated accuracy is 1/8 per cent.

* The material presented in this section is primarily derived from discussions with Prof. G. A. Korn, whose suggested notation is used throughout this paper.

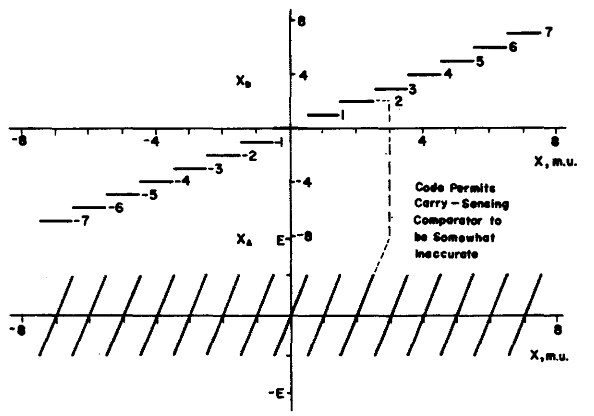


Figure 2. Hybrid Variable Representation.

For any analog voltage e between 0 and 0.5 machine unit, note that both $X_D + e$ and $(X_D + 1) + (e - 1)$ represent the same value of the hybrid machine variable X. Although this redundancy halves the analog resolution, it permits us to use relatively inaccurate analog comparators to generate carries.

Representation of the Independent or Time Variable (Fig. 3)

The range of the independent variable $t \geq 0$ is divided into equal increments Δt, so that

$$t = (k - 1)\,\Delta t + \frac{1}{a_t}\tau;\ (k = 1, 2, \ldots)$$

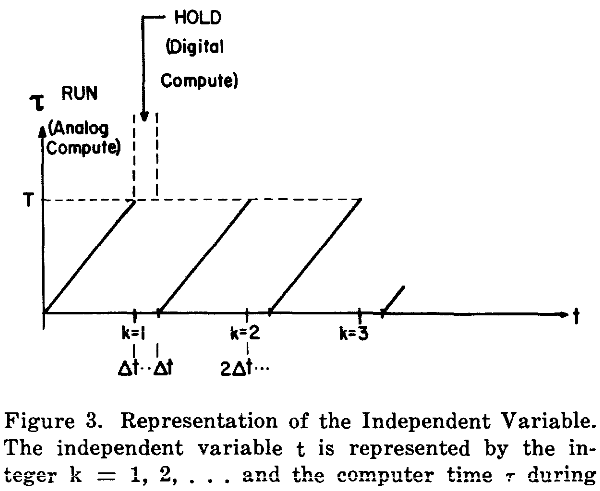


Figure 3. Representation of the Independent Variable. The independent variable t is represented by the integer k = 1, 2, . . . and the computer time τ during successive computing periods.

where τ varies periodically between 0 and $a_t \Delta t$ as t increases. Each interval of length Δt will correspond to an individual analog-computing period of duration T, during which

$$\tau = a_t [t - (k-1) \Delta t] \geq 0;$$
$$(k = 1, 2, \ldots); \; (0 \leq \tau \leq T)$$

is the *computer time* (real time); a_t is a *time scale factor* suitably chosen so that

$$a_t \Delta t = T$$

After each run, a holding interval of length T_H is used for performing digital updating operations, generating analog carries and resets, etc. This is a significant departure from earlier hybrid differential analyzer systems. The interruptions in the computation complicate the introduction of real-time data inputs, but otherwise do not place any major restrictions on system capabilities.

Variation of the Machine Variables with Time: The Analog-Computing Period

At the start of the kth computing period $[t = (k-1) \Delta t, \tau = 0]$, the digital component X_D and the analog component X_A of each machine variable X are *reset* to their correct values†. Each digital components X_D remains constant during the entire computing period, while each analog component varies as a function of the computer time τ as dictated by the computing interconnections for the given problem.

The Digital-Computing and Carry-Generating Periods

At the end of the kth analog-computing period, each analog voltage is held and generates a positive or negative carry ($\pm$ 1 m.u. increment) if

$$\left| {}^{k}X_A (a_t \Delta t) \right| > \frac{1}{2} \text{m.u.} \; (k = 1, 2, \ldots)$$

The carries are used as digital $\pm$ 1-bit increments to update the digital components ${}^{k}X_D$, also new digital components ${}^{k+1}X_D$ are computed digitally.

During the same holding period, the positive or negative carry machine units are subtracted from the corresponding analog voltages (which can, therefore, never exceed 1 m.u.); and precise fractional parts of the digital components ${}^{k+1}X_D$ are computed digitally, to be introduced into the next analog computation.

† Note that the actual values of X_A and X_D do not necessarily differ between the end of one run and the beginning of another; they will differ only if a carry is made.

Computing Speed

No machine variable $X = X(t)$ may be allowed to increase or decrease by more than ½ m.u. (½ bit) during any one computing period of T seconds; hence we must scale so that

$$\left|\frac{dX}{dt}\right| \leq \frac{1}{2\Delta t} = \frac{a_t}{T} \text{ m.u./sec}$$

If the computer time τ is to represent t on a 1:1 time scale ($a_t = 1$), then we have $T = \Delta t$,

$$\left|\frac{dX}{dt}\right| \leq \frac{1}{2T} \text{ m.u./sec}$$

We can, in this case, represent a full-scale sinusoid

$$X(t) = 2^n \sin 2\pi f t$$

if

$$f \leq \frac{1}{4\pi T \cdot 2^n} = B_{\text{HYBRID}} \text{ cps}$$

We will call B_{HYBRID} the *full-scale bandwidth* of the hybrid computer. A given full-scale bandwidth B_{HYBRID} requires

$$T \leq \frac{1}{4\pi B_{\text{HYBRID}} 2^n} \text{ sec}$$

The analog computing elements of the hybrid computer must permit the full rate of change

$$\frac{dX_A}{dt} = 2\pi B_{\text{HYBRID}} 2^n = \frac{1}{2T} \text{ m.u./sec}$$

i.e., the analog computing element must be able to produce a full-scale analog sinusoid

$$X_A(t) = \sin 2\pi B_{\text{ANALOG}} t \; ;$$
$$X_A \text{ in m.u.}$$

with

$$B_{\text{ANALOG}} \geq 2^n B_{\text{HYBRID}} = \frac{1}{4\pi T} \text{ cps}$$

Note that increased digital accuracy necessarily requires a proportional increase in the required analog bandwidth once B_{HYBRID} is given.

In the experimental system T was chosen to be 1250 microseconds, corresponding to B_{ANALOG} of about 64 cps and $B_{\text{HYBRID}} \simeq 7.95$ cps. In order to insure ample time for performing digital operations, the analog holding interval, T_H, was made equal to $0.04T = 50$ microseconds. If T_H were zero, the system could keep up with full-scale sine waves of angular frequency 50 rad/sec. Inclusion of T_H, however, reduces this figure to 48 rad/sec. or about 7.65 cps.

The *speed-accuracy ratio* of the hybrid computer is $100/2pT$ distinguishable increments/sec. If we reduce this by $1/25$ to allow for the digital-computing periods, we have $625/13pT = 38{,}500$ distinguishable increments/sec., which permits a crude comparison to modern incremental digital differential analyzers.

Sequence of Operations

a. Reset (Initial Hold)

Prior to the beginning of a computer run, the INITIAL HOLD state (analogous to HOLD in an analog computer) is established by putting the proper digital and analog initial conditions into all integrators. At this time, a particular Total Run Time may be selected, which will stop computation and command solution read-out.

b. Run

When the computer run is initiated, all analog subsystems are made operative for the duration of the first computing interval T. A holding interval, T_H, is then initiated. During T_H, all analog integrators are in HOLD, and the following operations are performed:

1. *Digital Integration* (updating) simultaneously in all integrators, and transmission of *carries* from integrators to other computing subsystems.
2. *Digital Operations* in summers, multipliers, coefficient setters, function generators, and any other *zero-memory devices; transmission of carries.*

Some of these operations may overlap in time, but it is essential that digital updating in all integrators be performed and the necessary carries transmitted to subsequent elements in the computing loop. Digital data transfer will normally be incremental ternary transfer (carry pulse and DC carry sign signal). All digital operations are under the control of a subroutine clock, which can be expanded to drive a large number of digital subsystems simultaneously.

After the digital operations are completed, the new states of the digital system will automatically create step transients in the analog channels, through their effect on various D/A converters. Analog interpolation voltages will also be reset to zero. After any transients subside, another analog computing interval, T, may be initiated.

The above operations are repeated for the desired number of computing intervals. Figure 7 shows how the analog and digital parts of the analog and digital parts of solution combine to form the complete variable.

c. Read-Out

At the end of the desired number of computing steps, the computation is stopped. A high-speed sample-hold system and a digital voltmeter provide a digital display of the analog variable at the read-out time. The digital part of the variable is also displayed.

II. HYBRID INTEGRATION PRINCIPLES

A hybrid integrator implements the operation (assuming $a_t = 1$):

$$X = a \int_0^{mT+\tau} \dot{X}\, dt + X_0$$

Thus

$$X = X_{D0} + a \int_0^{mT+\tau} \dot{X}_D\, dt + a \int_0^{mT+\tau} \dot{X}_A\, dt + X_{A0}$$

$$= X_{D0} + a \sum_{k=1}^{m-1} \overset{k.}{X}_D T + a \int_0^{\tau} \overset{m.}{X}_D\, dt + a \int_0^{mT+\tau} \dot{X}_A\, dt + X_{A0}$$

or

$$X = \left\{ X_{DO} + aT \sum_{k=1}^{m-1} {}^{k}\dot{X}_D \right\} + \left\{ a \, {}^{m}\dot{X}_D \, \tau + a \int_0^{mT+\tau} \dot{X}_A \, dt + X_{AO} \right\}$$

The first bracketed expression is a digital operation with "Digital Value" if $aT = (\frac{1}{2})^i$. (Note that it contains both an integral and fractional part). The second bracketed expression is an analog quantity for general τ, and in general may be greater than 1 m.u.

Scaling

The magnitude and rate-of-change of $X(t)$ and $\dot{X}(t)$ are both assumed to be limited by appropriate scaling:

$$\left|\frac{dX}{d(a_t t)}\right| , \left|\frac{d\dot{X}}{d(a_t d_t)}\right| \leq \frac{1/2}{T} \text{ m.u./sec.} = \frac{1}{2T} \text{ m.u./sec.}$$

$$|X|, |\dot{X}| \leq (2^n - \frac{1}{2}) \leq 2^n$$

$$\leq 7\frac{1}{2} \leq 8$$

By making T small enough, we ensure that the change in the integral of the digital portion of X is less than 1/2 m.u. in machine time $a_t\Delta t = T$ sec. In a time interval T seconds long

$$\Delta X \leq a \int_0^T \dot{X} \, d(a_t t) \leq aT \, |\dot{X}|_{max} \leq \frac{1}{2} \text{ m.u. in T sec.}$$

$$aT \, |\dot{X}|_{max} \leq \frac{1}{2}$$

Or

$$a \leq \frac{1}{2^{n+1}T} \leq \frac{1}{16T} = \frac{1}{16 a_t \Delta t}$$

ensures meeting the scaling conditions above, regardless of $|\dot{X}|$. As an example, with $n = 3$ and $T = 1.25$ ms., $a \leq 50$ is satisfactory.

Method of Approach: "DDA-Plus-Interpolation"

The following method follows closely that of Skramstad (Ref. 13). Figure 1(a) shows the system with incremental digital data transfer.

In general, three operations must take place during the holding interval following the k-th computer run:

1. The value of $aT \, {}^{k}\dot{X}_D$ is added to the lower orders of the R register.
2. The analog input representing $a \, {}^{k}\dot{X}_D \, \tau$ at $\tau = T$ is set to zero to compensate for the addition performed in the above step. These two operations produce no net change in X_A.
3. If X_A exceeds $\frac{1}{2}$ m.u. in magnitude, it must be adjusted by removing or adding 1 m.u. from X_A and correspondingly correcting R. Simultaneously, a $\pm$ 1 m.u. increment is sent to the next computing element (carry operation).

Figure 4 shows the digital portion of the hybrid integrator. All digital variables are represented in a 2's complement code. The input register ($\dot{X}$) contains $\dot{X}_D$, the actual digital portion of the input variable in this code. The initial value of $\dot{X}$ is selected manually; as the computation proceeds, incremental changes in $\dot{X}_D$ are made upon receipt of $S(\Delta\dot{X})$ and $\Delta\dot{X}$ signals from the preceding computing element. During the digital updating period, $\dot{X}_D$ is serially added to the R register; simultaneously, the R-register is corrected by $\pm$ 1 m.u., if there is

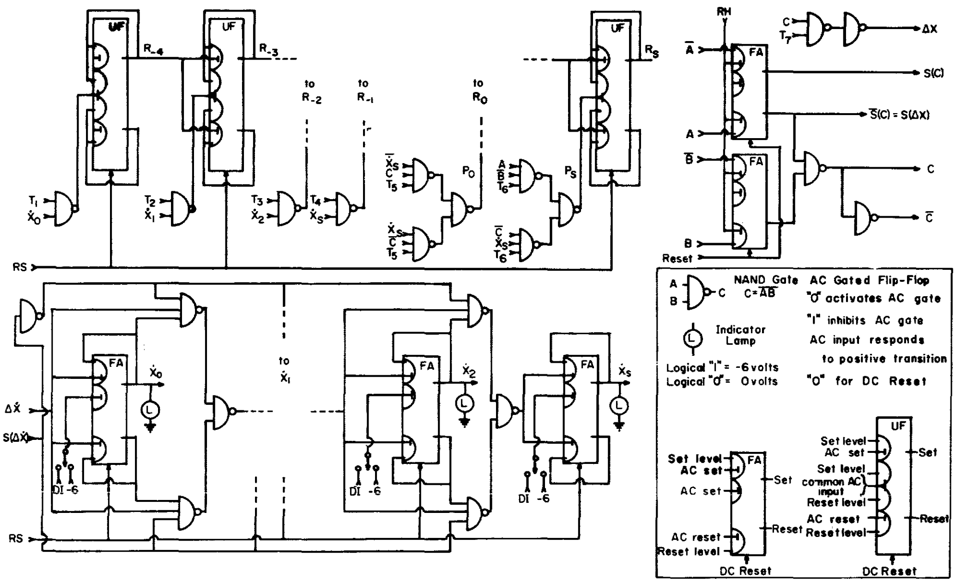


Figure 4. Digital Section.

to be a carry. Note that an increment $\triangle X$, $S(\triangle X)$, is transmitted to the next computing element only if a carry is made. Flip-flops A and B store the states of the Comparators at the beginning of a HOLD interval.

The entire digital operation is completed in about 35 microseconds. During this time, the sawtooth waveforms, which are supplied to the "X" D/A Converter are resetting to zero (this requires about 20 μsec). Allowing another 15 μsec for analog transients to settle, it is possible to use a HOLD interval T_H of 50 microseconds, or about 4 per cent of the computing interval, T. Additional hold time is required for other digital operations associated with such elements as summers and multipliers, since they must wait for the receipt of carries from integrators prior to the initiation of their digital operations. Therefore, the prototype system includes an optional 10 per cent (125 μsec) holding interval, to permit the inclusion of these extra sequences at a later date.

Figure 5 shows the analog system which accompanies the above digital system. Q_1 and Q_2 are used to put the integrator into HOLD, the six-diode bridge is shorted during the RESET period, to insure that the proper initial condition on X_A. Comparator A detects the condition $X_A > 5$ volts, Comparator B detects $X_A < -5$ volts. D/A Converter $\dot{X}$ is a four-bit bipolar unit which provides the interpolation component $\dot{X}_D\,\tau$. D/A Converter R provides the component $E \cdot R_D$. The entire analog system was designed for an overall accuracy of 1 per cent of half-scale. Calibration tests indicated that the errors in the various components of X_A were typically less than 50 mv., i.e., ½ per cent of half-scale (see also Refs. 14 and 16).

Polarity Inversion

The integrator may be operated in the inverted mode by using an analog unity-gain inverter to invert X_A and logically inverting $S(\triangle X)$; this method is, of course, much simpler than using a separate inverting component.

Interpolating Waveforms

The ramp interpolating waveforms used to drive the "$\dot{X}$" D/A Converter are supplied by a separate waveform generator, which can thus service a number of integrators simultaneously. This unit is described in Ref. 15. It provides 1 per cent-linear positive and negative 10 v ramps which are reset to zero in about 20 microseconds after the HOLD interval T_H begins.

Computer Read-Out

The hybrid differential analyzer is equipped

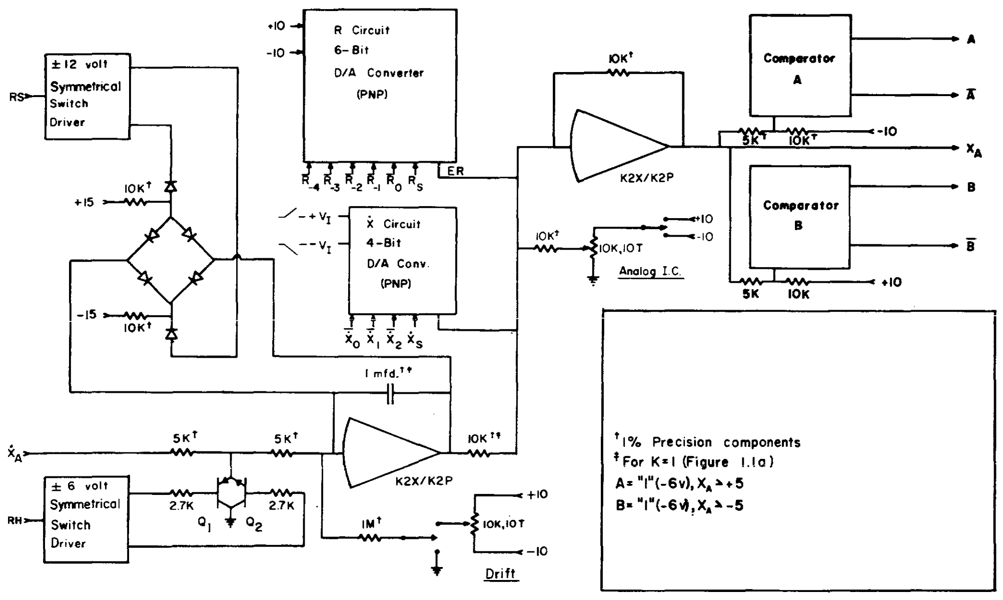


Figure 5. Analog Section.

with a precisely-timed read-out system, which performs the following functions:

a. Upon receipt of a read-out signal, a fast analog sample-hold circuit stores the value of the desired analog machine variable. Simultaneously, a digital voltmeter is commanded to convert this voltage to a digitally displayed value for X_A.

b. During the computer run, the digital portion of the read-out system has been receiving incremental information about the value of a particular digital variable, X_D. When read-out is commanded, the digital system will retain the value of X_D, and display it (in this system as a 2's complement binary number).

c. For diagnostic and display purposes, a full-analog read-out of the sum of both X_D and X_A versus either real time or machine time is provided.

The read-out system is described in Ref. 17. It has an accuracy of better than 25 mv on read-out of X_A.

Control Clock (See also Ref. 6)

The computing sequence is controlled by a digital clock system. This system uses an 80 KC crystal and 1 MC transistorized digital logic modules. It provides the necessary timing signals for precise control of the analog computing operations. A pre-selectable read-out time may be set from 0.01 to 999.99 T.

III. OTHER HYBRID COMPONENTS

This section describes the general features of components for summing, coefficient changing, multiplying, and function generation. These elements perform their digital operations subsequent to the updating of all hybrid integrators. They are all considered to be zero-memory elements, i.e., at any instant, the value of the output variable is related to the instantaneous value of the input variable(s).‡ If several such elements are connected in cascade, the digital and carry generating operations should proceed forward from the first element following an integrator, until all zero-memory elements in a cascade chain have been updated successively.

Summing

Figure 6 shows a block diagram of a summing component to form

‡ Note that this is not quite a simple one-to-one mapping, since each value of a hybrid variable may be represented in two different forms, e.g., $X = 6.3$ can be represented as $X_D = 6$, $X_A = 0.3$ or $X_D = 7$, $X_A = —0.7$.

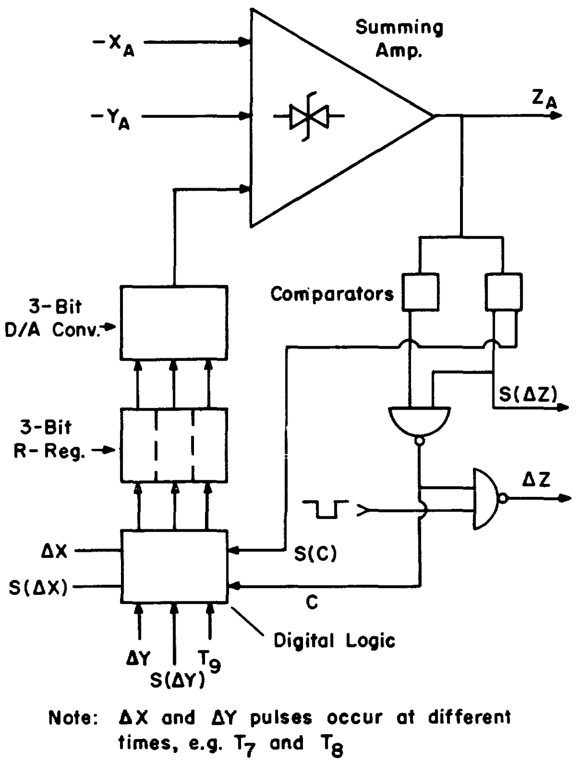


Figure 6. Summing Component.

$$Z(t) = Z_D + Z_A = \frac{(X_D + X_A) + (Y_D + Y_A)}{2}$$

In Ref. 18, it is shown that the essential digital operation is the formation of the digital quantity

$$R = \frac{X_{D0} + Y_{D0}}{2} - Z_{D0} - N + \tfrac{1}{2}\Sigma({}^{m}X_D + {}^{m}Y_D)$$

Note that R is a three-digit number, with a value ranging from $-1\tfrac{1}{2}$ to $+1\tfrac{1}{2}$ m.u.

Summing at Integrator Inputs

Note that digital summation may also be performed at the input to an integrator by providing separate inputs for each input variable. This requires separate carry-transmitting pulses for each incremental digital input. Proper scaling insures that there is no need for additional carry operations within the integrator, since it is the *integral* of the input, rather than the value of the input that affects the integrator output magnitude. A simple digital device for summing variables at the input of an integrator has already been tested successfully (Ref. 19).

Multiplication

The operations involved in hybrid multiplication and coefficient changing are essentially the same, except that in the latter case, the coefficient is a fixed hybrid constant $C = C_D + C_A$. Figure 1 (b) is a general diagram of a hybrid multiplier; it illustrates the basic operations required; i.e.,

$$Z = Z_D + Z_A = \frac{(X_D + X_A)\ (Y_D + Y_A)}{2^n}$$
$$= \tfrac{1}{8}\ (X_D + X_A)\ (Y_D + Y_A)$$
$$= \tfrac{1}{8}\ [X_DY_D + X_DY_A + X_AY_D + X_AY_A]$$

The hybrid multiplier requires a fast analog multiplier to form the term X_AY_A. Note that its accuracy would not have to be high in a hybrid system with a large number of digital bits; indeed, the analog multiplier might even be dispensed with. However, in a four-bit hybrid system, the analog multiplier is required, and it should have an accuracy of better than 8 per cent, to maintain consistent accuracy with the other hybrid components thus far discussed.

Coefficient Changing

The requirements for a hybrid coefficient-changing component follow directly from the above. Some simplifications are now possible. Specifically, the analog multiplier is replaced by a potentiometer, the Y_D register is replaced by a manually-settable group of digital lines and the D/A converter for X_DC_A now receives only constant analog inputs C_A and $-C_A$. The R-register is also simpler.

Generating a Function of One Variable

Considerable prior work has been done in this area, with the object of developing hybrid function generators for use with conventional analog computing systems (Refs. 5, 10, and 12).

Figure 1(c) shows a typical block diagram for a hybrid function generator. The function will in general be generated by performing interpolations about digitally located values of the independent variable, using a Taylor's series approximation. If fewer digital bits are used, then higher-ordered terms in the approximation may be required to achieve a given accuracy. To maintain an accuracy consistent with the other computing elements in a four-bit system, second-order interpolation terms might be required for some functions.

Another added complication in function generators is that digital functions must be formed with sufficient precision to match the *full accuracy* of the analog system.

As in analog computers, function generators should generate corrections to analytical (usually linear) approximations for F(x) whenever possible; *this improves the effective accuracy of even first-order interpolation.*

Summary

All of the above systems are organized on a similar basis. They contain an R-register for forming digital increments of the output variable, which normally contains digital fractional parts of a machine unit and has magnitude strictly less than two. There are also D/A converters or multipliers and conventional analog elements. Each component contains two comparators and associated logic to perform carries by correcting the R-register and transmitting an increment to the next computing element.

IV. EXPERIMENTAL RESULTS

Two hybrid integrators, with associated control and read-out equipment were used in three simple computing configurations. Each problem was scaled to approach the maximum amplitude and rate limitations of the system. Figure 7 shows an analog display of the three problem solutions; also shown are the two solution components, X_D and X_A.

Free-Fall Parabolic Trajectory

The first computer problem was the solution of the differential equation:

$$x(t) = -32.2 \text{ ft/sec}^2;$$
$$X(0) = 0 \text{ ft};$$
$$\dot{X}(0) = 600 \text{ ft/sec}$$

with solution

$x(t) = 600\,t - 16.1\,t^2$ ft.

Maximum Height: 5590.092 ft.

Time to Impact (zero crossing): 37.267 sec.

This equation, of course, simulates an elementary flat-earth vacuum trajectory problem,

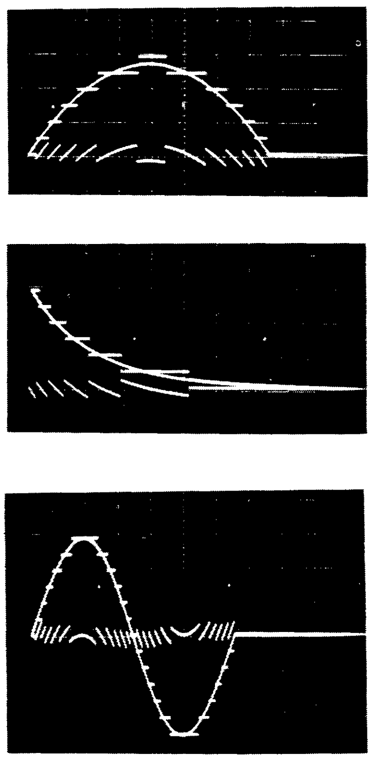


Figure 7. X_D, X_A, and Their Sum.

and is of interest because it involves an open-loop computation, where cumulative errors should become readily apparent. Figure 8 shows a plot of solution error versus "time" in computing intervals. Some of the more pertinent results were:

a. Error in impact time: 0.02 per cent.
b. Error in maximum height: 2 ft. (0.04 per cent of theoretical value).
c. Maximum measured machine error: 70 mv (0.09 per cent of half-scale machine range).

Decaying Exponential

The first closed-loop problem was the simple first-order linear differential equation

$$\dot{X} = -50\,X;\ X(0) = 60 \text{ volts}$$

with solution

$$X = 60\,e^{-50t}$$

Stated in terms of computing intervals, the problem becomes

$$\dot{X} = -\frac{X}{16};\ X(0) = 60 \text{ volts (6 digital m.u.)}$$

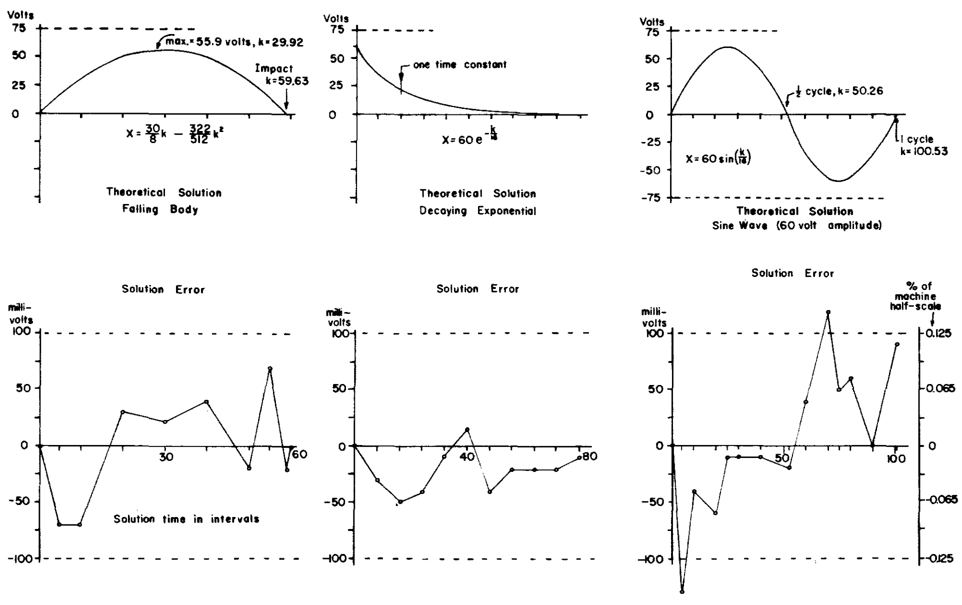


Figure 8. Typical Solution Errors.

with solution

$$X = 60\ e^{-k/16}$$

One Computing Run = 1.25 ms of machine time
One Time Constant = 16 computing intervals

Figure 8 shows a plot of the nominal solution error; it indicates that the maximum error in the solution was 0.05 volt (0.07 per cent of half-scale).

Undamped Sinusoid (Circle or Sine-Loop Test)

Two hybrid integrators and an inverter were used to study the simple undamped second-order differential equation:

$$\ddot{X} = -2500\ X$$

with solution

$$X = X(0)\cos 50\,t + \frac{1}{50}\dot{X}(0)\sin 50\,t$$

Stated in terms of computing intervals:

$$\ddot{X} = -X/256$$

with solution

$X = X(0)\cos k/16 + 16\,\dot{X}(0)\sin k/16$
One Period = 100.26 computing intervals
One Period = 0.1253 sec. of machine time
One Period = 0.1303 sec. of real time (7.67 cps)

Figure 8 shows that the maximum solution error was 0.13 v (0.17 per cent of half-scale), but that the errors were typically much smaller. The average error over one cycle is 0.017 volt, with an rms error of 0.061 volt.

The nominal solution exhibited a slight exponential build-up, which could be measured by observing the solution over several cycles. The amplitude of the waveform increased approximately 0.06 volts/cycle. The natural frequency was low by 0.02 per cent (2×10^{-4}). Thus the overall accuracy of location of the system poles in the complex plane was about 0.025 per cent.

Errors Due to Removal of Sawtooth Interpolation and Analog Integrator

Figures 9 and 10 (a) and (d) show the effect on the computer solutions caused by removing the sawtooth interpolation and/or the analog integrators. In general, the absence of the sawtooth produced less solution error than the absence of the analog integrators. This would be expected, since the digital updating operations tend to reduce the long-term effects of interpolating errors. In the sine loop test, removal of these functions produces a solution with a rapid exponential build-up.

Effects of Artificial Errors in the Analog Channel

Various types of errors were introduced into the analog channels of the integrators. The effects of these errors on the computer perform-

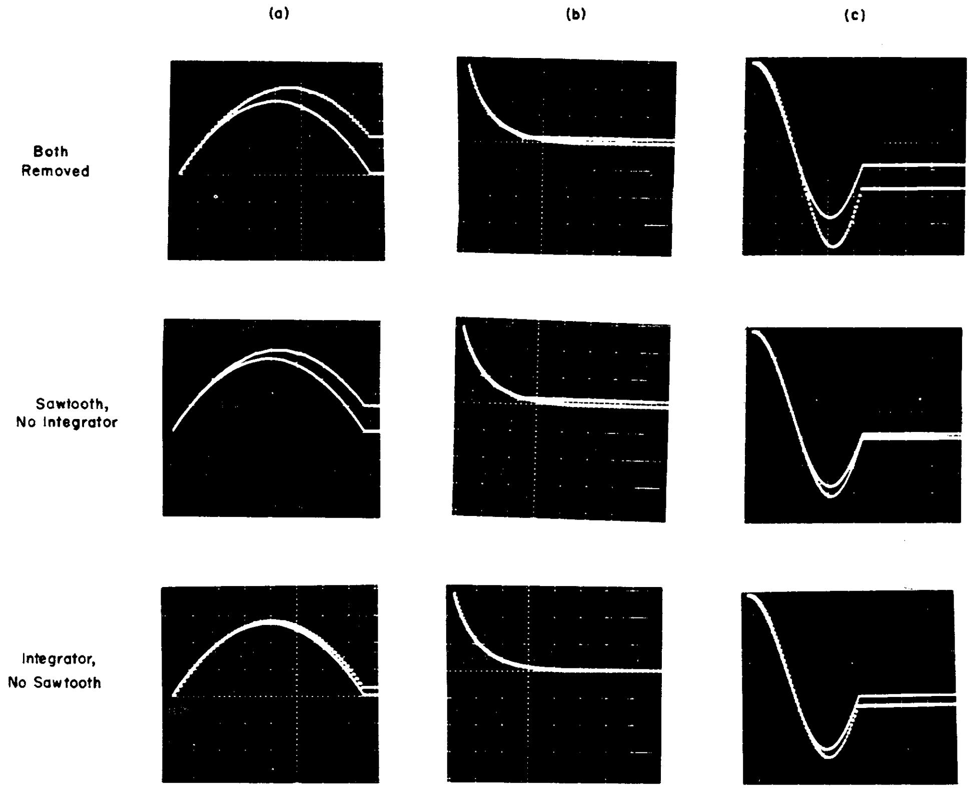


Figure 9. Effect of Removing Sawtooth and/or Analog Integrator.

ance is summarized in Table I and in Fig. 10. In the parabola and decaying exponential problems, Table I summarizes the maximum solution errors. For the sine loop, the results are summarized in terms of frequency/error and damping (exponential build-up).

Effect of the Hold Interval

In addition to the normal HOLD interval, T_H, of 50 microseconds, intervals of 125 and 250 microseconds were also used in several experiments. No noticeable effect on the computer performance was observed.

Comparator Accuracy

The accuracy of the analog comparator is not critical. In fact, since most operational amplifiers will not overload until their nominal computing range is exceeded by perhaps 10 per cent, a comparator accuracy of $\pm 1/2$ volts would suffice.

V. COMPARISON OF DIFFERENTIAL ANALYZER SYSTEMS

The inherent differences among analog, digital and hybrid, differential analyzers make them difficult to compare. This section presents an attempt at such a comparison in terms of a gross accuracy-bandwidth figure-of-merit based on a per cent of half-scale accuracy.

The accuracy-bandwidth figures are based upon the assumption that the total available range of machine variables is utilized, regardless of the type of computer system being discussed. The term "bandwidth," as applied to the speed of computers is related to the maximum slewing rate, or rate-of-change of variables attainable by the computing system.

Consider a system with restrictions:

$$|X(t)| \leq A : |dX(t)/dt| \leq R$$

For sinusoids the maximum full-scale frequency of operation is

$$f_{max} = R/2\pi A$$
$$= \left|\frac{dX/dt}{2\pi A}\right|$$

Analog Differential Analyzers

Figure 12 shows a typical estimate of the accuracy-bandwidth capability of conventional

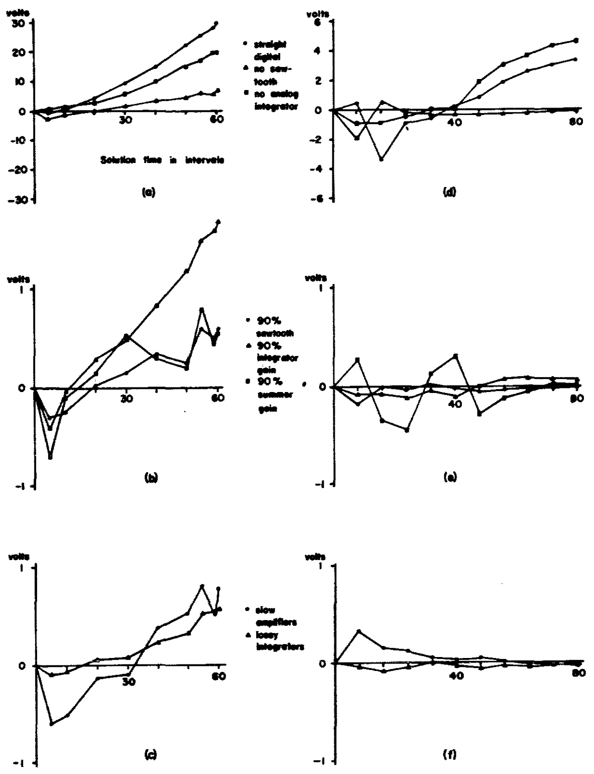


Figure 10. Solution Error Due to Artificial Analog Errors; (a), (b) and (c) are for the parabola; (d), (e), and (f) for the decaying exponential.

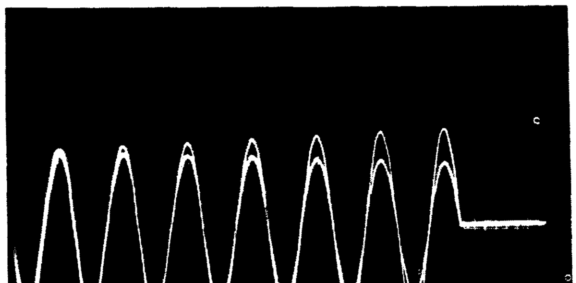


Figure 11. Effect of Summing Amplifier Gain.

analog equipment (see also Ref. 4, Chapter 4 and Ref. 5). To achieve the higher range of accuracies, precision components must be used in a controlled environment. The curve for repetitive analog systems is also shown, since estimates of ultimate hybrid system capabil-

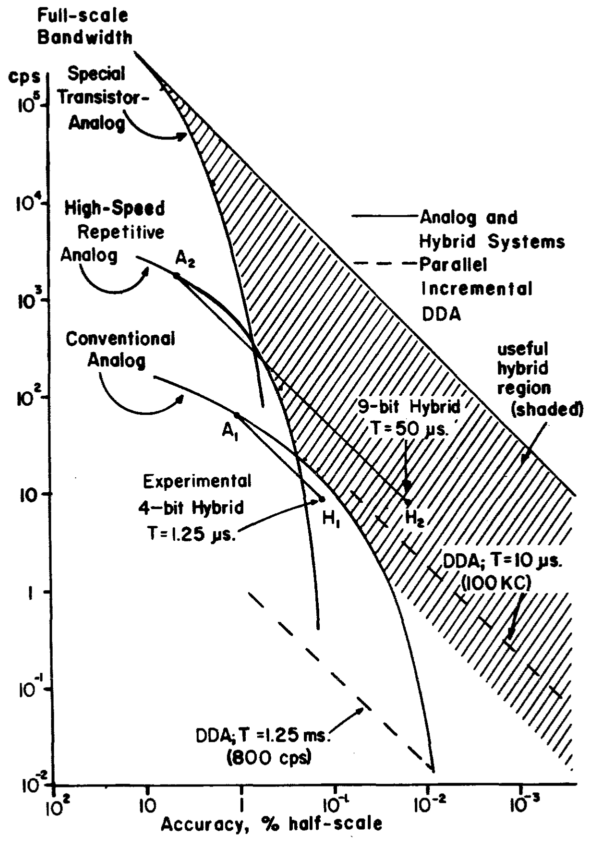


Figure 12. Gross Accuracy-Bandwidth Comparison of Differential Analyzers.

ities should be made assuming the analog elements are similar to those found in repetitive systems. The accuracies shown are grossly related to an absolute *half-scale* accuracy of the computer.

DDA Capabilities

Considerable work has been done on the analysis of errors in digital differential analyzers.[§] Various assumptions have been made which lead to somewhat different results. One point that is easily overlooked is that parallel-organized DDA's almost invariably use incremental data transfer. Moreover, the integrators must operate on "stale" values of the machine variables. Conventional DDA's usually use an open (extrapolative) trapezoidal integration rule. Assuming this type of machine organization, an estimate of DDA capabilities can be made (in the case of linear computing config-

[§] For example, see Refs. 3, 8, and 9. Reference 9 is especially comprehensive.

TABLE I
EFFECT OF ARTIFICIAL ERRORS ON COMPUTER SOLUTION: A SUMMARY
(See also Figures 9 and 10)

Type of Artificial Error	Problem Error				Remarks
	Falling Body (Parabola)	Decaying Exponential	Sinusoid		
None	0.07 v max.	0.05 v max.	0.13 v max. 0.06 v RMS		Normal errors typically 0.1v
Straight Digital	28.7 v. error at impact	3.3 v error in final value	20 v/cycle build-up		Computer errors generally larger when analog integrator is absent, than when sawtooth is absent
Sawtooth, No Integrator	19.9 v	4.53 v	7.5 v/cycle		
Integrator, No Sawtooth	7.3 v	Small Final, Large Initial Errors	3 v/cycle		
Analog Gain Errors			% $\Delta a/\omega$	% $\Delta\omega/\omega$	
90 o/o Summing Amp.	0.8 v max.	0.45 v max.	0.44	0.05	Sawtooth accuracy generally less important than amplifier and integrator gain
90 o/o Integrator Gain	1.7 v max.	0.16 v max.	0.30	0.065	
90 o/o Sawtooth	0.6 v max.	0.17 v max.	0.30	0.03	
Poor High-Freq. Resp.	0.79 v max.	0.34 v max.	0.07	0.02	Good High-Frequency Response more important
Finite Integrator Time Const.	0.57 v max.	0.09 v max.	0.01	0.055	

urations) which may be used for comparison of the DDA to other types of differential analyzers.

DDA's with Incremental Transfer

Most modern commercial DDA's use open *trapezoidal integration* and *incremental* data transfer. The results of the analysis shows that round-off errors will predominate in this type of system.||

and

$$\frac{\text{error}}{\text{frequency}} \simeq 200\pi\, T_I \ \%\ \text{cps};\ (T_I \text{ small})$$

where T_I is the iteration time. Equivalently, DDA's of this type have a capability of computing approximately $1/T_I$ distinguishable increments-per-second.#

Figure 12 shows the gross accuracy-bandwidth capability for a *parallel-organized incremental DDA*, using trapezoidal integration. The

|| This conclusion does not hold for general-purpose digital computers, where total transfer is used. Truncation errors may predominate when general-purpose machines are used as differential analyzers, depending upon the type of integration rule used.

# This simple result has been pointed out previously by Korn (Ref. 5).

machine is assumed to have an integration time of T_I seconds. This figure illustrates how the performance ranges of analog and digital systems overlap, and also indicates roughly the regions where one or the other might be considered distinctly superior.

Hybrid Differential Analyzers: Theoretical Performance

The hybrid differential analyzer can be fitted into the above analysis quite easily. Briefly, one can estimate the gross accuracy-bandwidth capability of a hybrid system as follows:

Given the analog computing interval T, estimate the full-scale bandwidth

$$B_{ANALOG} = 1/4\pi T$$

Pick the point on the estimated accuracy-bandwidth curve for the analog system corresponding to B_{ANALOG}. From this point draw a line downward along a slope corresponding to a constant accuracy-bandwidth product (45° slope in Figure 12). The length of the line depends upon the number of digital bits. If the system uses n bits plus sign, the line should correspond to a reduction in bandwidth and an increase in half-scale accuracy of 2^n. For a given T, the rate-scaling limitation precludes computation at higher speeds. In Figure 12, point A_1 corresponds to a 1 per cent analog system, with T = 1250 microseconds. Point H_1 then shows the predicted accuracy of a four-bit hybrid system, which should be attainable at all computing frequencies below $B_{HYBRID} = B_{ANALOG}/2^n$. In this example, B_{HYBRID} is about 8 cps, and the estimated error is correspondingly about ⅛ per cent of half-scale (0.125 volts for E = 10).

As pointed out earlier, the analog HOLD intervals, T_H, reduce the effective real-time hybrid computing speed by a factor $T/(T + T_H)$; in this case, this factor is approximately 0.96, so that the effective real-time value of B_{HYBRID} is about 7.67 cps.

Point H_2 on Fig. 12 shows the estimated capability of a hybrid system using 9 bits (n = 8) with a 50 microsecond computing interval and a 5 per cent-accurate analog channel. Such a system is technically feasible using modern wideband transistorized amplifiers, and would provide an estimated accuracy of better than 0.03 per cent of half-scale at a maximum B_{HYBRID} of about 6.2 cps. Note that an equivalent incremental DDA would have to operate at a minimum iteration rate of about 300,000 per second.

VI. CONCLUSIONS: PROJECTED APPLICATIONS AND FUTURE STUDIES

The experimental results verify the theoretically predicted accuracy of the prototype hybrid system, and demonstrate the feasibility of this type of computing technique. It is felt that the results justify the concept that the accuracy of the hybrid system can be directly improved by increasing the number of digital bits, as originally predicted.

In linear computing configurations, it appears that the location of the system roots can be established with an accuracy of better than 0.1 per cent. The long-term accuracy of the integration process appears to be relatively insensitive to small errors in the analog system components. The digital portion of the system preserves the static gain of the integrators with surprising precision, and the effect of errors in the analog elements normally appears as a phase shift rather than a gain error. The effects of random variations in the static values of the various computing elements (precision resistors and capacitors) thus would tend to cancel in a large system.

Based on the parameter-influence studies described above, some general conclusions about the different sections of the hybrid integrator are suggested:

a. It is important that the summing amplifier gain be accurate; also, its bandwidth should be adequate to insure negligible phase shift at the maximum computing speed.
b. It is important that the gain and high-frequency response of the analog integrator also be accurate; however, the DC or low-frequency response is not critical. This suggests that a passive RC network using precision elements could be used in place of the active operational integrator, particularly for systems with a short analog computing interval and modest analog accuracy requirements. *Note that this could save a complete d-c amplifier per integrator.*

c. The accuracy of the hybrid integrator is less sensitive to errors in the sawtooth interpolation channel than it is to errors in the summing amplifier and analog integrator. This suggests that a hybrid system utilizing a large number of digital bits (e.g., $n > 8$) would still operate well with only 5-6 bit D/A converters in the interpolation channel.

d. Simple two- or three-transistor comparators could replace the seven-transistor units used in the prototype system.

e. Since the analog parts of the hybrid machine variables normally have an average value close to zero, long-term overloading of the analog integrators is not a serious problem, except in certain pathological problems. Integrator drift affects the computer solution in a manner quite similar to drift in analog systems but is divided by 2^n.

f. The use of a HOLD interval, T_H, successfully eliminates the problem of accommodating the analog transients which accompany the digital operations. The HOLD period may be made an appreciable fraction of the RUN time, T, without noticeably affecting the solution accuracy. This permits using relatively slow serial digital arithmetic schemes.

Future Areas of Development: Projected Capabilities

This work has been primarily a feasibility study, with experimental verification of the basic operating characteristics of the hybrid integrator. Development and testing of additional prototype summing, coefficient-setting, multiplying and function-generating elements would be useful in deriving further information about this type of system.

Through the use of high-speed transistorized operational amplifiers, the hybrid differential analyzer system described in this work could be given a substantially increased accuracy-bandwidth capability. It would be feasible to operate with an analog computing interval of perhaps 100 microseconds. At this rate, timing errors would become more of a problem. Nevertheless, an accuracy of perhaps five per cent of half-scale could easily be maintained in this type of application. This would permit adding more digital bits, while maintaining reasonable computing speed. For example, with $n = 8$, $T = 50$ microseconds, $T_H = 5$ microseconds, and a five per cent analog accuracy, it would be possible to achieve a hybrid computing accuracy of perhaps 0.02 per cent of half-scale, while retaining a real-time computing bandwidth of over five cps. Note that such a system would utilize even cruder analog components than the present system, particularly in the multiplier and function generator. Moreover, it could be expected to maintain its accuracy without *any* periodic adjustment or calibration, and without careful environmental control. A parallel-organized incremental DDA using trapezoidal integration would have to operate at an integration rate of at least 300,000 per second to achieve a similar accuracy and speed.

Applications

Obviously, a hybrid differential analyzer is a special-purpose computing system, and one may legitimately ask where useful applications for a hybrid system might arise.

One specific area could be the solution of differential equations associated with the trajectories, orbits, and impact "footprints" of space vehicles. Such calculations usually require nonlinear operations, including division, coordinate transformations, and the use of nonlinear functions to represent effects of gravitational corrections, atmospheric drag, etc. In many cases, a moderately accurate (0.01-0.05 per cent) computing system might be adequate, particularly if it had a computing speed much faster than real-time.

The use of hybrid techniques might permit increased computing speed in such situations; more important, it should be considerably easier to program a hybrid machine to insure a given degree of accuracy.

Conclusions

As a final conclusion, the results of this study confirm that the predicted capabilities of this type of hybrid differential analyzer system can be achieved in practice, and such a system can be used in applications where moderately high-accuracy real-time and faster-than-real-time computing is required. A more conclusive judgment of the practical advantages of this system can be made after the capabilities of hybrid multipliers and function generators have been studied. The outlook is encouraging at this point.

ACKNOWLEDGEMENT

This paper is based upon a Dissertation submitted in partial fulfillment of the degree of Doctor of Philosophy at the University of Arizona.

The author must first of all express gratitude to Professor G. A. Korn for his guidance and inspiration throughout this study. Messrs. R. L. Maybach and T. A. Brubaker provided invaluable assistance in the design of the master timing clock and read-out systems. Messrs. R. L. T. Hampton, B. A. Mitchell, E. P. O'Grady, and H. R. Eckes assisted in the development of special solid-state circuits.

The author is also grateful to the Office of Aerospace Research, Information Research Division, Air Force Office of Scientific Research and to the Office of Space Sciences, National Aeronautics and Space Administration for their continuing support of this study under joint grant AF-AFOSR-89-63; and to Drs. T. L. Martin, Dean of Engineering and P. E. Russell, Head, Electrical Engineering Department, for their encouragement and contribution of University facilities.

LIST OF REFERENCES

1. BRADLEY, R. E., and GENA, J. F., *Design of a 1 MC Iteration Rate DDA*, Hazeltine Technical Development Center, Inc., Indianapolis, Indiana, circa 1961.
2. CONNELLY, M. E., "Real-Time Analog-Digital Computation," *IRE Trans. PGEC*, Vol. EC–11, No. 1, Feb., 1962, pp. 31–41.
3. GSCHWIND, H. W., "Digital Differential Analyzers," *Electronic Computers*, Edited by P. Von Handel, Englewood Cliffs, New Jersey: Prentice-Hall, 1961.
4. KORN, G. A., and KORN, T. M., *Electronic Analog and Hybrid Computers*, New York: McGraw-Hill, in preparation.
5. KORN, G. A., "The Impact of the Hybrid Analog-Digital Techniques on the Analog Computer Art," *Proc. IRE*, Vol. 50, No. 5, May, 1962 Anniv. Issue, pp. 1077-1086.
6. MAYBACH, R L., O'GRADY, E. P., and WAIT, J. V., "A Master Control Clock for a Hybrid Differential Analyzer," Dept. of Elec. Engr. ACL Memo No. 81, Univ. of Arizona, 1963.
7. MITCHELL, J. M., and RUHMAN, S., "The Trice—A High Speed Incremental Computer," *IRE Nat. Conv. Record*, 1958, Pt. 4, pp. 206-216.
8. NELSON, D. J., *A Foundation for the Analysis of Analog-Oriented Combined Computer Systems*, Radioscience Laboratory Report TR 1002-1, SEL-62-069· Stanford, California, April 1962.
9. PALEVSKY, M., "The Digital Differential Analyzer," *Computer Handbook*, Edited by G. A. Korn and H. D. Huskey, New York: McGraw-Hill, 1961, Chapter 19.
10. SCHMID, H., *High Accuracy, Single-Variable, Linear Segment Hybrid Function Generator*, Link Aviation Report DLR 558, Binghamton, New York, Oct., 1960.
11. SCHMID, H., "Combined Analog-Digital Computing Elements," *Proc. West. Joint Comp. Conf.*, May, 1961.
12. SCHMID, H., "Linear-Segment Function Generator," *IRE Trans. PGEC*, Vol. EC-11, No. 5, Dec., 1962, pp. 780-788.
13. SKRAMSTAD, H. K., "A Combined Analog-Digital Differential Analyizer," *Proc. East. Joint Comp. Conf.*, Dec., 1959, pp. 94–100.
14. WAIT, J. V., and MITCHELL, B. A., "A Simple Solid-State Digital-to-Analog Converter for Hybrid Computing Systems," Dept. of Elec. Engr., ACL Memo No. 61, Univ. of Arizona, Feb. 10, 1963.
15. WAIT, J. V., "An Interpolation Waveform Generator for Use in Hybrid Computing Systems," Dept. of Elec. Engr., ACL Memo No. 64, Univ. of Arizona, Feb. 11, 1963.
16. WAIT, J. V., and HAMPTON, R. L. T., "A Solid-State Analog Comparator for Hybrid Analog-Digital Computers," Dept. of Elec. Engr., ACL Memo No. 63, Univ. of Arizona, Jan. 20, 1963.

17. Wait, J. V., "A Read-Out System for a Hybrid Differential Analyzer," Dept. of Elec. Engr., ACL Memo No. 80, Univ. of Arizona, May, 1963.

18. Wait, J. V., "A Hybrid Analog-Digital Differential Analyzer System," Dept. of Elec. Engr., ACL Memo No. 76, Univ. of Arizona, July, 1963 (Ph.D. Dissertation).

19. Wait, J. V., and O'Grady, E. P., "Simple Integrator-Input Addition of Hybrid Variables," Dept. of Elec. Engr., ACL Memo No. 82, Univ. of Arizona, August, 1963.

REVIEW AND SURVEY OF MASS MEMORIES

L. C. Hobbs
Hobbs Associates
Corona del Mar, California

INTRODUCTION AND HISTORY

Prior to the advent of electronic digital computers, large files of data and records were stored primarily in printed form or in punched cards for use with standard tabulating and business machine equipment. The introduction of electronic digital computers in the late 1940's, and their commercial applications in the early 1950's, led to the requirement for storing (in a machine readable code and media) large volumes of data generated by computers that were expected to be used again by the computer. These large external files were stored primarily in punched cards and on magnetic tapes. Until approximately 1955, these serial off-line storage devices provided the only method of storing large volume files of computer records that were to be used by the computer again at a later time. Their applications suffered from the inherent disadvantages of serial access and lack of on-line availability, under computer control, of all records in the file.

The recognition of these shortcomings and the need for a large capacity, semi-random access, on-line file storage under computer control led to the development of large magnetic drums, devices utilizing multiple short loops of magnetic tape, and early efforts in the development of magnetic discs. The large magnetic drums offered relatively fast access times in the milliseconds, but they had limited capacities in the order of one or two million bits per unit and relatively high costs per bit of storage. The magnetic tape loops offered larger capacity and lower costs per bit, but at the expense of significantly longer access times.

The shortcomings of both techniques turned the industry's attention to stacks of magnetic discs as a means of combining capacities in excess of 20 million bits with access times of a few hundred milliseconds at a reasonable cost per bit. The first disc unit constructed by the Bureau of Standards consisted of a donut shaped array of stationary discs with a head mechanism moving around the center of the donut (through a gap in each disc) and stopping at the selected disc. This disc was then accelerated and spun past the head for reading. After the demonstration of this unit in 1952, a number of organizations worked on various types of multiple magnetic disc arrays.

The first one introduced commercially was the RAMAC by IBM in 1956. This unit consisted of a stack of 50 discs rigidly mounted to a shaft rotated at 1200 rpm and providing a capacity of 5 million alphanumeric characters. Two heads were mounted on an arm that moved up and down on a post parallel to the shaft of the rotating disc stack. After this arm was positioned opposite the selected disc, the arm holding the heads was inserted into the disc stack straddling the selected disc so that one head could read a track on the upper surface, and the other head could read the corresponding track on the lower surface. In this way, it was possible to position the pair of heads up and down to one of 50 discs and in and out to one of 100 pairs of tracks on the selected disc.

The next unit made commercially available in 1958 was the RANDEX drum by Remington Rand. This unit, which was a derivative of the large drum developed for the LARC System, stored seven million alphanumeric characters on two large drums rotating on parallel shafts. A head mechanism moved laterally between and along the surface of the two drums positioning the heads to read a set of tracks on each drum.

It is unlikely that large drums and disc files would have progressed farther than the large fixed-head drum with its limited capacity and high cost per bit had it not been for the development of the floating head. Different versions of the floating head were developed somewhat independently by several organizations, but most of them basically involved mounting and positioning the head very close to the surface by a loading force so that it effectively floats on an air slider bearing.[1, 2] This maintains a relatively constant and close spacing between the head and the disc. Since the head to surface spacing is self adjusting, the use of a floating head permits three significant advantages to be realized in mass memories:

1. The necessity for close mechanical tolerances is greatly alleviated. Thus, it becomes feasible to move the head without worrying about mechanically repositioning it within microinches with respect to the surface.
2. The head will follow eccentricities in the surface of reasonable magnitudes. Thus, the requirements for maintaining surface uniformity as the media rotates are greatly alleviated whether these be caused by bearings on a drum or by warpage in a disc.
3. Much closer head to surface spacing also results from the ability of the head to follow surface eccentricities. Thus a significantly higher bit density can be realized in each track. This in turn permits either an increase in the capacity of the device or a reduction in the number of heads, tracks, or discs to be accessed.

The use of a nickel-cobalt metallic coating has significantly increased the reliability, life, and capacity of some drum units. In the past, the plating process has tended to be an art rather than a science. As a result, this type of coating has only recently been used on commercially available disc files. The oxide coatings most commonly used are perhaps easier to apply and the techniques better known, but they are thicker and do not have the wear characteristics of the metallic surface. Nickel-cobalt surface thicknesses at least an order of magnitude less than those of oxide films can be achieved readily. The thinner recording surface permits higher bit densities and consequently larger capacities. With an oxide coating, there is also a greater likelihood of permanently damaging the surface. Although considerable progress has been made in developing harder oxide coatings, it appears likely that the widespread use of nickel-cobalt metallic surfaces will further enhance the reliability and capability of future mass memories.

A number of magnetic drum memories have used phase recording techniques in the past. There now appears to be an increasing tendency to use this type of recording in mass memories. The combination of phase recording and self-clocking techniques may well permit significant increases in capacities for all present types of mass memories using moving-magnetic-surface media for storage.

In both the RAMAC and RANDEX units, head positioning was used to minimize the number of heads and the electrical switching of tracks to provide a larger storage capacity at a lower cost than would have been possible otherwise. This general principle has been carried forward to most of the present day mass memories although other present day approaches involve moving individual magnetic cards. All of the past and present commercially available mass memories involve mechanical motion of magnetic-surface media. For a long time, industry has dreamed of a non-moving static memory that would provide the larger capacity required of mass memories with the faster access times and higher reliability inherent in an all electronic approach. As we will see later, there is some work on such devices underway at this time. Although we hope for a long range solution to this dream, practical commercially feasible equipments are probably many years in the future.

DEFINITION OF MASS MEMORY

At this point in our discussion, we should clarify exactly what we mean when we speak of a mass memory. In the sense in which the term "mass memory" is used in this paper and in the subsequent papers in this session, we are referring to an external storage device that can provide a large capacity and fast semi-random access, that is under direct on-line control of the computer, that is addressable by the computer (although not necessarily by individual word), and that is erasable and reusable.

The term external is used to differentiate the mass memory from the main high-speed internal memory of the machine. To qualify as a mass memory, we would probably expect a device to provide a storage capacity in excess of 10 million bits. Mass memories provide semi-random access in the sense that relatively fast access can be made from any location to any other randomly chosen location without directly passing over all of the intervening records. If the advantages of fast access are to be utilized, it is necessary that the memory be operating on-line and under the direct control of the computer without the necessity for manual intervention. The concept of being directly controllable by the computer also implies that blocks of information in the mass memory must be addressable by the computer although it would be permissible to have individual words or records within the block selected after transferring the block into the machine's internal memory.

This definition represents the general usage in the field although it is occasionally stretched by manufacturers to cover one of their devices that doesn't really meet these criteria. It eliminates, for different reasons, devices such as magnetic tape units, photographic storages, and high-speed magnetic core matrices.

THE NEED FOR MASS MEMORIES

The intensive development efforts in mass memories have been initiated and sustained by a definite need in business, scientific, and military applications. In addition to fulfilling these needs, mass memories offer other advantages that may not be generally recognized. Undoubtedly, others will appear that have not yet been considered. A brief review of some of the major applications and uses for mass memories will serve to better channel our consideration of the performance requirements and the advantages and disadvantages of the different types of mass memories under consideration. Figure 1 summarizes a number of applications and uses that will be discussed as examples.

In general, the use of mass memories can be divided into two major categories—those in which mass memories are required by the nature of the problem, e.g., random interrogations; and those in which mass memories serve to facilitate the processing but in which the requirements of the problem could be and have been met in other ways, e.g., processing of individual entries against two files that are sequenced differently.

A. *BUSINESS APPLICATIONS*

1. Random Interrogations and Look-ups

In many business applications, it is necessary to be able to locate information in large files to answer inquiries that occur at random relative to the file sequence. Policy files in insurance companies are an example.

2. Processing Individual Entries Against Two Files Sequenced by Different Criteria.

Some applications require processing each input transaction against more than one file. For example, in a payroll and labor cost distribution problem, it is necessary to process each time entry for each employee first against the payroll file sequenced by employee number and then against the labor-cost-distribution file sequenced by job or work order number. The use of a serial file, such as magnetic tapes, would require sorting the input transactions into different sequences and making separate processing runs for each file. With a mass memory, the input transactions can be processed against both files in a single pass with no sorting.

3. Random Processing (Psuedo Real-Time)

Applications such as department store customer accounts receivable or airline reservation

BUSINESS APPLICATIONS
- RANDOM INTERROGATIONS AND LOOK-UPS
- PROCESSING ENTRIES AGAINST MULTIPLE FILES
- RANDOM PROCESSING
- STORAGE OF LARGE TABLES

SCIENTIFIC APPLICATIONS
- DATA STORAGE
- MULTIPLE PROGRAMS AND SUBROUTINES
- MEMORY DUMPS

MILITARY
- INTELLIGENCE
- DISPLAY GENERATION
- RANDOM PROCESSING

REAL-TIME
- HISTORICAL DATA
- ALTERNATE PROGRAMS FOR INTERRUPTS
- MEMORY DUMPS

DATA STORAGE AND RETRIEVAL
- INDEXES
- CROSS REFERENCES
- ABSTRACTS

COMMUNICATIONS
- MESSAGE SWITCHING
- STORE-AND-FORWARD
- PRIORITY SEQUENCING

PROGRAMMING
- MULTIPLE PROGRAMS AND SUBROUTINES
- MEMORY DUMPS
- STORAGE OF COMPILER AND LIBRARY
- AID IN COMPILING OPERATION

Figure 1. Summary of Mass Memory Applications.

systems require processing individual transactions as they occur—while the customer waits in these two examples. Storing the file records in a mass memory eliminates the need for batching, sorting, and sequential processing and permits entries to be processed as they occur.

4. Storage of Large Tables

In some applications such as those found in transportation and public utility companies, it is necessary to store very large tables of rates or other types of information. Somewhat related applications are the storing of catalogue information and indexes and cross references such as the telephone directory.

B. *SCIENTIFIC*

1. Data Storage

Many sophisticated and complex scientific applications, such as the inversion of very large matrices, require the storage of more data than can reasonably be handled by the machine's internal memory. Consequently, these types of problems are frequently segmented with the majority of the data stored in the mass memory and brought into the internal memory in blocks for processing.

2. Storage of Multiple Programs and Subroutines.

Frequently, scientific problems are handled on a job shop basis with a large number of relatively short programs being run consecutively and independently of one another. Maintaining the programs for these problems in mass memory reduces the "turn around time" and permits running such problems in random sequence by eliminating the need for selecting and sequentially accessing the programs on a reel of magnetic tape.

3. Memory Dumps

Requirements for dumping the memory contents with the ability to recall them rapidly is an important part of many types of applications other than scientific applications. For example, memory dumps play an important role in error-correction and in debugging new programs. However, the requirements in high-speed scientific applications are more critical. It is frequently necessary to be able to dump the contents of different portions of the internal memory at different times. In such cases, the semi-random access capability of mass memories is very significant in permitting the rapid recall of the information as required. Examples are multi-programming type operations where the computer may be working on several different programs at the same time and applications where a long problem with a relatively low priority may be interrupted at any time to run shorter problems with higher priority. Under these circumstances, a mass memory for dumping the storage contents representing the current status of the running program offers significant time savings over the use of magnetic tapes. If the shorter program could itself be interrupted to run a still higher priority one, the use of a mass memory is even more advantageous.

C. *MILITARY*

In certain military applications, mass memories are required to store special data, such as intelligence information, to store data for generating displays upon call and to permit random processing in applications such as air traffic surveillance and control. Other military applications (e.g., personnel and logistics) involve problems very similar to those of business data processing or scientific applications.

D. *REAL-TIME*

Real-time applications of computers require the ready availability of historical data and of alternate programs and subroutines required for handling many different interrupt conditions. When external conditions arise that cause an interrupt of the computer's normal program sequence, it is necessary to very rapidly obtain the program for processing the interrupt on a real-time basis. A memory dump may also be required with a later recall to recreate the conditions prior to the interrupt. In a large system with multiple interrupt lines, a mass memory permits the required rapid access to any one of a large number of programs or subroutines where the total is too extensive to be stored in the internal memory. This type of real-time operation is commonly found in military command and control systems and will probably be found to an increasing extent in industrial applications, such as process control, when these become more sophisticated and all encompassing.

E. *DATA STORAGE AND RETRIEVAL*

A comprehensive data storage and retrieval system usually requires a mass memory even though the great volume of information might be stored in some other media such as printed documents, microfilms, etc. The mass memory permits rapid indexing, cross referencing, and, perhaps, abstracting.

F. *COMMUNICATIONS*

The use of computers and data processing equipment in large communications systems for message switching and store-and-forward applications is increasing rapidly. These applications require mass memories to permit the system to operate on-line in controlling a multiplicity of independent communication channels with many different sources and destinations. With a mass memory available, messages from the individual channels can be accepted and stored as they are received and later routed and forwarded in the proper sequence to the desired destinations with a priority system superimposed.

G. *PROGRAMMING*

There are a number of programming uses of mass memories that overlap the different types of applications considered previously. For example, the storage of multiple programs and the processing of interrupt conditions have been discussed. In addition to those types of uses, mass memories can facilitate and speed up automatic programming operations by permitting rapid access to the compiler program and the library of subroutines any time a new program is to be compiled. Their semi-random access capability can also be advantageously utilized within the actual compiling operation—e.g., storing lists generated in the compiling process.

Each of the applications and uses discussed above has characteristics that place somewhat different requirements on the mass memory. The relative importance and optimum combination of capacity, cost, access time, data transfer rate, and addressing and buffering techniques may be different for each of these uses. It is unlikely that a single mass memory design would satisfy the requirements for all of the different types of applications and functions discussed. In the following sections, the major types of mass memories and their characteristics are considered and briefly related to the requirements of some of these applications.

TYPES OF MASS MEMORIES

Mass memories can be divided into two major categories—those that involve a moving magnetic surface and those that do not. All of the present commercially available mass memories utilize a moving magnetic surface. Those that do not utilize a moving media offer promise for the future but are still in the development stage at this time. The characteristics of the major types of mass memories are summarized in the table shown in Figure 2. The values shown were chosen as being typical of each type of unit. In some cases, certain characteristics of an individual device may vary significantly from the values shown. The woven screen memory is included in Figure 2 as an example of a type of static mass memory that may be available in the future.

For military applications, other characteristics, such as mobility, weight, maintainability, and ruggedness, may be more important in the final selection between units than the characteristics summarized in Figure 2.

The characteristics with which we are primarily concerned include capacity, cost, average access time, data transfer rate, and addressing techniques. Some of these are difficult to compare because of the different physical characteristics of the devices. For example, a large magnetic drum with a head for every track will have a continuous data transfer rate equal to the instantaneous transfer rate if the heads are switched electronically. However, the continuous data transfer rate for a disc file with moving heads will be significantly greater than the instantaneous transfer rate due to the necessity for interrupting data transfer while moving the head from one position to the next. Similar differences on a more detailed level exist between different devices of the same type.

In comparing costs, a detailed investigation is usually required to determine whether prices quoted for different units include comparable electronics (i.e., controllers, buffers, switching, amplifiers, etc.). The estimated costs shown in Figure 2 are user's costs (rather than manufacturing costs) and assume a moderate amount of associated electronics.

Access time offers another illustration of the difficulties of comparing different types of mass memories. It is difficult to compare the access times even for different devices of the same type —for example, different designs and makes of disc files. It is considerably more difficult to compare the access times for completely different types of mass memories due to differences in the methods of making mechanical access. The total mechanical access is usually made up of a number of separate components. For example, in one type of disc file mechanism it is necessary, when addressing a new location, to release a mechanical interlock, extract the head mount from between two disc surfaces, move it parallel to the stack of discs, insert it between another pair of discs, mechanically interlock the mount in its new position, and then wait for the desired location around the circumference of the selected track. Each of these specific mechanical motions requires a certain amount of time. Some of them also depend upon the location of the new address relative to the previous one. If the new address is on the same disc as the old one, two of the motions are eliminated completely. It may even be possible to read the new track with a different head on the same arm without repositioning it.

Obviously, comparing the access time for this type of unit with one that has a head for each disc or with one that has a head for each track would require a precise and generally accepted definition of "access time." For practical considerations on the part of a user, it could also depend upon the way in which the problem is organized or the purpose for which the unit is to be used. Several years ago in preparing the 1956 IRE Computer Glossary, we tried to tackle the problem of defining access time. We very quickly came to the conclusion that the only universally valid definition would be, "access time—that time which is faster in our machine than in our competitors". In the mass memory area, this is particularly true of "average" access time since it depends on just what is being averaged. Theoretically, at least, it could be the average time to access any randomly selected location from any other random location. In practice, there is a tendency to include the assumption that the user or programmer would organize his problem in a way to assure that certain disastrous times do not occur.

As a result of problems such as these, comparison tables such as the one shown in Figure 2, and more detailed comparisons that might be made of specific manufacturers' devices of each type, present at best a gross comparison. In selecting a device for any specific application, it would be necessary to go into a more detailed comparison of the specific peculiarities and quirks of each of the leading contenders as they relate to that application if a proper decision is to be made. Unfortunately, it is not possible within the time limit and scope of this paper to go into these individual differences between the units of different manufacturers.

A. *MOVING-MAGNETIC-MEDIA TYPES OF MASS MEMORIES*

The major devices in this category are those involving short tape loops, large magnetic drums, magnetic discs, and magnetic cards.[3,4,5,6] Those using short tape loops have an excessively long access time that prevents their being seriously competitors with other recent mass memories for most applications; hence, they will not be considered further here. Each of the other types will be discussed briefly.

1. Large Magnetic Drums

Until recently, the capacity of large magnetic drums ranged from approximately 200,000 to 1,000,000 characters per unit for those with fixed heads and approximately 4 to 10 million characters for those with moving heads. However, one manufacturer recently announced a large dual drum unit with moving heads providing a capacity of 65 million alphanumeric characters.[3] In this unit, two very long drums (over six feet) are rotated on parallel centers with the surfaces close enough to one another to permit a single access mechanism to position sets of 64 heads—32 on each drum.

Average access times have been in the order of 15 milliseconds for the fixed-head drums and 100 milliseconds for the larger moving-head drums. The choice between these two types of mass memories depends largely upon whether access time or capacity is the more important consideration. The fixed-head drum also implies a higher cost per bit of storage due to the number of heads and the switching circuitry required.

2. Magnetic Disc Files

A magnetic disc file consists of a stack of disks (usually in the order of 5 to 100) rotating on a common shaft. The discs are usually between 1½ and 3 feet in diameter. Magnetic disc files can be classified as those with fixed heads (one head per track on each disc), those with moving heads, and those with removable disc stacks (and moving heads). Disc files with moving heads can further be divided into those in which the heads move in one dimension only (in and out among the stack of discs) and those in which the heads move in two dimensions (up and down the stack of discs as well as in and out among the stack). The major effects that these differences have on the characteristics of the devices are indicated in Figure 2.

TYPE OF DEVICE	ON-LINE CAPACITY PER-UNIT IN CHAR.	TYPICAL ON-LINE COSTS IN ¢/CHAR.	AVERAGE ACCESS TIME	DATA TRANS-FER RATE IN CH/SEC.	REMOV-ABLE MEDIA	MULTIPLE ACCESS CAPA-BILITY	MAJOR ADVAN-TAGES	MAJOR DISADVAN-TAGES
MAGNETIC TAPE LOOPS	50×10^6 to 500×10^6	0.1	8 sec.	20,000 to 100,000	YES	NO	LOW COST	VERY SLOW ACCESS
LARGE FIXED-HEAD MAG. DRUMS	0.2×10^6 to 1.0×10^6	2.0	15 ms	100,000 to 200,000	NO	POSSIBLE	FAST ACCESS	HIGH COST, LOW CAPACITY
MOVING-HEAD MAGNETIC DRUMS	4.0×10^6 to 65×10^6	0.3	100 ms	50,000 to 150,000	NO	NO	LARGE CAPACITY, LOW COST	MEDIUM SPEED ACCESS
FIXED-HEAD MAGNETIC DISC FILES	10×10^6 to 25×10^6	0.6	20 ms	100,000 to 350,000	NO	POSSIBLE	FAST ACCESS	HIGH COST
1 DIMENSION MOVING-HEAD MAG. DISC.	10×10^6 to 150×10^6	0.2	100 ms	100,000 to 400,000	NO	POSSIBLE	LARGE CAPACITY, LOW COST	MEDIUM SPEED ACCESS
2 DIMENSION MOVING-HEAD MAG. DISC	10×10^6 to 150×10^6	0.15	500 ms	50,000 to 100,000	NO	NO	LARGE CAPACITY, LOW COST	SLOW ACCESS
REMOVABLE-STACK DISC FILES	2.0×10^6	1.2 (on-line) 0.02 (off-line)	150 ms	80,000	YES	POSSIBLE	LARGE OFF-LINE CAPACITY, LOW COST	SMALL ON-LINE CAPACITY
MAGNETIC CARD FILES	5.5×10^6	1.0 (on-line) 0.003 (off-line)	200 ms	100,000	YES	NO	LARGE OFF-LINE CAP., LOW COST, DISCRETE CARD	SMALL ON-LINE CAPACITY
* WOVEN SCREEN MEMORY	1.0×10^6 to 10×10^6	9.0	10 us	100,000	NO	NO	FAST ACCESS, NON-MECH.	HIGH COST, NOT CURRENTLY AVAILABLE

* Note: All figures shown for Woven Screen Memory are estimates of future developments.

Figure 2. Summary of Characteristics of Mass Memories.

Magnetic disc files, including fixed and moving head types, have on-line capacities ranging from approximately 2 million to over 100 million alphanumeric characters per unit and access times ranging from 20 milliseconds to several hundred milliseconds. The moving-head disc files have lower costs per bit of storage and slower access times. The fixed-head disc files have access times roughly comparable to those of fixed-head magnetic drums. In general, the larger the number of bits that can be accessed by a single head and selection mechanism, the lower the cost per bit and the longer the access time.

a. Fixed-Head Magnetic Disc Files

Disc file storage units with fixed heads usually involve a limited number of discs, a maximum number of bits per track, and a fixed head for each track.[7] This type of storage permits a higher track density since the fixed heads eliminate the need for mechanical positioning of the head and the resulting allowances for mechanical tolerances. The large multiplicity of heads and the required electronic switching between heads results in a significantly higher cost per bit than for the moving-head type disc storage.

Although this type of disc storage is somewhat similar in functions and characteristics to fixed-head magnetic drums, the use of three dimensions instead of two permits greater volumetric efficiency—greater storage capacity in a more compact unit. There are, of course, also differences in the mechanical design problems between such disc and drum units, but these are outside the scope of this paper.

Fixed-head magnetic disc units provide capacities in the order of 20 million alphanumeric characters and average access times of approximately 20 milliseconds. The penalty paid for this is a higher cost per bit of storage.

b. Moving-Head Magnetic Disc Files

The first commercially available magnetic disc files involved a two-dimensional head movement. A single head mechanism was moved up and down parallel to the disc stack and shaft to select one of a number of discs and then moved in between adjacent discs to select the desired track. In this unit, the head-mount arm straddled a disc providing a head to read the upper surface of the disc and another head to read the lower surface.

Although some modern large capacity disc files also operate on this principle, most of the present units involve a one-dimensional movement. A head mount is inserted between each pair of adjacent discs, usually with one head reading the lower surface of the upper disc and another head on the same mount reading the upper surface of the lower disc. This type of disc file provides a much faster access by eliminating the necessity for moving the heads in the dimension parallel to the disc shaft. The penalty paid for this faster access is the increase in the cost per bit of storage due to the cost of the larger number of heads and the electronic switching between heads compared against the cost of a disc selector mechanism.

The one-dimensional movement permits two secondary advantages that are not apparent from the comparisons in Figure 2. Since there is at least one head for each disk, it is possible to provide a larger number of read and write amplifiers to permit reading or writing multiple tracks simultaneously with a significant increase in the effective instantaneous data transfer rate. This is of particular significance in some high-speed binary scientific machines. It could permit all bits of a complete binary word to be read or written in parallel. In practice, the instantaneous data transfer rate could be even higher since the number of heads is greater than one per disc in most cases. In order to reduce the access time, the distance that the arm must move is reduced by spacing several heads equidistant along the arm. This has the effect of dividing the disc surface into several bands. All of the tracks within one band are accessed by a single head.

Even without simultaneous reading or writing from all heads, another advantage can be achieved. For any given arm position, the heads are switched electronically. This increases the information that can be transferred without moving the arm. With appropriate organization of the problem, this can reduce the number of arm movements required with a consequent increase in the effective speed of operation.

The insertion of the set of head mounts between pairs of adjacent discs can be, and has been, accomplished in several different ways mechanically. In one design, the head mounts for all discs are moved together by a common track selection mechanism. As a result, all of the heads are moved in and out simultaneously to corresponding tracks on each disc. This can be pictured as a comb of head mounts moving in and out perpendicular to the disc shaft. For any one position, the tracks being read or written on each of the discs describes a cylinder conceptually similar to a magnetic drum with wide track spacing. Another design provides independent head positioning mechanisms for each disc. If utilized, the ability to independently access tracks on different discs can permit a significant decrease in effective access time since several accesses to different discs can be overlapped or performed simultaneously.[8]

The moving-head types of disc files provide capacities from 10 to 150 million characters and average access times from 100 to several hundred milliseconds. The cost per bit of storage is somewhat greater for those using a one-dimensional head movement than for those using a two-dimensional head movement. However, this is still significantly less than for fixed-head disc and drum units.

c. Removable-Stack Disc File

The newest addition to the disc storage family is the removable-disc-stack unit.[9] In this device, a drive mechanism is provided to handle a small stack of discs that can be removed, replaced, and interchanged with other stacks. Each stack of discs stores approximately 2 million alphanumeric.

This device provides a compromise between the off-line storage capability of magnetic tape and the on-line fast access capability of larger mass memories. A series of disc stacks can be stored away on a shelf and put on the drive mechanism as required. Each disc stack has approximately one fourth the capacity of a tape reel with an order of magnitude higher cost. However, all data within a stack can be on-line and addressable by the computer to provide fast access within blocks of two million characters at a time. This is particularly well suited to the requirements of many types of business problems for large total file storage capability but on-line fast access to only a segment of this in any given processing operation.

The average access time of this type of unit is in the order of 150 milliseconds with a relatively low capacity of two million characters per unit. The cost per bit of on-line storage is relatively expensive compared to the other types of disc units, but the cost per bit of total storage including disc stacks stored off-line on shelves is cheaper. The obvious question that must be considered in using this type of unit is the relative importance of on-line vs. off-line storage capacity.

3. Magnetic Cards

The magnetic-card type of mass memory, which preceded the removable-stack disc storage by over two years, is quite different physically and mechanically. However, from a systems and applications standpoint the two are somewhat similar in that the magnetic card memory also provides a certain amount of on-line storage capacity (approximately 5.5 million characters per unit) and an almost limitless amount of off-line storage capacity. The cost for on-line storage capacity is roughly equivalent for the two types of devices, but the off-line storage capacity is cheaper for the magnetic-card mass memory. The magnetic-card type offers another advantage over disc files in that individual cards can be copied, inserted, removed, or replaced.

The only random-access magnetic-card memory available commercially at present has an average access time of approximately 200 milliseconds and an on-line capacity of 5.5 million characters per unit.[10] In actual usage, these access times of 200 milliseconds may be effectively reduced in many cases since, while one card is being read or written, the next card may be selected from a magazine and the preceding card returned to the magazine.

In this particular unit, oxide-coated Mylar cards, approximately 3″ x 14″ in size, are hung from rods in the magazine. These rods may be selectively turned to select the card with binary-coded notches corresponding to the rods that have been turned, thus providing the ability to select any card from the magazine at random. The selected card is then dropped to the surface of a rotating drum and accelerated to the surface speed of the drum so that it can be read or written while passing under a set of heads. The card may be held on the drum for rereading or for reading another set of tracks on the same card. When it is released from the drum, it is automatically returned to the magazine. Its location in the magazine is immaterial since the selection is by the coded notches in the card and the combination of rods that are turned rather than by physical location.

B. *STATIC OR NON-MOVING-MEDIA TYPES OF MASS MEMORIES*

Consideration of static or non-moving-media types of mass memories in a survey at this time is largely conjectural since none are commercially available. It is unlikely that any will be available in the foreseeable future with capacities that would qualify them to be considered mass memories. However, truly random access times in the order of microseconds and the absence of moving parts will continue to lend an impetus to the work on devices of this type. Techniques that offer possible promise for non-mechanical mass memories, would include thin-magnetic-film memories, super-conductive memories, and the woven screen memory.[11]

It is likely in the long run that an all magnetic and/or electronic mass memory will replace those involving mechanical motion. However, there does not seem to be a strong likelihood that this will happen within the next few years for memories with the capacities that we have been discussing. It may be technically feasible to do this within the next few years, but it is doubtful that it will be economically feasible. From the user and the systems standpoints, the penalties paid in terms of the slower access times of the moving-media memories are not sufficient to justify significantly higher costs for faster access for large-capacity auxiliary memories. In most applications for mass memories, reducing the access time from milliseconds to microseconds would not justify an order of magnitude increase in the cost per bit of storage and probably not an increase of four or five times. It is either simply not worthwhile or there are cheaper ways of realizing most of the advantages.

One way in which many of the advantages of a large-capacity, fast-access, low-cost mass memory can be realized is by using hierarchies of storage. This is becoming more widespread and the trend will probably continue. Since the days of UNIVAC I, we have had high-speed one-word registers, main internal storage, and magnetic tapes used together in a machine—each fulfilling the role for which it is best qualified. Similarly, small-capacity, high-speed, random-access memories (e.g., magnetic cores); medium-speed, medium-capacity, semi-random-access memories (e.g., fixed-head magnetic drums or discs); and large-capacity, slower semi-random-access mass memories (e.g., magnetic disc files) can be used in conjunction with one another with each fulfilling the role where its advantages are maximized and its disadvantages minimized. With proper hardware, systems, and programming design, such memory combinations can achieve most of the advantages of large capacity, fast access, and moderate cost.

Non-mechanical mass memories will enjoy the advantages of faster access, and probably higher reliability, as a result of not having to move the magnetic media and position a head. However, this blessing may be a curse in disguise since these mechanical motions serve a selection function that will have to be provided by electronic switching in the static memories. As the memory capacities become larger and larger, this electronic switching problem may be all but insurmountable in the foreseeable future. Certainly, as the cost per bit of the memory array is decreased, the selection and switching circuitry will become a significant, if not dominant, part of the total cost of such a memory.

The emphasis placed on the difficulties facing this type of mass memory may seem out of proportion or baised relative to the discussion of drums, magnetic discs, and magnetic cards where the emphasis was primarily on their characteristics rather than their difficulties. This is not due to a deliberate intent to discredit this type of mass memory but rather to the fact that the moving-media memories are commercially available whereas the static ones are still in the laboratory development stage at this time. There is certainly a need and a place for this type of mass memory when the technology has evolved and the cost has dropped to the point that they can compete on the basis of performance vs. cost. The amount of work being expended in this area by a number of major organizations in the computer field attests to the interest and to the applications for this type of mass memory.

Based on laboratory results that have been reported to date, a static mass memory of approximately one to ten million characters with access times in the order of 10 microseconds would seem a reasonable expectation for the future.

THE EFFECT OF THE ADVANTAGES AND DISADVANTAGES OF DIFFERENT MASS MEMORIES ON THE CHOICE OF TYPES FOR SPECIFIC APPLICATIONS

From the information in Figure 2 and the preceding discussion, the major advantages and disadvantages of the different types of mass memories can be summarized as follows:

TYPE OF MASS MEMORY	ADVANTAGES	DISADVANTAGES
Fixed-Head Magnetic Drums	Fast access, no mechanical head motion, high continuous data transfer rate	Low capacity, high cost per bit, poorer volumetric efficiency, large electronic switching matrix, large number of heads
Moving-Head Magnetic Drums	Large capacity, low cost per bit, possibility of parallel reading or writing from multiple heads to greatly increase instantaneous data transfer rate	Poorer volumetric efficiency, relatively large number of heads for medium speed access or slower access if fewer heads
Fixed-Head Magnetic Discs	Fast access, medium capacity, no mechanical head motion, high continuous data transfer rate	High cost per bit of storage, large electronic switching matrix, large number of heads
Two-Dimension Moving-Head Magnetic Discs	Large capacity, minimum number of heads, low cost per bit	More complex positioning mechanism, slowest access, slow continuous data transfer rate
One-Dimension Moving-Head Magnetic Discs	Large capacity, possibility of multiple simultaneous accesses if heads are positioned independently, low cost per bit compared to fixed head units, possibility of parallel reading or writing from multiple heads to greatly increase instantaneous data transfer rate	Relatively large number of heads, somewhat higher cost per bit compared to two-dimension disc unit, medium speed access
Removable-Stack Discs	Large off-line capacity, low cost per bit of off-line storage, combines on-line random-access capability with large off-line capacity	Limited on-line capacity, higher cost per bit of on-line storage
Magnetic Card Memory	Large off-line capacity, low cost per bit of off-line storage, combines on-line random-access capability with large off-line capacity, individual cards can be copied, replaced, or inserted.	Limited on-line capacity, higher cost per bit of on-line storage
Woven Screen Memory	Fastest access, no mechanical motion	Lower capacity, higher cost per bit, not currently available

In selecting a mass memory for a particular application, it would first be necessary to use the advantages and disadvantages listed above to select the types of mass memory best suited to the particular application. It would then be necessary to compare the detailed characteristics of those types and relate these detailed characteristics to the specific requirements of the problem.

The relative importance of the different advantages and disadvantages may vary greatly from one application to another. In the use of a mass memory for dumping the contents of the internal memory of a large scientific computer, fast access time and the effective data transfer rate for continuous transfer of large blocks of data would be more important than extremely large capacities. Therefore, this type of application would favor large drums or disc units with fixed heads for each track; certainly not the type of disc file that positions the head from disc to disc. On the other hand, in a large inventory control application (or applications involving large files requiring relatively infrequent random interrogations) large capacity and low cost would be more important than data transfer rate and access time. Hence, disc files with a single head mount moving from disc to disc as well as from track to track might be chosen. For other types of business problems (e.g., payroll and labor cost distribution applications) where it is necessary to process each transaction against two or more different files, a mass memory with the ability to independently access two or more locations wculd be more advantageous.

These few examples serve to illustrate the point that we have to compare the individual requirements of the problem against the detailed characteristics of each type of mass memory to determine the one best suited for each specific application. A mass memory with a fast random-access time, a high data transfer rate, a large capacity, and a very low cost per bit would meet the requirements for most mass memory applications. Unfortunately, this device is not available at present and is not likely to be in the near future. Since the user must continue to select the best available device for his particular application, there is a place and a need for almost all of the types of mass memories discussed here.

This emphasizes the need indicated earlier for combinations of different types of internal and external memories. Instead of waiting and longing for the millennium, the system designer must learn to use a hierarchy of storage to achieve a compromise between capacity, access time, data transfer rate, and cost. Kilburn, Edwards, Lanigan, and Sumner have described one such system in use today in which a combination of storage devices with different characteristics are made to appear to the programmer as a single large internal memory.[12] This concept can and should be extended to mass memories.

PROGRAMMING AND APPLICATION CONSIDERATIONS

Most of the problems of mass memories discussed so far have been concerned with the hardware aspects—mechanical, electrical, and magnetic. At least a passing mention should be made of several significant problems involved in the use and application of mass memories that are common to most types. In some cases, it is possible to provide hardware features to alleviate problems such as those involved in programming, addressing, formatting, organization, and the protection of stored information.

The programming of systems involving a mass memory is directly influenced by the characteristics of the individual mass memory being used—e.g., the access time, availability of interrupt signals, error control techniques, block or record size, and addressing techniques. The format and file organization is affected by the number of bits or characters per track, the number of tracks and heads per arm position, and the addressing and buffering techniques used. The protection of critical information stored in the mass memory is a difficult problem. The possibility of the operator, the programmer, or machine errors causing a writing operation in a particular location containing vital information must be avoided. A combination of programming and hardware techniques appears to be the best answer but there is still much work remaining in this area before a good solution is achieved. Most of these problems, of course, are different for magnetic card memories or non-moving-media mass memories than for drums and disc files, but they still exist to a greater or lesser degree.

The actual programming and application of a mass memory can be simplified by hardware features if adequate attention is given to this in the design of the mass memory. We would like to design standardized mass memories that can be used with any computer, but it becomes difficult to provide hardware features to facilitate the use of the memory with one machine without their becoming an unusable and uneconomic burden for another machine. Fortunately, some hardware techniques have been developed that offer advantages for use with most computers. An example of this is the provision, in some disc files, of one or more discs having fixed heads in addition to the bulk of the disc file that is served by moving heads. The fixed-head tracks permit faster access and synchronized storage that can be used for addressing, indexing, and buffering purposes.

FUTURE MASS MEMORY CAPABILITIES AND LIMITS OF PERFORMANCE FOR DIFFERENT TYPES OF MASS MEMORIES

Since we must live with existing types of mass memories—largely moving-magnetic-media types—for the foreseeable future, we should consider what potential capabilities might be realized with these devices, when we might expect to reach these limitations, and when non-moving-media mass memories might reasonably be expected to be practical, feasible, and commercially available on a competitive basis.

In 1962, A. S. Hoagland pointed out that the storage density of one manufacturer's commercial disc files increased from 2000 bits per square inch in 1956 to 25,000 bits per square inch in 1961.[13] He then predicted that storage densities of "one million bits per square inch (e.g., approximately 5000 bpi, 200 tpi) will become the state of the art" within the next few years. A few months earlier M. Jacoby predicted densities of 3000 bpi and 500 tpi (1.5 million bits per square inch) would "become common-place in a few years".[4] He then indicated that these densities could provide storage capacities of 10 to 100 billion bits if a possible increase in the physical size is also considered. Thus, increases in capacity of one to two orders of magnitude over the largest present mass memories can be anticipated.

The cost per unit can be expected to decrease even with the larger capacities as the technology is improved and more manufacturing experience is obtained. Hence, the cost per bit of storage can also be expected to decrease by one to two orders of magnitude—possibly to 0.0001 cents per bit for the mass memory itself (not including control and buffering electronics). Although the picture for the future of capacity and cost appear bright, there is little hope for significant improvements in average access times for moving-media mass memories. Due to inherent mechanical motions involved we cannot expect improvements of as much as an order of magnitude over presently available devices. For significant improvements in access time we must turn to the non-moving-media type devices.

With respect to non-moving memories J. A. Rajchman wrote in 1962, "Capacities of several million bits are the maximum attainable with reasonable economy by any magnetic technique."[14] He held somewhat greater hopes for superconductive memories stating that they "may have storage capacities of billions of bits". An idea of the cost of such memories was given by his estimate that a one billion bit coincident-current magnetic matrix memory would require about 20 million semiconductor devices. He then drew the conclusion that batch fabrication techniques would be essential for both the storage elements and the semiconductor devices. A diagram that he presented to summarize the limits of speed and storage capacity indicated approximate limits of less than 10 million bits capacity at 10 us access time for magnetics and less than 10 billion bits at 100 us for superconductive memories. The availability of devices of these types, with capacities approaching these limits, at reasonable costs is certainly many years away.

It is this speaker's firm conviction that moving-magnetic-media mass memories such as drums, disc files, and magnetic-card files will be around for a long time to come. Although these devices may ultimately be replaced by new techniques, it appears unlikely that these other techniques will provide generally competitive devices before 1970 at the earliest. The moving-media types will be used in the great majority of mass memory applications through 1970 with newer techniques, such as the woven screen memory, coming gradually into wider usage in applications where fast access time is more important than capacity and cost. It is difficult to believe that such devices will compete with moving-media mass memories on a cost and capacity basis until the post 1970 era.

It is likely that continued improvements and innovations in moving-media mass memories will provide ultimate capacities, access times, data transfer rates, and costs superior to those indicated above. Just as the development of the floating head permitted densities and rates in excess of those previously anticipated for fixed heads, presently unforeseen developments may well serve to push the limits of these devices beyond present expectations. An example of work on one such development has been described by Hoagland.[15] This is a disc unit in which the head is positioned on a track under control of a servo system with the signal read from the track being part of the control loop to permit far greater track density and multiple access arms.

Ultimately, the non-moving-media types of mass memories will have to prove themselves on a basis of purely competitive costs if they are to supplant the majority of moving-media mass memories. Although there are certain applications in which a premium will be paid for the faster random-access capability of devices such as the woven screen memory, their widespread use will be limited until the cost per bit reaches a competitive point. The hard cold fact is that despite such talk of fast access, the advantages in most applications do not justify significantly higher costs per bit. The use of hierarchies of memories, discussed previously, will help to assure that this is the case.

One area in which the moving-media mass memory will be at a disadvantage compared to devices such as the woven screen memory is in military applications requiring ruggedness and high reliability under conditions of mobility and adverse environment. Inherently, the mechanical motions involved in moving-media devices put them at a disadvantage. As a result, military applications of this type may be among the first large scale users of non-moving-media mass memories since relatively standard military packaging techniques can be utilized to meet the requirements for ruggedness, mobility, reliability, and operating conditions such as temperature, humidity, dust, shock, and vibration. This is an area to which insufficient attention has been given in the past. There have been a number of efforts to militarize essentially commercial type devices, but it is doubtful whether this is the proper approach and whether it will be fruitful in the long run when compared to the development of new types of mass memories that are inherently more suitable to these adverse conditions. This is an example of the type of application in which a cost premium could be paid for improvements in performance and operating conditions.

CONCLUSION

This paper has briefly discussed some of the major requirements for mass memories, the major categories and types presently available, and the significant advantages and disadvantages of each. It has been predicted that the moving-media type devices will continue to dominate the mass memory field at least through 1970, but that non-mechanical techniques providing faster access times will gradually come into wider spread usage as the cost is decreased and the capacity increased. The remaining papers in this session will discuss four specific types of mass memories. It is hoped that this discussion will serve to present a basis and framework for a better understanding of the significance of the points raised in the subsequent papers.

REFERENCES

1. Gross, W. A.: A Gas Film Lubrication Study—Part I—Some Theoretical Analyses of Slider Bearings, IBM Journal of Research and Development, Vol. 3, pp. 237–274, July 1959.

2. HAUGHTON, K. E.: Air-Lubricated Slider Bearings for Magnetic Recording Spacing Control, Large-Capacity Memory Techniques for Computing Systems, pp. 341-350. The MacMillan Company, New York, 1962.
3. Preprints of Papers Presented at First Disc File Symposium, Informatics Inc., March 1963.
4. JACOBY, M.: A Critical Study of Mass Storage Devices and Techniques with Emphasis on Design Criteria, IRE–PG MIL, National Winter Convention on Military Electronics, 1962.
5. STUART-WILLIAMS, R. and WIESELMAN, I. L.: File Storage—Existing File-Storage Systems and the Design of Disc Files, Special Technical Presentation, Data Products Corporation, 1962.
6. NELSON, R. C.: Magnetic Drums and Discs—Survey, Instruments and Control Systems, pp. 109–120, January 1962.
7. JACK, R. W.: Engineering Description of the Burroughs Disc File, Proceedings Fall Joint Computer Conference, Las Vegas, Nevada, pp. – , November 12–14, 1963.
8. WIESELMAN, IRVING L., SAMPSON, DONALD K., STUART-WILLIAMS, RAYMOND: A Multiple Access Disc File, Proceedings Fall Joint Computer Conference, Las Vegas, Nevada, pp. – , November 12–14, 1963.
9. CAROTHERS, J. D., BRUNNER, R. K., DAWSON, J. L., HALFHILL, M. O., and KUBEC, R. E.: A New High Density Recording System: The IBM 1311 Disc Storage Drive With Interchangeable Disc Packs, Proceedings Fall Joint Computer Conference, Las Vegas, Nevada, pp. – , November 12–14, 1963.
10. BLOOM, L., PARDO, I., KENTING, W., and MAYNE, E.: Card Random Access Memory (CRAM): Functions and Use, Proceedings Eastern Joint Computer Conference, Washington, D.C., pp. 147–157, December 12–14, 1961.
11. DAVIS, J. S.: Investigation of a Woven Screen Mass Memory System, Proceedings Fall Joint Computer Conference, Las Vegas, Nevada, pp. – , November 12–14, 1963.
12. KILBURN, T., EDWARDS, D. B. G., LANIGAN, M. J., and SUMNER, F. H.: One-Level Storage System, University of Manchester, IRE Transactions on Electronic Computers, Vol. E. C–11, pp. 223–235, April 1962.
13. HOAGLAND, A. S.: Mass Storage, Proceedings, IRE, Vol. 50, pp. 1087–1092, May 1962.
14. RAJCHMAN, J. A.: Computer Memories—Possible Future Developments, RCA Review, Vol. 23, pp. 147–151, June 1962.
15. HOAGLAND, A. S.: A High Track-Density Servo-Access System for Magnetic Recording Disc Storage, IBM Journal of Research and Development, Vol. 5, pp. 287–296, October 1961.
16. COIL, E. A.: A Multi-Addressable Random Access File System, 1960 IRE WESCON Convention Record, Part 4, pp. 42–47, August, 1960.
17. MCLAUGHLIN, H. J.: Disc File Memories, Instruments Control Systems, Vol. 34, pp. 2063–2068, November, 1961.
18. OTS, Dept. Commerce: Information Storage and Retrieval, U.S. Government Research Departments, Vol. 37, p.s.–29 (A), January, 1962.
19. LENNON, W. T. JR., and JORDON, W. F.: Auxiliary Memory Speeds Information Retrieval, Computer Control Company, Electronics, Vol. 35, pp. 102–104, May 11, 1962.
20. CARVER, W. W.: Comparing Storage Methods, Burroughs, Electronic Industries, Vol. 21, pp. 120–130, August 1962.

INVESTIGATION OF A WOVEN SCREEN MEMORY SYSTEM

J. S. Davis, P. E. Wells
Members of the Technical Staff
TRW Computer Division
Thompson Ramo Wooldridge Inc.
Canoga Park, California

INTRODUCTION

The woven aperture screen plane concept is a new fabrication technique that could lead to high speed production of memory stacks. The woven aperture screen planes have many characteristics which are similar to ferrite core planes. However, the environmental characteristics of the woven aperture screen planes are superior in many respects to those of ferrite core planes. Large planes have operated at +105°C. Tests on small planes have been conducted at +195° C.

The most significant aspect of the woven aperture screen plane technique is the inherent low cost of this approach providing the yield in production is as expected. The potential low cost feature of this techniue (0.1 to 1.0 cents per bit) makes it extremely attractive when one considers the possibility of designing a random access mass memory system.

Early work in this field[1] led to the conclusion that this technique might only be applied to linear select memory planes. However, subsequent improvements in electro-deposition of nickel alloys and in the preparation of the substrate now indicates that coincident-current quality planes can be achieved. Further investigation in the weaving areas and improvements in insulation materials for wire have resulted in machine woven planes of substantial sizes.

At the present time, the experimental laboratory developmental program has reached its conclusion. A detailed plating procedure has been developed; production designs for the planes have been implemented; a fabrication procedure has been formulated; and several designs for the high density connector have been proposed.

The major emphasis to date has been placed on the plating problem. Uniformity and reproducibility are the most important factors with any batch process technique. The largest hand woven plane which has been successfully uniformly plated contains slightly over 8000 bits. The reproducibility from plane to plane has been encouraging considering the limitations imposed by the small laboratory beakers used.

The initial machine woven planes were fabricated on wire looms but this experiment was partially successful. The insulation of the wire was not suitable and the forces encountered in the wire loom were excessive. Subsequent weaving experiments on production cloth looms achieved the desired results but the aperture uniformity varied slightly because uniform tension could not be applied to each wire in the warp. Small sections of machine woven planes have been processed to demonstrate the selective plating feature and the characteristics of these planes are similar to hand woven planes. Modifications to existing cloth looms should result in machine woven planes with the desired characteristics.

The average characteristics of the cell are presented. The unique cell geometry influences the cell characteristics as shown.

A detailed presentation is given of the characteristics of the drive lines. The resistance and capacitance of the drive and sense lines are relatively high compared to the characteristics of core drive lines.

A double selection scheme is proposed for all three driver matrices and the possibility of switching sense lines is discussed. The size of the plane is estimated and a discussion given of the production procedure and cost.

The general characteristics of a suitable memory system are proposed.

CHARACTERISTICS OF THE PLANES

It is important to first discuss the characteristics of the woven planes before discussing memory organization. The basic module size, plane size, cell density and drive requirements are based on the individual cell characteristics.

Cell Geometry

The woven cell is formed by weaving insulated wires along with bare metallic wires to form an orthogonal matrix of drive and sense wires which thread aperture cells. Square aperture cells are formed by the bare metallic wires which are coated with a remnant magnetic material in an electro-deposition process. To eliminate the interaction between cells, each cell is surrounded by a buffer cell. The X and Y Buffer cells are rectangular. The diagonal buffer cell is square.

Figure 1 is a schematic drawing of a single cell showing the buffer cells. The woven cell is a four wire, single turn configuration. The Y drive line is woven in an over-under pattern and is parallel to the sense line. The X drive line is parallel to the inhibit line and perpendicular to the sense line.

The dummy insulated wires terminate at the edge of the plane and one wire is necessary in a plain weave to provide the buffer cell. For a

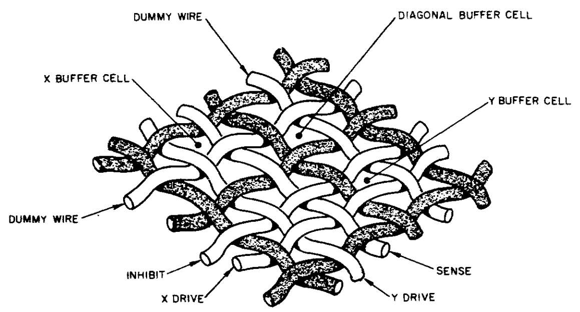


Figure 1. Typical 4-Wire Single Turn Memory Cell.

wire to thread the cell in a plain over and under weave, it is necessary for the cell to contain an even number of wires. Conversely, for a wire not to thread a buffer cell, it is necessary that the buffer cell contain an odd number of wires. Since the minimum odd number is one, the buffer cell contains a single wire.

The total mmf around a square cell is equal to NI; where I is the sum of the currents in the X and Y drive line; N is the number of turns. The mmf around a buffer cell is equal to zero.

In the plain weave, the diameter of the copper and insulated wires are almost equal. Also, the diameter of the wire is almost equal to the open distance between adjacent wires. Thus, to a first approximation, the path length, (L), for a cell with a single X and Y drive is

$$L \approx 4 \times 6 \times D \qquad (1)$$

where D = diameter wire in inches.

There are two opposite requirements on the diameter size of the wire. The diameter of the bare copper wire should be as large as possible to increase the total flux in each cell. Also, the diameter of the insulated wire should be large to decrease the resistance and inductance of the drive and sense lines. Conversely, the diameter of both wires (copper and insulated) should be decreased to increase cell density and reduce the length of the drive and sense wires.

The best compromise, at this time, appears to be a wire diameter of 5×10^{-3} inches. This choice is not only influenced by cell geometry, circuit, and plating considerations but also by the problems encountered in connector fabrication, wire manufacture, and plane assembly.

The path length for a single turn cell is approximately equal to 0.12 inch.

Cell Density

The cell density can be calculated by assuming a wire diameter, wire mesh and cell configuration. The lineal cell density CD_x is

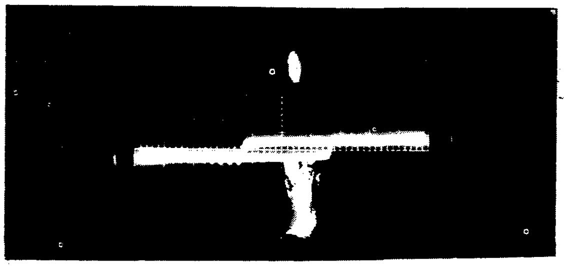
$$CD_x = \frac{\text{mesh of plane}}{\text{openings per cell}} \qquad (2)$$

Hence, if the mesh of the plane is 100 (openings per inch) and the openings per cell is 5 (including buffer zone), the lineal cell density is 20 cells per inch. Since the weave is square, the area cell density is 400 cells/square inch.

Cell density can be improved by reducing the diameter of the wire and by eliminating the buffer cell. These points will be discussed later under the high density plane configuration.

COERCIVE FORCE

Figure 2 shows the quality of the B-H loop for a plane which has been completely machine woven with drive and sense wires and plated with an iron-nickel and cobalt alloy. This experimental plane was woven with 5×10^{-3} inch wire. The coercive force, Hc, is 1.8 oersteds. The quality of the B-H loop indicates that the woven plane could be driven in a coincident current mode.



Wire Diameter	— 5 mils
Mesh	— 65
Horizontal Scale	— 1Oe/div

Figure 2. B-H Curve for a Machine Woven Plane.

The B-H loop for the plane was obtained by driving the woven screen to saturation along one axis of the plane. Screens are tested in a B-H tester prior to mounting them on a connector frame in experimental work.

Cell Drive Requirements

Assuming a path length of 0.120 inch and a coercive force of 2 oersteds for full select, the required mmf to switch the cell can be computed from the following equation:

$$\begin{aligned}\text{mmf (ampere turns)} &\approx 2 \times H \times \text{path length (inches)} \\ &\approx 4 \times 0.12 = 0.48 \text{ ampere turns}\end{aligned} \qquad (3)$$

Thus, with a one turn X and Y drive line, the half select current is 240 ma. The inhibit current requirement is also 240 ma.

Readback Signal and Switching Time

With a given path length and coercive force, the readback signal is a function of the residual induction, diameter of the wire, thickness of the magnetic layer, rise time of the drive currents and the amount of overdrive in the full select current. The maximum read drive current is not limited by the interaction between adjacent cells.[2] Experiments have been conducted to show that interference with adjacent cells is essentially zero even when the drive current is 1.5 times the optimum value. The optimum cell drive current is that value which is required for switching on the major hysteresis loop. Each cell contains a family of hysteresis loops. The minor hysteresis loops are not square and are, therefore, not desirable. Therefore, in presenting the cell characteristics, the read drive current is held constant at the optimum value, and the write current is varied to obtain characteristic curves. Figure 3 shows the typical response curves for a hand woven screen where the path was 0.25 inch. The read current is 800 ma. The undisturbed one voltage curve, μV_1, is a plot of the peak voltage and increases with write current.

At 600 ma write current, the peak voltage decreases. The disturbed one peak voltage, dV_1, follows the same general pattern. The disturbed zero peak voltage curve, dV_z, shows that the material is essentially square to a half select

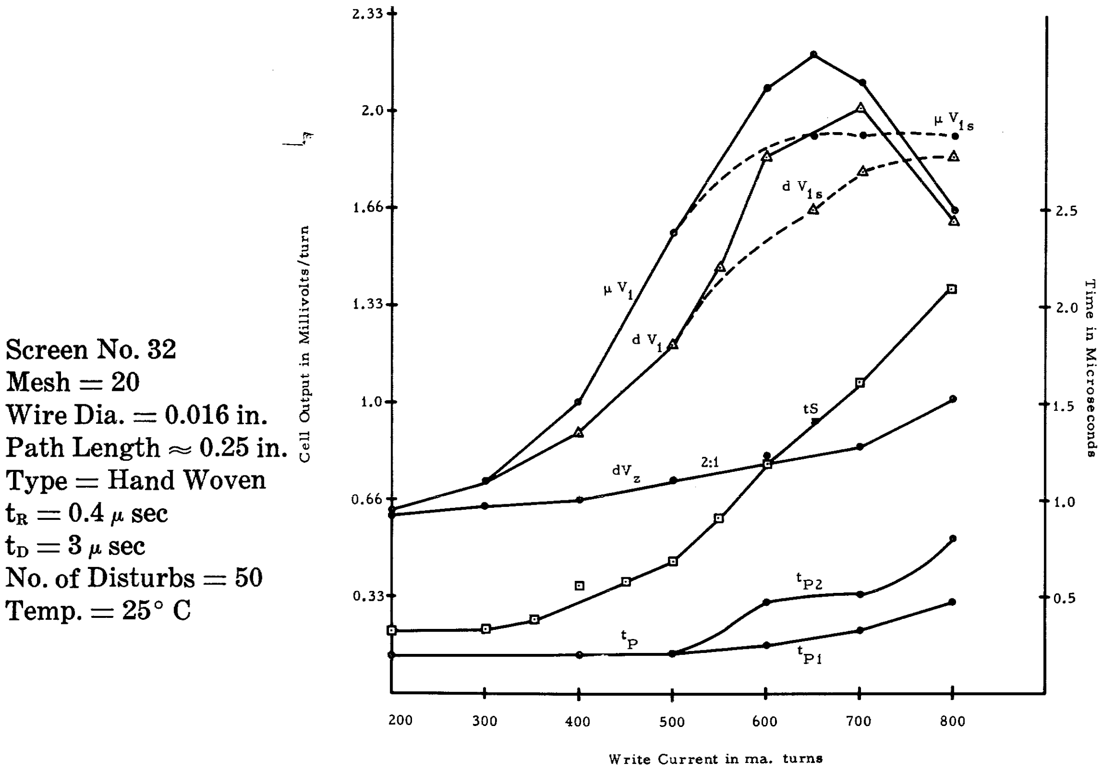


Figure 3. Average Characteristics.

current of around 350 ma. The number of disturb pulses in both cases, zeros and ones, was equal to fifty.

The curve for the switching time, ts, increases with drive current and starts to level off. If the write drive currents were increased beyond 800 ma, ts would decrease, as expected, due to the excess drive current.

The curve for the peaking time, tp, is extremely interesting in that it shows that a double peak occurs at write currents above 500 ma. The first peak, tp_1, increases until it finally decreases when the write drive current exceeds 800 ma. The second peak, tp_2, follows the first peak voltage by approximately 0.3 μsec. If the read and write drive currents were increased past 800 ma, the second peak, tp_2, would move in the direction of the first peak, tp_1, as the excess drive increased.

The curves in Figure 3 are extremely interesting in that they suggest the possibility of a dual path length around the cell on the major hysteresis loop. This broadening of the peak voltage curve can be extremely useful in achieving high signal to noise ratios at strobe time. Hence, the strobe is always positioned at the second peak time at full write drive current. The curves μV_{1s} and dV_{1s} are the curves of the undisturbed one voltage and disturbed one voltage measured at strobe time. Thus, it can be seen that although the peak voltage curves shift slightly with drive current, the values at strobe time are constant providing the write drive current exceeds 700 ma.

Figure 4 shows the shape of the readback signals for twenty five cells which were sampled over the entire surface of the screen. The double hump read signal is clearly visible.

Figure 5 shows the μV_{1s} cell output for different constant drive currents as the write current is varied. In the linear select mode, where the ⅓ select digit current is equal to 400 ma, the cell output is 5 mv per turn and switching time, ts, is 0.5 μsec. The output signals shown in Figure 3 and Figure 5 are somewhat low because of the slow rise time on the current pulses.

It was not possible to achieve faster rise time than 0.4 μsec with existing equipment. The knee of the material as shown by Figure 5 is 400 ma. The 600 ma read curve for μV_{1s} shows that the minor hysteresis loop is not as square as the major hysteresis loop which is achieved when the read current reaches 800 ma.

Noise Cancellation

There are three types of noise problems that must be overcome in sensing a plane; capacitive noise, inductive noise and magnetic noise. The capacitive noise program is generally solved by utilizing a differential input stage in the sense amplifier. The inductive noise problem is very serious and must be solved by utilizing special wiring or planar cancellation techniques.[3] Special cancellation techniques must be used to overcome magnetic noise.

Figure 1 shows that the sense line parallel to Y drive line although it is woven in the opposite phase. The inductive noise on the sense line is cancelled by sharing the sense line between two planes in the Y axis. In wiring a memory stack, the X axis would be divided into two sections.

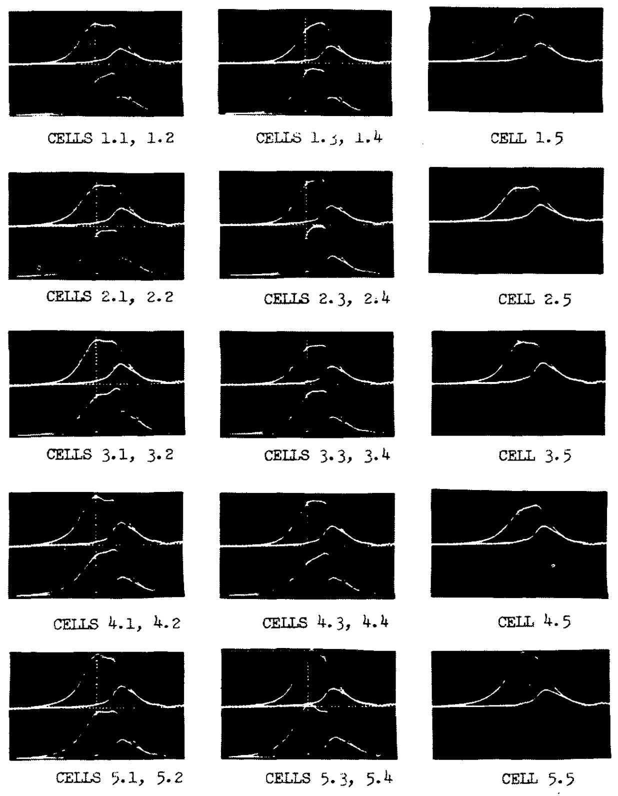


Figure 4. Uniformity Data on Screen No. 31.

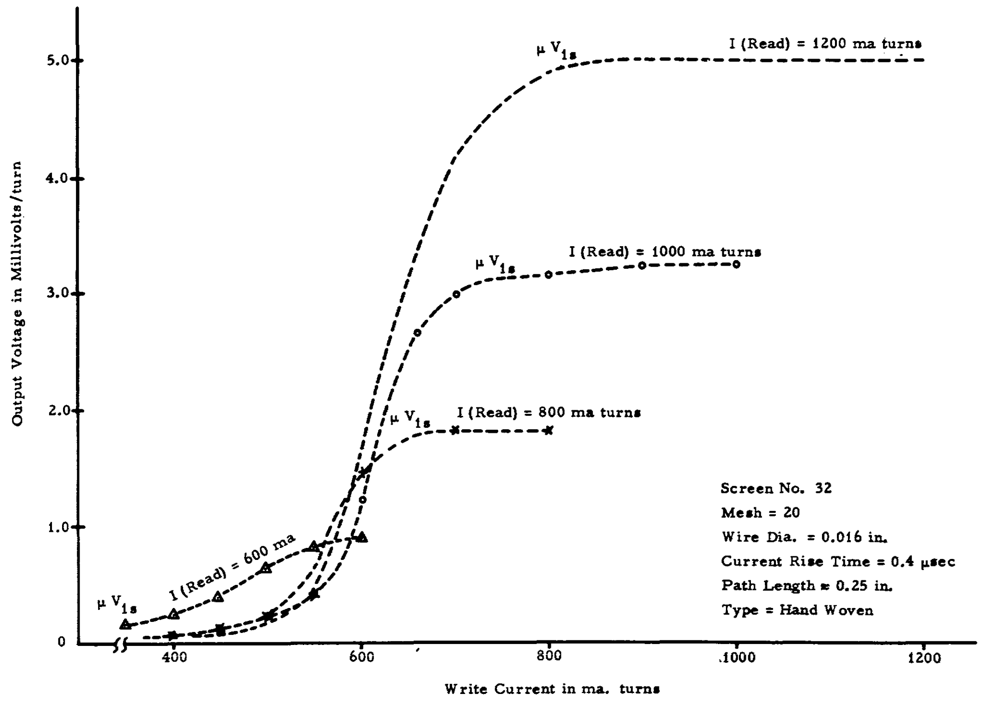


Figure 5. Cell Output Vs. Read Drive.

Each bit has an upper section plane which would share the same sense line with the plane of the lower section. In stacking the planes, during manufacture, the lower section of the first and second bit planes would be followed by the upper section of the first and second bit planes with the process repeating with the next higher bit planes. Hence, the Y digit drive line would be connected in series by jumpers at two edges of the stack assembly.

Half select magnetic noise is cancelled by storing ones in both the clockwise and counterclockwise orientations in adjacent pairs of cells. The sense winding polarity with respect to the Y drive line is woven in such a manner that the polarity changes during weaving on every odd cell. There is always a half select uncancelled cell in the X and Y coordinates during reading.

The worst case noise pattern, because of the sense winding, is identical to core planes and consists of the double checkerboard pattern.

Figure 6 shows the degree of inductive noise cancellation that was achieved in a hand woven, digit and sense, plane where the wires were not accurately positioned. Figure 6a shows a complete cycle for the case where all ones are being stored and read repetitively. The first group of signals results from the flux seversal during read time. The second group of signals are the noise pulses caused by reversing the ½ select Y drive current prior to writing. The third group of signals results from reversing the flux during writing. The fourth group of signals results from noise on the sense line which is picked up by reversing the ½ select drive current prior to reading again. The fifth group of signals is again the flux reversals during read time.

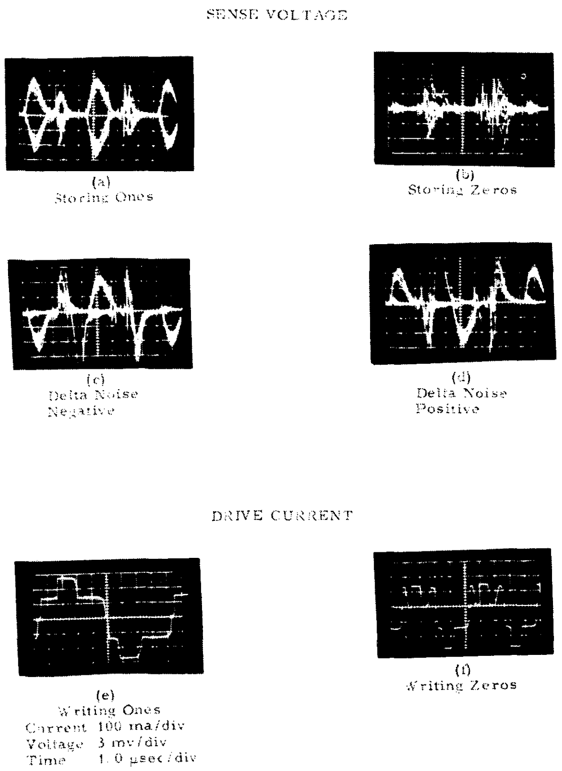


Figure 6. Read-Write Cycle.

Figure 6b shows the same sequence of signals while stored "zeros" which are repetitively being read and written. In this case, the uncancelled inductive noise is increased because of the inhibit current.

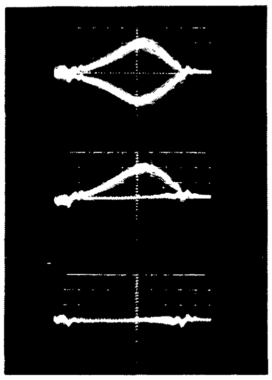
Figure 6c and 6d show the same group of signals except that both the negative and positive worst case noise patterns are being read and stored repetitively. In these two cases, magnetic noise can be seen in addition to inhibit and uncancelled Y drive line noise.

Figure 6e shows the summation of the drive currents for the pedestal coincident current technique for case writing and reading ones in the plane. Figure 6f shows the summation of the drive currents for each cell when a zero is being stored and read from the plane. The horizontal time scale in Figure 6f is 2 μsec/div to show two write cycles.

Signal To Noise Ratio

As indicated by Figure 6, the signal to noise ratio of a given cell may be much greater than 20 to 1. In experimental hand woven planes, the signal to noise ratio has been as high as 15 to 1 for over 1100 adjacent bits in a 2500 bit plane. Figure 7 shows the signal to noise ratio for 512 adjacent bits while storing the worst case noise pattern in a plane. It is expected that the signal to noise ratio for the 32,768 bit plane will be greater than 2 to 1, by dividing the sense wire in sections.

It will be noted that the switching time, ts, as shown in Figure 7 is less than that shown in Figure 3, and the output voltage is greater than that shown in Figure 3. There are several basic differences between the screens used and in operating points as shown by the readback waveforms. First, the screen used in Figure 7 was woven as a 22 mesh plane with 0.015 inch diameter wire. Second, this screen has a 5% thinner coating. Third, the drive currents selected exceed the optimum value by approximately 10% so that the cells are slightly overdriven which accounts for the disappearance of the double hump characteristic and results in faster switching times for the cells.



TOP	512 "1's"
MIDDLE	256 "1's" 256 "0's" (delta noise pattern
BOTTOM	512 "0's"

SCALE: Horizontal 0.2 μsec/cm
Vertical 3 millivolts/cm

Figure 7. Adjacent Cell Output.

Timing

The significance of the pedestal coincident current system is that it allows the designer to trade speed for equipment cost in the design of a memory system. The switching speed of the woven screen memory cell, Figure 7, indicates that the plane could be used for a small coincident current memory system with a 3 to 4 μsec cycle time. However, in a large memory system, it appears that an 8 to 10 μsec is the most desirable to minimize cost.

Figure 8 shows the basic timing system for the memory cell. The X digit pulse current should be 1.4 μsec wide at the 90% points on the plateau. The rise time of the X digit pulse current should be 0.25 ± 0.05 μsec. The fall time is not important as long as it reaches zero before the initiation of the succeeding cycle.

The Y pulse current will start changing polarity during the width of the clock pulse. The rise time on this pulse is not important. The only requirement is that the current reach maximum value not later than 2.2 μsec from the start of the clock pulse. The Y current can be turned off at clock time. If the Y drive current is gated to be switched at clock time by selecting a different word (usual case), there will be a period of approximately 1 μsec during which time two Y drive lines are activated, re-

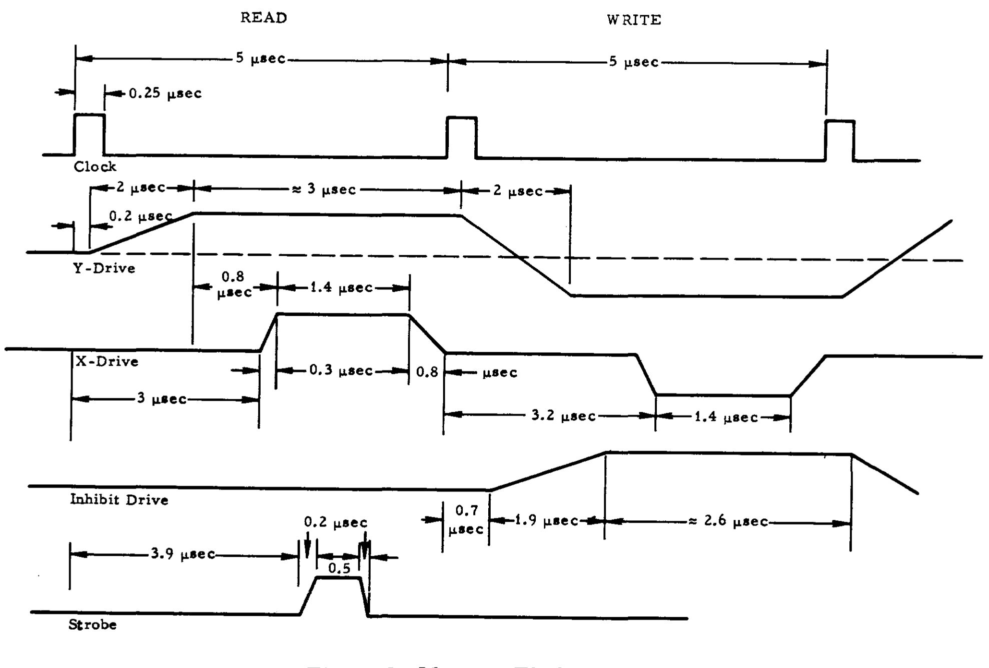


Figure 8. Memory Timing.

sulting in splitting the Y current. This change is transitory and does not degrade performance in any way because the current has 3 μsec to stabilize. This point is very important when selecting the first Y cell in high speed, time coincident current memories.

The inhibit pulse is turned on as soon as possible by the gating signals. The delay in the inhibit line for a large system may be as high as 0.7 μsec. The rise time of the inhibit pulse is also exceedingly slow. The only requirement is that the current stabilize in a maximum of 2.6 μsec.

Many of the significant cost advantages of the pedestal system can be noted in Figure 8. No special time generator is required for the Y drive current or the inhibit current. The Y drive current is reversed at clock time. The start of the inhibit current is dependent on gating delays and is turned off at clock time.

Only the X drive current and the strobe signal require specialized timing generators. The most critical rise time is on the X half selected current. The strobe rise time is the next most critical parameter. The pulse width and pulse delay on the X drive current can vary considerably without degrading system performance.

The strobe must follow the pulse delay and rise time of the X driver because of the fast switching speed of the cell. The strobe margin will be approximately 0.4 μsec.

ESTIMATED SIZE OF THE PLANES

With given average characteristics for a memory cell and a fixed cell density, the maximum possible size of a single plane is based on the following basic factors: (1) The maximum area that can be uniformly plated with the desired magnetic characteristics; (2) the magnetic quality of the individual cell; (3) the voltage and current limitations which are fixed by present semiconductors.

Uniformity of Magnetic Plating

Uniformity of plating over a large area is the most practical serious limitation. The plating process consists of five steps: (1) Electropolishing of the copper wire to remove foreign material; (2) tin plating to mask impurities in the copper substrate; (3) hot oil dip to flow the tin in order to bond the corners and produce a smooth substrate; (4) plating of a nickel substrate; and (5) plating the magnetic layer.

Almost all of these plating processes are carried out at room temperature with no agitation of the baths or electrodes. In the three element magnetic bath, pure nickel anodes have been used even though nickel, iron and cobalt ions are deposited on the screen, which is the cathode. The magnetic coating has such a high nickel content that the solution concentration of iron and cobalt ions are not significantly depleted during plating. This technique has allowed personnel in the laboratory to use a bath daily for over one month. The iron and cobalt concentration ratios are not critical.

The important control factors in the bath are current density, pH, solution concentration, volume of solution to surface area and positioning of the anodes relative to the cathode. Each of these factors can be easily determined physically before plating. Accurate cathode to anode distances can be maintained by building anode fixtures which are used in plating. PH and current density can be monitored and adjusted automatically. Standard production tanks are large enough so that a group of planes will be plated at a time.

Thirty-three square inches is the maximum surface area of planes which have been uniformly plated in small laboratory beakers. With the availability of large production tanks, it is conceivable that satisfactory uniformity can be achieved while plating planes whose cross-sectioned area exceeds 4 square feet.

Magnetic Quality

The second significant factor effecting maximum plane size is the magnetic quality of the cell. The anisotrophy of the magnetic surface is in the desired direction which is around the cell. The screens are plated without the aid of an external orientating field. The squareness, tilt and coercive force of the B-H loop are controlled by setting the pH and current density of the bath to obtain a definite magnetic alloy which is nonmagnetostrictive in addition to not aging to any degree.[4]

The coercive force of the magnetic surface has been adjusted specifically to be greater than 1.8 oersteds. Experimental planes have been operated in memory systems where the variation in the full select current has been varied ± 15%.

The maximum number of bits that can be sensed by a given sense line is basically limited by the quality of the B-H loop.[5] [6] The limiting factor is still the unbalance voltages between pairs of half selected cells which are in opposite states while storing "ones". Also, the B-H loop is not quite symmetrical with respect to positive and negative pulses.[7]

It appears that the sense line length will be primarily determined either by the input sensitivity of the sense amplifier or by the propagation delay time in the sense line. The sense line for each plane must be divided into four sections and selected pairs of planes sensed by the same line. From the results of preliminary experiments on the 2500 bit plane and 4000 bit plane, it appears possible to thread 16,384 cells with a single loop if the X axis is divided in half. Although only 512 bits were sensed in Figure 7, the length of the sense line was 64 feet.

Voltage and Generator Limitations

The general circuit limitation in any large, medium speed, array is the voltage limitation. The design of a driver for a ferrite core plane is extremely difficult because of the inductance per core in any array. 6×10^{9} henries is a general inductance figure quoted for a 50 mil ferrite core. The d. c. resistance of the line is normally low compared to the total core inductance. The impedance of the drive line changes drastically during the current pulse rise time making the design of the current source difficult. If a large number of cores are strung on a given line, the driver voltage requirement to establish the given current rise time becomes the limiting factor. Transistor drivers are commercially available today with collector to emitter voltage ratings of 100 volts.

In the woven screen memory, the impedance of the drive line is constant and almost a pure resistance. The cell inductance is negligible and, hence, the wire resistance, inductance and capacitance become extremely important.

The characteristics of the inhibit line become a major factor in limiting the practical size of a screen plane in a large memory system. The limit is imposed by the d. c. resistance of the wire and the voltage limitation of the driver transistors (neglecting impedance transformation). The maximum size of a coincident current screen plane envisioned at this time is 32,768 bits. With a cell density of 400 cells/in^2, the resulting rectangular matrix is approximately 13 inches by 6½ inches.

The length of the inhibit line per row is 1.25 feet. The total length (two planes per bit) is 320 feet. The d. c. resistance of the inhibit line is approximately 160 ohms. The wire inductance has been both calculated and measured at 100 KC and is 39×10^{-6} henries.

The back voltage on the inhibit line caused by disturbing the individual cores can be approximated for the absolute worst case where all the individual noise voltages add in phase. With a squareness ratio of the B-H loop of 0.95, the disturbed output of a given cell is approximately 0.15×10^{-3} volts. The back emf due to the cell inductance, by the inhibit driver is approximately 10.0 volts. The peak voltage required across the inhibit line to establish 240 ma of current is thus 38.5 volts plus 10 volts or 48.5 volts.

For the inhibit line mentioned, the time constant, neglecting capacity, is 0.25×10^{-6} sec. This means that at least 1.2 μsec must be allowed in the timing system to allow the inhibit current to approximate its final value.

In large memory arrays, or in extremely fast cycle time memory systems, extreme care must be paid to the transmission line characteristics of the drive and sense lines. Making several assumptions about the geometry it is possible to arrive at a reasonable approximation of the line capacity to ground. The capacity of the line is approximately 16×10^{-12} farads per foot. The wire inductance is approximately $\frac{1}{8} \times 10^{6}$ henries per foot. Assuming the shunt conductance per foot to be zero, the propagation delay time of a 320 foot line is approximately 0.45×10^{6} sec. The attenuation constant is approximately 23×10^{-3} db per foot. The characteristic impedance of the line is approximately 90 ohms and the line is slightly capacitive. The transmission characteristics of the inhibit line indicate that a ten microsecond time line is feasible.

The rise time requirements of the Y drive line is even less than that of the inhibit line as shown by Figure 8. Thus, it should be possible to design a Y driver that is capable of driving 512 planes, if desired.

The X drive line cannot be driven from a constant voltage source. A higher voltage source per given length of line is needed in X drive line because of the fast rise time required. A resistance must be placed in series with the voltage source to give a smaller time constant. The X drive line must be short in comparison with the Y drive line. Assuming a voltage limitation of 100 VDC, the X drive line can be somewhat greater than 72 planes long. The length of the X line would be 90 feet with a dc resistance of 45 ohms. The minimum drive voltage necessary is approximately 27 volts. The inductance of the X drive line would be approximately 60×10^{-6} henries. The inductanace per foot of the X and Y drive lines should be slightly greater than the inductance per foot of the inhibit and sense line because of plane and stack wiring. The capacitance and resistance per foot should be the same for all four wires. Thus, the propagation delay time for the X drive current should be approximately 0.15 μsec.

The maximum length of the sense line is limited by the impedance of the sense line, the signal to noise ratio of the worst case disturb noise signals, the output of a single cell, and the sensitivity of the input stage of the sense amplifier. In theory, regardless of the signal amplitude, the signal can be detected providing the signal to noise ratio is good. In practice, however, the input sensitivity is limited by the recovery time of the sense amplifier. The band width of the sense amplifier must be at least 2 megacycles. The saturated recovery time is on the order of 0.5 μsec. Fast recovery is needed because of the characteristics of signal to noise pulses. The signal to noise ratio is determined from time discrimination and not amplitude discrimination. The sensitivity threshold of the sense amplifier is less than the maximum single cell noise. Inhibit and Y drive noise pulses may exceed the signal amplitudes by a factor of twenty.

The output from the cell and the attentuation of the sense line determine the level of input sensitivity. It is assumed that the preamplifier or input stages of the sense amplifier will be mounted directly on the plane to reduce noise pickup and signal attenuation. From the experience gained on 2500 bit and 4000 bit planes, it is expected that the maximum number of cells on any given sense line will be 16,000 cells. Each pair of planes will have four sense lines with a preamplifier or switching network on each line before they are combined into a single line.

The length of a paired sense line should be 80 feet. The d. c. resistance is 40 ohms. The characteristic impedance is approximately 90 ohms. The propagation delay time should be 0.11 μsec. The signal attenuation is approximately 1.75 db. The input amplitude to the sense amplifier is in the 1 to 2 millivolt range.

MANUFACTURE OF THE PLANES

The plane manufacturing process has been given considerable thought throughout the company sponsored development of the woven screen memory program. The basic reason for such early production planning lies in the significant cost reduction potential of the woven screen memory concept. Each of the eight basic production steps of plane fabrication has been considered with respect to achieving a substantial unit rate.

A study of the production costs has shown that the connector is the limiting factor in establishing the costs per bit of a plane. Material costs per plane in terms of wire and chemicals are very small. The production steps are the following: (1) weaving; (2) connector fabrication; (3) assembly of a plane to a connector; (4) plating; (5) circuit assembly; (6) testing of a single plane; (7) stack assembly; and (8) stack testing. Each step in the process is followed by an electrical or mechanical inspection procedure.

Weaving

Many physical configurations have been conceived for weaving a single plane, multiple planes or complete memory stacks. These multiple plane configurations have been developed to solve or alleviate certain mechanical problems such as plane interconnections. It is easy to visualize a square memory stack which was assembled by folding a long rectangular plane. Also, it is easy to visualize weaving a large square plane and reducing it to a small cube by continually folding each axis in half. Even cylindrical shapes have been proposed.

Once a given cell configuration is chosen, the weaving pattern is established for the plane. However, it is not feasible to think only of weaving the cells, sense and drive wires while ignoring the problem of assembling the planes to the connector frame. Assembly can be made relatively easy if the weaving pattern of the plane is carefully designed.

The weaving of woven aperture screen planes may initiate the development of a new family of looms. The optimum screen memory loom will have the flexibility of the most advanced loom designed to weave cloth fabrics of the most complicated patterns. Also, the loom must incorporate many of the features of looms which are specifically designed to weave wire cloth.

Compared to cotton yarn, wire yarn is inelastic and relatively strong. Hence, a wire loom is designed to stand greater forces than a cloth loom. Also, special features are designed in the wire loom to minimize the problems of extremely low elasticity of the yarn.

Wire looms are used to weave extremely simple over and under patterns in the manufacture of wire cloth. There is a significant problem in weaving to some dimensional aperture tolerance when wires of different physical characteristics are woven simultaneously. If one of the wires is insulated, the problem is compounded because the strength and characteristics of the insulated material must be considered. The wire is permanently deformed during the weaving process. This deformation of the wire is extremely useful in achieving dimensional stability of the open apertures as each wire tends to be locked in place during weaving.

Extreme care must be maintained in weaving woven aperture screen planes so as to minimize the damage to the surface characteristics of the bare metallic wires and the insulation of the coated wires. Surface imperfections will influence the orientation of the magnetic surface during electro-deposition.[8] If the weaving forces are excessive, the insulation will be stripped and the screens will be shorted.

Many facets of the woven screen memory concept have been successfully demonstrated in experimental weaving programs. First, suitable insulated materials have been found to demonstrate that unshorted planes can be successfully woven of bare metallic and insulated copper wire. Second, path length tolerance around a cell can be held to less than 2% in weaving. Third, the design concept of selective weaving of drive and sense wires to provide easy separation from the dummy, or spacer wire, has been successfully verified in weaving. Fourth, the basic discontinuous design of the planes has been achieved. Fifth, weaving designs for cells have been accomplished even using the simple over and under plane weave pattern with a periodic odd, even seqence.

A single production loom should be capable of producing over 2.5×10^6 bits per day. In production, the yield of the planes in the weaving process is not expected to be less than 50% in the initial stages.

Assembly

The planes will be spot welded to a connector frame in less than five minutes prior to the plating cycle. Each spot weld will be subsequently tin plated and fused in the first two steps of the plating cycle. In experimental planes, the entire frame is magnetic because it is plated along with the screen.

Cost

Present estimates indicate that the cost per bit, of coincident current woven planes, in assembled stack modules, in sizable quantities, will be in the range of 0.1 cents to 1.0 cents. As the planes become larger, the cost per bit decreases and the yield decreases. Thus, a finite limit will be eventually reached. The yield on a linear select plane is expected to exceed 90%.

MEMORY SELECTION

Based upon laboratory plating experiments to date it appears possible to manufacture coincident current planes which contain 32,768 bits. The planes would be rectangular 128 by 256 matrix and two of these planes connected in series would make a 65,536 bit plane. 16,384 bits is the maximum number which has been estimated that can be sensed on a single sense line. The maximum number of planes which can be driven in series by the most critical driver, X, was shown to exceed 72 planes and may reach as high as 100 planes.

Thus, the basic memory stack module could consist of between 128 to 192 planes. There is no agreement between commercial or military computers on the number of bits per word. The popular data processing word lengths are: 30, 36, 39, 48, with each computer manufacturer emphasizing the virtues of their particular word length. The best compromise at this time for the word length appears to be 36 bits.

In any mass memory system, the memory will be composed of basic modules whose bit capacity is limited by the characteristics of the storage device or fabrication problems. The proposed woven screen memory module would contain in excess of 2^{22} bits. Current memory capacity requirements proposed by both the military and civilian computer users vary generally in the range of 10^8 to 10^{12} bits with some isolated requirements for even higher random access capabilities. However, because of the present state-of-the-art in memory devices, most of these memory requirements have been scaled down to 10^8 bits to 10^9 bits. Although the need for true random access can be debated in many individual cases, the requirements for data transfer rates is very high from a bulk storage system.

There is a significant desire, especially on the part of military users, for a true solid-state random access bulk storage memory system. This desire has been prompted by reliability or maintenance problems with electro-mechanical equipment in a hostile environment. The reliability problems with relays, especially in peripheral equipment, which has been experienced by users has significantly affected the proposed design because the possibility of module relay switching to reduce cost by decreasing access time has been eliminated.

The X and Y selection problem for the 10^8 bit memory system consists of selecting one in $256 \times m$ lines; where m is the number of modules required and is equal to 22. The selection would be divided into a $16 \times 22 \times 16$ matrix. Since bidirectional currents are required, the X and Y matrix selection consists of 352 transformers and 16 pairs of transistor current switches as shown in Figure 9.

Extreme care must be exercised in the design of the matrix to minimize the capacitance problem. The selection of the diodes is important because of the junction capacity and the high conductance required. However, circuit techniques can be employed to minimize matrix capacity and leakage shunt paths.[9]

Inhibit current is unidirectional. However, the design of the current source is a problem because of the power required. Since at least 72 planes could be energized in parallel, the current source must be capable of providing approximately 18 amps. An investigation of the problem has led to the conclusion that the inhibit selection should be split in two sections. Each selection matrix consists of a 22 by 36 selection, as shown in Figure 10. The row switch, SM1, is controlled by the module selec-

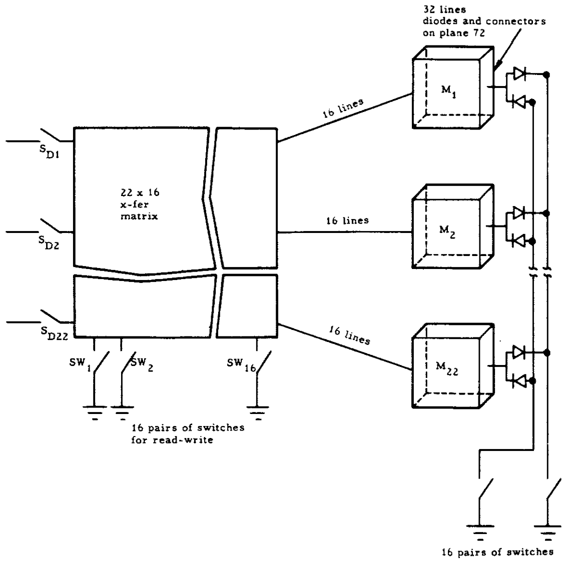


Figure 9. X-Coordinate Selection Block Diagram.

tion and is essentially a gated series regulator with a current limiting feature to protect the control element under abnormal load conditions. The column switch, En, has a 240 ma current capability.

The most difficut selection problem is the sense selection matrix. Each sense amplifier must have a minimum of 22 gated inputs. The problem is further complicated because the sense line is split into four sections on each pair of planes which make a single bit.

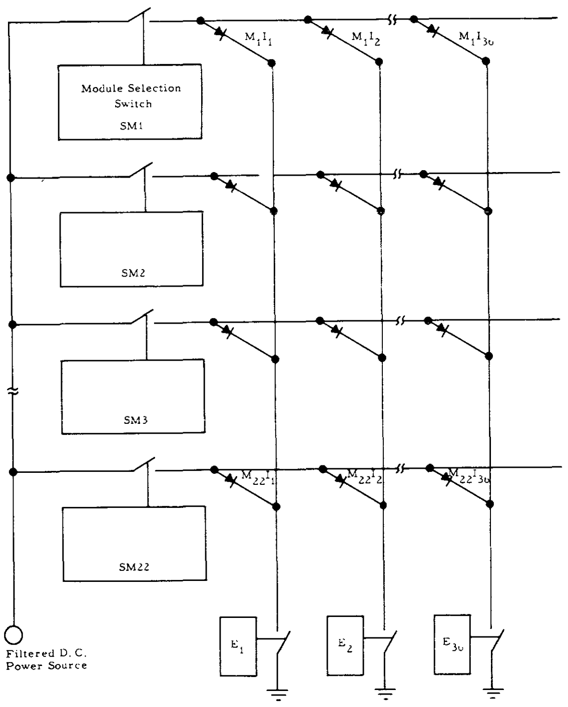


Figure 10. Inhibit Selection.

The propagation delay times in the sense line are critical. The total time delay must also include the propagation delay from the plane to the selection matrix. It appears at this time, that the total delay will be 0.3 μsec.

Early investigations of the sense selection problem led to the conclusion that pre-amplifiers must be mounted on each plane. This approach is highly undesirable because 6,336 pre-amplifiers would be needed. The outputs of the pre-amplifiers would then be gated by a tree selection matrix to 72 sense amplifier.

Subsequent laboratory experiments indicate that it may be possible to gate select the input signal even at 2 mv level. Each set of sense lines would then be selected by forward biasing selection diodes which would connect the sense line to an input transformer. Extreme care must be exercised in selecting the gate diodes and in designing the transformer. The diode unbalance will saturate the amplifier. If sufficient settling time is allowed and the sense amplifier is properly designed, the amplifier will recover in time to read the first pulse. The settling time required is on the order of 4 μsec. Fortunately, the sense line resistance is high enough to insure that both selection diodes are conducting. Sufficient tests have not been conducted at this time to insure that 88 sense lines can be paralleled.

The component count of the selection techniques proposed can be estimated very closely for the current driver matricies.

Type	*Diodes*	*Trans-formers*	*Current Switches*	*Power Switches*	*Sel. Dr.*
X-Selection	11,968	352	64	0	22
Y-Selection	11,968	352	64	0	22
Inhibit Selection	1,584	0	72	44	0

The module selection diodes would be mounted on a plane and become part of the stack module. The selection electronics would be mounted in a driver module. The stack modules would plug into the driver module.

SYSTEM DESIGN

Extreme difficulty has been encountered in attempting to consolidate various ideas and system design requirements into a valid set of system specifications for a true, fast, random access solid state, bulk memory system. The fundamental obstacle to overcome is one of cost over desirability. Most users have been conditioned to view bulk storage devices in terms of magnetic tape storage, disc file storage, or magnetic card storage. The cost and operational advantage of tape storage systems have so far outweighed their extremely slow access times. Disc storage and even drum systems are desirable in some data processing complexes because of the faster access times. The magnetic card storage should fall in user desirability, because of access times, somewhere between the two competitive systems.

Random access core memory bulk storage systems have never really become acceptable because of the cost factor. A 10^8 bit core memory system would probably cost between 3 and 9 million dollars. The bulk of this cost is in the core stacks. The magnetic tape, card or disc storage systems would cost between 50 and 500 thousand dollars. The cost of the woven screen and possibly super conducting memory systems[10] should overlap both ends of the spectrum.

The two fundamental system design problems with true random access in a bulk storage system are speed and auxiliary features. Once potential users are accustomed to true random access, the tendency is to require a data rate that is equal to the exisiting computer memory. In addition, there is a strong tendency to request multiple input-output channels and elegant indexing features. Each of these features increase the control or input-output electronics. The possibility then exists of designing at least three different systems all with the same capacity.

System I could be organized to be a minimum cost version where the access time was 10 μsec and the data rate was 10 μsec per character. This system would have a single input-output channel. Block information could be transferred by initial address selection and a termination signal.

System II could be organized as an intermediate cost system with 10 μsec access time and a data rate of 10 μsec per 36 bit word. The addressing system is more flexible to facilitate block transfers. A single input-output channel would still be provided.

System III would then be organized to give maximum data rates and flexibility. The access time would be 8 μsec and the data rate 4 μsec per 48 bit word. A minimum of four, two speed, input-output channels could be provided with a symbolic addressing feature to ease programming problems. Simultaneous computer and tape, loading and unloading, would be provided.

System II appears to be the most practical system to look at in some detail. The control module would contain a 36 bit input-output register, together with a 22 bit address register counter and a 16 bit block counter. The memory would always be connected to the computer unless manually switched to tape units for loading or unloading of the memory. Thus, the memory control commands could easily be handled by a five bit control register which included parity. Special input-output amplifiers would probably be needed to match external impedance and signal levels. With existing commercial packaging techniques, the control electronics could easily be packaged in less than 12 cubic feet.

A separate power supply module would be provided to hold at least five supplies whose total power output would be about 2500 watts.

The memory unit would consist of the 22 stack modules which would plug into the driver module. Each stack module would be 14 inches by 14 inches by 7 inches. The driver module would be 42 inches x 56 inches x 10 inches. Thus, the total size of the memory unit would be 3½ feet x 4⅔ feet x 2 feet.

The following electronics would be mounted in the driver module; two 352 transformer selection matrices, each with 704 diodes; 164 current switches for X-Y and inhibit selection; 44 selection drivers for X and Y selection; 22 power switches for inhibit selection; 1,584 diodes for inhibit selection; 36 sense amplifiers and possibly 6,336 pre-amplifiers, diodes and resistors for sense line switching; 2 timing generators; 1 clock amplifier and 1 clock oscillator. It is slightly premature to estimate component count any closer because of the uncertainty in the sense amplifier area.

It appears that the various government agencies are being coordinated on this system design problem so that the first true set of system specifications may appear in the near future. The characteristics of the commercial systems would probably follow the military specifications. Few business commercial data processing systems are sufficiently large in volume and need to justify a specialized set of system specifications.

HIGH DENSITY PLANES

The 100 mesh weave plane which has been proposed has been experimentally fabricated with existing commercial looms. However, looms have already been designed which are capable of weaving over 1000 mesh planes. It is feasible to think of weaving 500 mesh planes. Such a plane would be capable of storing between 10,000 bits/in^2 to 27,000 bits/in^2. Experiments indicate that a buffer cell may not be required.

A 500 mesh plane would be woven with 1×10^{-3} inch diameter wire. The path length would be 24×10^{-3} inches. If a plane were folded, storage could be provided with a density of approximately 4×10^{6} bits/in^3. The connection problem becomes severe and may be impossible. A further material problem is encountered with the insulated wire. Machinery will probably have to be designed to coat 1×10^{-3} inch diameter wire properly.

Past experience indicates that it should be possible to electro-deposit on a woven matrix of this type. However, the plating parameters appear to be a function of the wire diameter. Since the alloy and stress condition will change with pH and current density,[11] there are some questions as to the quality of the B-H loop that can be achieved.

CONCLUSION

Sufficient laboratory data has been accumulated to determine the characteristics of the cell, drive and sense lines on coincident current hand woven and machine woven planes. Good uniformity has been achieved on 36 in^2 planes and small sample planes[12] which have been hand woven. It appears reasonable to expect comparable uniformity on 72 in^2 machine woven planes.

A cycle time of 10 μsec has been determined to be a reasonable value. The drive selection techniques have been formulated but the sensing selection needs additional investigation.

Sufficient data has not been accumulated to solidify the system design because of the lack of firm specifications.

ACKNOWLEDGEMENTS

The author wishes to gratefully acknowledge the technical assistance of R. A. Howard; of L. Menigoz on sense amplifier design problems; of R. E. Lee and D. R. Boles who patiently prepared the hundreds of screens, and who meticulously tested each bit; and A. D. Scarbrough without whose support this work could not have been possible. The author is also indebted to L. Miller and Dr. R. B. Kershner of APL, JHU, who supported a program to determine the feasibility of fabricating coincident woven screens on APL/JHU Subcontract No. 181305.

REFERENCES

1. R. A. HOWARD, P. E. WELLS, L. CANN and J. S. DAVIS; "Investigation of Woven Screen Memory Techniques." Large Capacity Memory Techniques for Computing Systems, Macmillan, 1962, pp. 361-373
2. J. A. RAJCHMAN "Ferrite Apertured Plate for Random Access Memory." Proceedings of the Eastern Joint Computer Conference, December, 1956, pp. 107-115
3. D. A. MEIER, "Millimicrosecond Magnetic Switching and Store Elements." Journal of Applied Physics, April 1959, pp. 45S-46S
4. R. C. FLAKER, F. J. KASPER "Long Term Aging of Thin Magnetic Films of Nickel-Iron." Proceedings of the Intermag Conference, April 1963, pp. (12-6-1) - (12-6-3)
5. J. R. FREEMAN "Pulse Responses of Ferrite Memory Cores." IRE Wescon Convention Record, 1954, pp. 50-61.
6. J. D. CHILDRESS, "Noise Problems in the Coincident Current Memory Matrix." Proc. AIGE Conference on Magnetism and Magnetic Material, 1955, pp. 210-218
7. R. M. BOZORTH, "Ferromagnetism." D. Van Nostrand Company, Inc. 1955, pp. 177
8. A. G. GRAY, Editor, "Modern Electro-Plating." John Wiley and Sons, 1953, pp. 36

9. S. TAKAHASHI, S. WATONABLE, "Capacitance Type Fixed Memory," Large Capacity Memory Techniques for Computing Systems, Macmillan, 1962, pp. 53-62.

10. J. A. RAJCHMAN, "Computer Memories—Possible Future Developments." RCA Review, Volume XXIII, June 1962, 137-151

11. R. S. SMITH, "Measurement of Crystallite Size and Strain of Electroplated Films." IBM, Journal of Research and Development, Volume 4, Number 2, April 1960

12. J. J. CARR, R. F. MUNNICK, "Evaluation of Thompson Ramo Wooldridge (TRW) Woven Screen Memory Plane AMCMS 5522.11.441 DA Project 513-01-008, Technical Note TN-1112, Frankford Arsenal, April 1963

A NEW HIGH DENSITY RECORDING SYSTEM: THE IBM 1311 DISK STORAGE DRIVE WITH INTERCHANGEABLE DISK PACKS

J. D. Carothers, R. K. Brunner, J. L. Dawson, M. O. Halfhill, and R. E. Kubec
General Products Division Development Laboratory
International Business Machines Corporation
San Jose, California

I. INTRODUCTION

The development of the 1311 Disk Storage Drive began with the realization that although systems were available to meet the data processing requirements of many businesses, these systems were too big for small industries and businesses. Files which had been announced or were in development up to this time had large capacities with fast data handling capabilities but could not meet the low cost objective necessary for the smaller users.

To meet the needs of the smaller user, a development program was initiated in 1960 to provide a file-oriented low-cost system. The objectives of this development program were to maximize file capacity and its data handling capabilities while maintaining a size and cost of the file system compatible with the smaller user.

Application studies made for the smaller business areas indicated that a basic file unit capacity of 2 million characters were required. Different storage configurations was investigated and finally the idea of making a storage unit with removable disks became dominant. With this approach the user is provided with virtually unlimited off-line storage at a low cost.

The end product of this development program was the IBM 1311 Disk Storage Drive (Fig. 1), a memory device which is a slave to the using system. Its functions are to access, write, or read as directed by the computer control signals. To perform these functions, the file contains: (1) A hydraulic-mechanical accessing system which locates the read-write heads in a given position over the magnetic disk surfaces; (2) 10 read/write heads; (3) Interchangeable disk packs, each containing 10 recording disk sur-

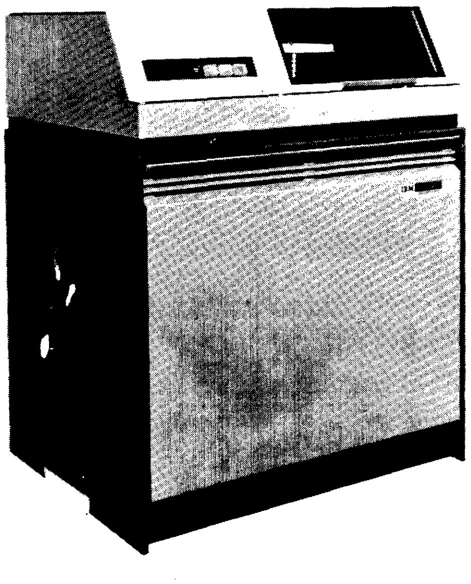


Figure 1. Basic IBM 1311 Disk Storage Drive: a) Photo

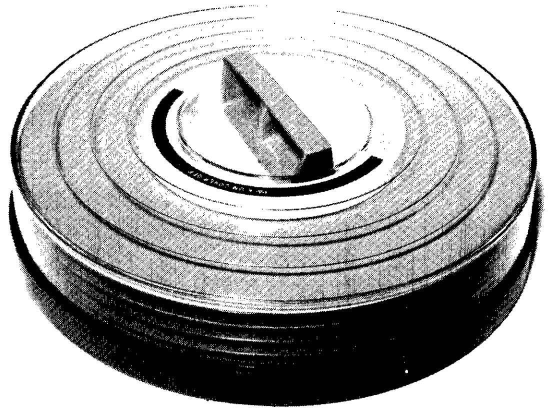


Figure 1. Basic IBM 1311 Disk Storage Drive: b) Disk Pack

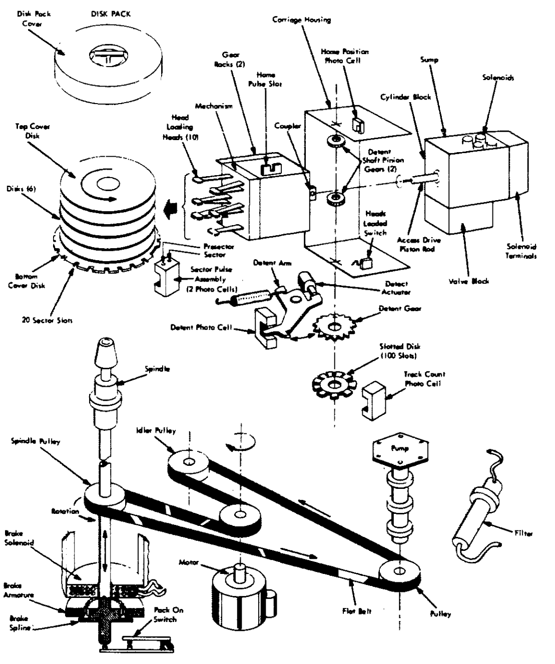


Figure 1. Basic IBM 1311 Disk Storage Drive: c) Schematic

faces (6 disks); (4) A motor which turns the disk pack and provides power for the file hydraulics; (5) Read/write electronics and control electronics; (6) Head selection diode matrix and circuits; and (7) A clocking reference system for recording data on, or retrieving stored data from the disk surfaces.

The accessing system moves the 10 heads as a unit over the 10 disk surfaces on a line parallel to the disk radius. The rotational motion of the disk pack at 1500 RPM provides a thin air bearing on which a head glides over the disk surface. There is no physical contact between head and disk.

Each disk surface is divided into 100 concentric magnetic tracks (50 TPI), and each track is divided into 20 sectors of 100-character (900-bit) records each, or into one sector of 2, 980-character records. Since the length of the inside track is less than that of any other track, the linear bit density is greatest on the inside track (1000 BPI). As many as five files can be attached to a system, providing up to a maximum of 15 million characters of on-line storage. The 1311 Disk Storage Drive also allows interchangeable disk packs. The interchangeability is achieved at a storage density of up to 50 $\times$ 10^3 bits per square inch with non-contact magnetic recording.

This paper describes the primary file technologies which were developed to meet the requirements of the machine described above.

II. INTERCHANGEABILITY

Whereas removability signifies the ability of the disk storage drive to read disk pack information previously written on this same disk storage drive, interchangeability denotes the ability of *any* disk storage drive to read disk pack information previously written by *any other* disk storage drive, regardless of which system created it.

Though the concept of interchangeability permits unlimited off-line storage for the system and intercommunication between systems through the media of the disk packs, it brings with it a host of engineering problems, such as mechanical registration, electrical tolerances, human factors, and contamination.

Mechanical Registration

If the disk storage drive is to read and write information reliably from different drives, then all heads for all tracks must be positioned accurately to within ± 0.002 inch of their ideal track locations. This means that the summation of all manufacturing tolerances, wear tolerances, tolerances due to temperature variations, and magnetic head adjustment tolerances must be considered and controlled within this tolerance. Section IV briefly considers the engineering aspects of this problem.

Electrical Tolerances

The parameters affecting reading or clocking of data, such as motor speed variations (±1.86%), read/write oscillator variations (±0.75%), as well as the magnetic variations between different disk packs and different magnetic elements (Section III), of all disk storage drives must be considered.

Human Factors

Besides these machine factors are the human engineering factors attendant with the removable disk pack. Factors such as size, weight, ease of installation and removal, as well as the requirements for protection of surfaces during handling and storage, lead to major pack design considerations. These considerations led to the final disk pack configuration (Fig. 1) which weighs less than 10 pounds, and contains six 14-inch disks.

Contamination

Accompanying removal and storage capability of the disk pack is the exposure of the disk surfaces to dirt and other foreign contaminants. These problems were recognized and eliminated in the design of the disk drive environment and the disk pack covers. A review of this problem is contained in Section V.

Central Design Concept

Interchangeability is recognized as the major objective in the engineering design of the system. As this fact was recognized early in the development program, the major engineering decisions of the development program followed directly from considerations relative to the interchangeability feature of the machine.

III. RECORDING SYSTEM ELECTRICAL PARAMETERS

Since a file system may include from one to five files, low cost is achieved by sharing one set of read electronics and a read/write clock. The master file houses these circuits. During the read process, millivolt signals are transmitted from a preamplifier on a satellite to the master file where they are amplified and detected. In a write mode, the clocked write data is sent from the master clock to the satellite write amplifiers. The interchangeability concept exposes the electronics to a wide range of signal amplitudes and waveforms because of the large number of head and disk combinations.

The major nominal operating conditions are:

Disk Surface Speed	700 ips (inside track) to 1020 ips (outside track);
Air Bearing	125 microinches (inside track); to 162 microinches (outside track)
Tracking	±0.002 inch,

and the bit rate is set at a nominal 700 kilobits per second to obtain 1000 bpi at the inside track.

The function of the Read/Write System is to transfer data to and from the disk packs. To transfer data reliably, the components of the Read/Write System (i.e., disks, heads and electronics) must perform reliably within the tolerances dictated by the over-all system. The discussion to follow deals primarily with the three parameters most important to the Read/Write System:

Head Signal Amplitude	3.5 to 28 MV;
Noise	
Bit Shift	43% maximum.

Figure 2 shows the dependence of these primary parameters on some of the other important parameters affecting the system.

Amplitude

Even though the electronics were designed to operate over a large range of amplitudes (3.5 to 28 MV), the mean values and limits of

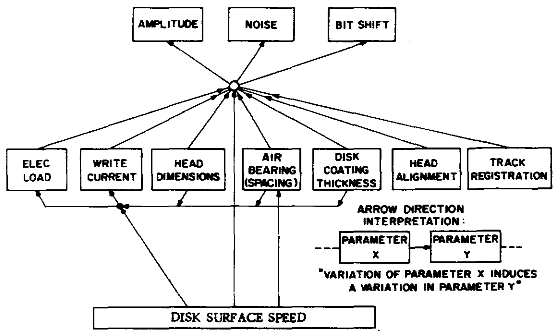


Figure 2. Interdependence of Recording System Parameters.

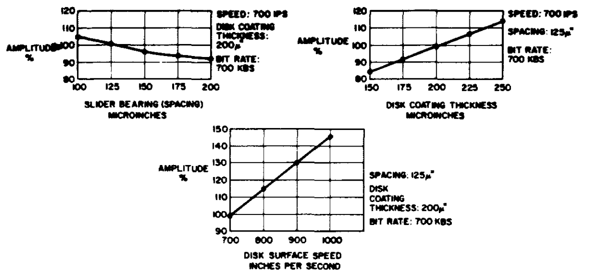


Figure 3. Amplitude as a Function of Three Parameters.

the secondary parameters were chosen carefully so that signal amplitude would meet requirements when a secondary parameter was at a worst case condition.

The nominal amplitude will vary as the head is moved over the disk surface. The disk coating is thicker at the higher surface-speed location; however, the air film thickness increases with speed, stabilizing signal amplitude. Summation of all three parameters shown in Fig. 3 gives about a 50% increase of outside over inside track head signal values. Since the disk coating thickness also varies from disk to disk, nominal amplitudes will be increased or decreased accordingly. Variation in head to disk spacing due to disk ripple and head characteristics, and changes in disk spindle speeds will also vary the signal amplitudes. Any track misregistration results in a reduced amplitude. At the worst case (0.004 inch, due to write +0.002 inch and read —0.002 inch), the signal amplitude will be 66% of its nominal value using the 0.010-inch wide Read/Write head (Fig. 4).

Another major variation in amplitude is caused by mechanical tolerances of the magnetic

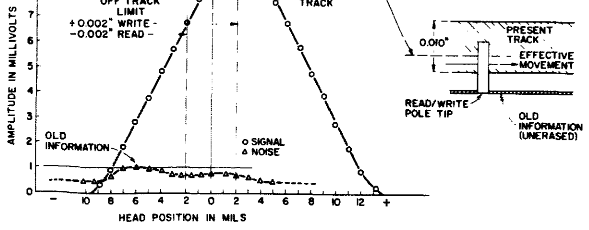


Figure 4. Head Tracking Character:stics.

head. Although close controls are placed on coil dimensions, lamination material quality, and magnetic core shape, subsequent tolerance buildups in head manufacturing affect amplitude. For example, a ±0.001 tolerance of a 0.0085-inch gap height (Fig. 5) causes a variation in amplitude of from 10 to 12 millivolts on the inside track and from 15.5 to 17.5 millivolts on the outside track.

In this head design the signal amplitude variation was reduced by shunt peaking the head. This peaking flattened the frequency response over the usable range. Figure 6 shows the frequency response at the inside track using 27, 175, and 250 pfd shunts. A 175-pfd capacitance was chosen.

In summary, the head, disk coating, air bearing, and electronics were designed such that in worst case tolerance conditions the allowable signal amplitude limits would be 3.5 MV to 28 MV (8 to 1 signal range).

Noise

Noise is defined here as any part of the head signal not contributing to meaningful data. Noise is of two types: (1) that introduced by magnetic and electrical variation and (2) that introduced by mechanical variation. In the readback system used, amplified and automatic gain controlled NRZI head signals are sensed by a level detector. If individual signal peaks were to fall below the sensing level, bit dropout would result. If noise in the absence of signal peaks raised above the detection level, bit pickup would occur. The discrimination of the detection circuit depends only on the amplitude of the maximum noise and the minimum signal presented to it, not on their ratio. Therefore, signal-to-noise ratio is used below as an indication of performance only. The following table gives the observed effect on signal-to-noise ratio when individual parameters are increased from their nominal values within their tolerance limits.

Parameter Increased	*Effect on S/N Ratio*
Shunt Peaking Capacitance	Decrease
Disk Surface Speed	Decrease
Disk Coating Thickness	Increase
Air Film Thickness	Increase

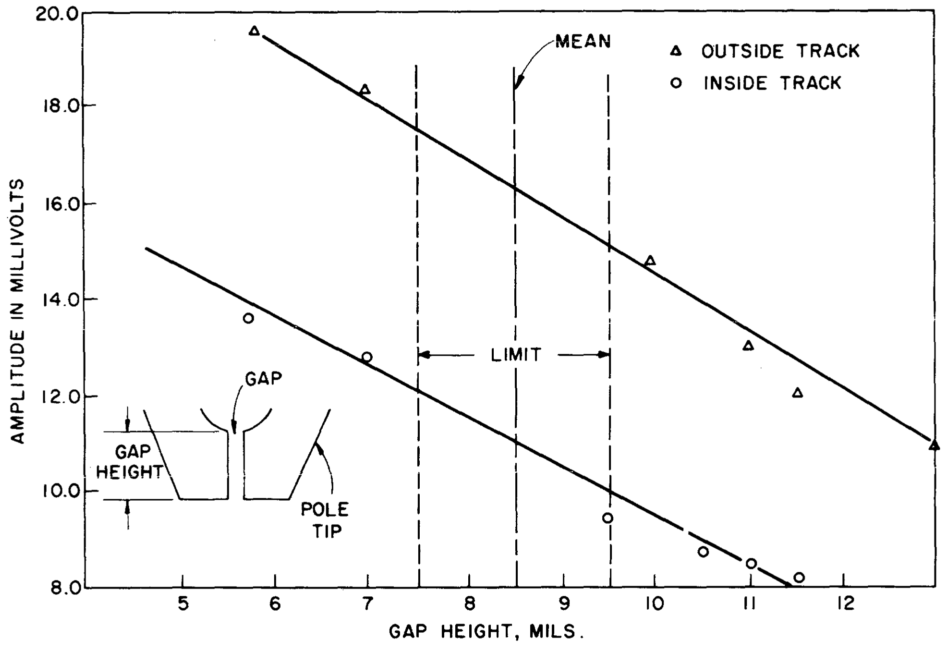


Figure 5. Gap Height vs. Amplitude.

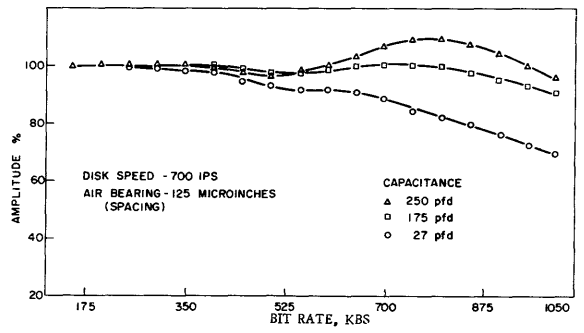


Figure 6. Effect of Head Shunting Capacitance on Frequency Response.

When shunt peaking capacitance or disk surface speed is increased from the optimum, the head signal will tend to ring, not greatly increasing the amplitude but significantly increasing the noise. This is shown in Figs. 7a and 7b. Figure 7a is a photograph of a normal inside track (700 ips) signal. Figure 7b is the same head at the outside track (1020 ips).

An increase in air film thickness or disk coating reduces the resolution of the head and decreases the ringing of the head signal, therefore giving a increase in S/N ratio.

In addition to the parameters mentioned above, tracking tolerances, pole tip alignment tolerances within the head, and variations in the disk coating degrade the signal.

Noise caused by variations in the disk coating is of two types: (1) The background grass noise resulting from such factors as surface roughness of the substrate and (2) The noise attributed to discontinuities, small air bubbles, and oxide conglomerates. These variations are held within the nominal disk coating test specifications.

Off-track registration of the head contributes to lowering of the S/N ratio by decreasing the signal level, by partial erasure from the adjacent track, and by pickup of old information noise left by incomplete erasure. Variations in amplitude and noise generated by a lateral mechanical variation are shown in Fig. 4. The profile of the signal is that of the most recently written track. The noise shown is composed of unerased previous data and inherent head noise (ringing, etc.).

In conclusion, the system component limits and mean values were chosen to insure that in worst-case tolerance build-ups, including the electronics, the minimum signal would be suffi-

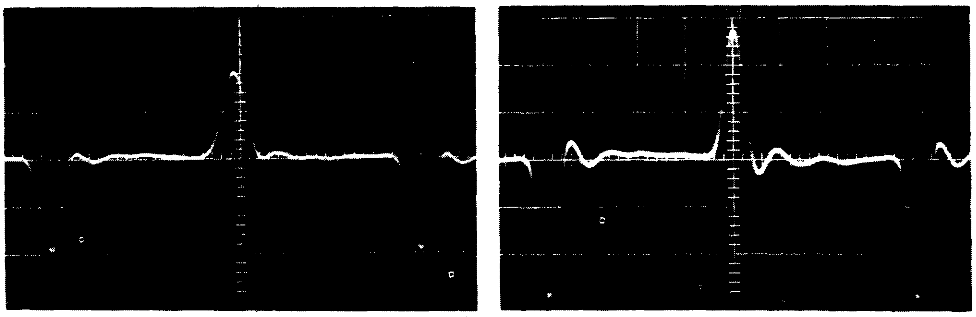


Figure 7. Normal IBM 1311 Head Signals.

ciently greater than the maximum noise, to permit detection.

Bit Shift

Bit shift is the timing variation between the read data pulses and the read clock pulses. The read/write system was designed so that the maximum range of bit shift could be allowed with no degradation of performance. The theoretical bit shift which can be accommodated by the low-cost clocking system used is ± 50%.

In the writing process, data is strobed by a free-running clock, serially by bit, serially by character. In the readback process, each data pulse resynchronizes a clock to provide a strobe for the detection of the data. The recorded data is coded so that the clock runs no more than 5 oscillations before being resynchronized by data.

The timing variation (bit shift) between the read data and the clock pulses is dependent upon the relative clocking oscillation frequency and relative disk rotational speed during writing and reading, upon the bit pattern written, and upon the bit shift caused by the head, the disk, and the electronic circuitry.

For example, the worst case bit shift occurs in the case of a five zero-bit gap with several one-bits written before and after the gap and under the following conditions:

(1) The writing is done with the oscillator frequency high (+ 0.75%) and the disk drive motor speed slow (—1.86% due to 0.5-cycle frequency variation and to 10% line voltage variation); the readback takes place with low oscillator frequency (—0.75%) and fast motor speed (+1.86%).
(2) The head bit shift due to bit crowding is maximum, 9%.
(3) The write amplifier bit shift is 2% (max.).
(4) The read amplifier bit shift is 6% (max.).

The total bit shift is therefore 5[2(0.75) + 2(1.86)] + 9 + 2 + 6 = 43%.

That is, the time between the last free-running strobe pulse and the next resynchronized pulse is 57% of the nominal bit cell time. This instantaneous bit shift can be clocked by the file circuit.

Read/Write Electronics

The write amplifier, read amplifier and clock circuit each perform a specific function in the NRZI transfer of data to or from a disk surface.

The Write Amplifier circuitry consists of a dc erase circuit (which supplies a dc current to saturate and polarize the disk coating) and a differential current driver (which supplies write current to the heads).

The Read Amplifier differentiates between valid signals and noise and presents the shaped signals to the clock. The circuitry consists of a linear amplifier and a threshold sensing detection circuit. The amplifying portion of the readback electronics is automatic-gain-controlled to provide a constant-amplitude output (± 3%). It is comprised of four ac-coupled linear differential gain stages with a maximum amplification of 72db.

The AGC feedback loop has a gain stage and a filter. The gain stage is present so that the amplifier output changes only a few millivolts for any change in the input.

To understand the detection circuit, the relationship between signal and noise at the head must be established. Only short term (bit to bit) variations in the signal are described, since the AGC smooths out the long term variations. Since a pulse indicates a binary one and the absence of a pulse indicates a binary zero, the detection circuit must be able to differentiate between the two conditions. (See section on noise.) Noise, such as unerased information or that caused by discontinuities on the disk surface, decreases the amplitude of written bits and/or appears on the baseline as pulses in the space occupied by zeros (absence of bits). To insure error-free operation, the detector has been designed to respond to amplified head signal peaks over 53% of the normal peak amplitude and not to respond to peak signals less than 43%. These levels correspond to the minimum signal and maximum noise as shown in Fig. 8.

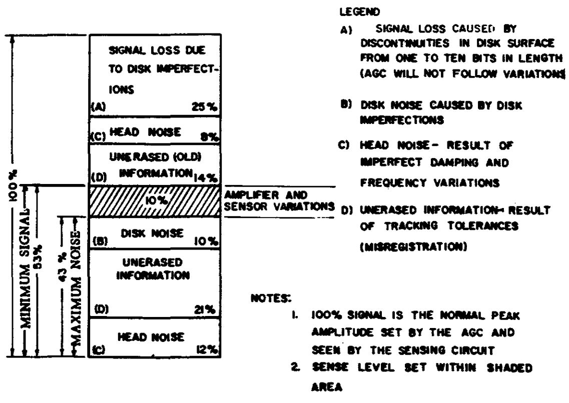

SIGNAL LOSS DUE TO DISK IMPERFECTIONS
(A) 25%
(C) HEAD NOISE 8%
UNERASED (OLD) INFORMATION
(D) 14%
10%
AMPLIFIER AND SENSOR VARIATIONS
(B) DISK NOISE 10%
UNERASED INFORMATION
(D) 21%
HEAD NOISE
(C) 12%
100%
MINIMUM SIGNAL 53%
43% MAXIMUM NOISE
LEGEND
A) SIGNAL LOSS CAUSED BY DISCONTINUITIES IN DISK SURFACE FROM ONE TO TEN BITS IN LENGTH (AGC WILL NOT FOLLOW VARIATIONS
B) DISK NOISE CAUSED BY DISK IMPERFECTIONS
C) HEAD NOISE - RESULT OF IMPERFECT DAMPING AND FREQUENCY VARIATIONS
D) UNERASED INFORMATION - RESULT OF TRACKING TOLERANCES (MISREGISTRATION)
NOTES:
1. 100% SIGNAL IS THE NORMAL PEAK AMPLITUDE SET BY THE AGC AND SEEN BY THE SENSING CIRCUIT
2. SENSE LEVEL SET WITHIN SHADED AREA

Figure 8. IBM 1311 Head and Disk—Worst Case Signal and Noise Distribution.

The detector provides data pulses coincident with the signal peaks sensed. These pulses are used to synchronize the clock.

The clock gives a time reference (or interpretation) both to write and read data. It is composed of two gated LC oscillators, a delay line, and the logic necessary to gate or synchronize the oscillators. During writing, only one oscillator is used in a free-running mode. During the read process the two oscillators are alternately turned on in synchronism with the data. The synchronized clock pulses are used to strobe the data.

In summary, because of the wider tolerances inherent with interchangeability, the read/write electronics must accommodate a broad range of signal amplitudes, bit shift, and signal to noise ratios. However, by using AGC, level sensing, and a synchronized clock, the file operates reliably for the different amplitude and timing conditions encountered in the worst case systems environment.

IV. RECORDING SYSTEM MECHANICAL PARAMETERS

Air Bearing Technology

The decision to use a recording density of 1000 bits per inch with an interchangeable disk pack introduced two problems in the air bearing area: (1) The high bit density made it necessary to operate the air bearing at a film thickness less than the 250 microinches for which the slider bearing was originally developed; (2) The combination of high density recording together with disk pack interchangeability necessitated tighter control on the orientation of the magnetic head relative to the recorded track.

The engineering approach to each of these problems is discussed with reference to Fig. 9,

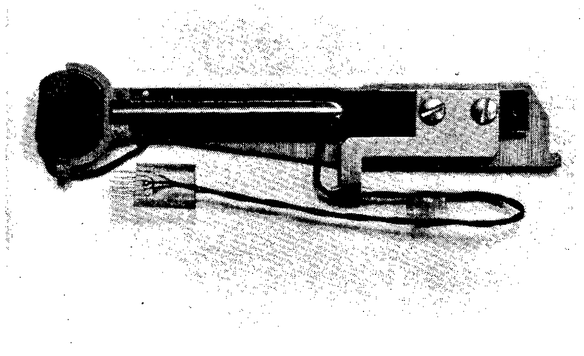


Figure 9. Complete 1000 BPI Head and Access Arm Assembly.

which illustrates the complete 1000 BPI head and access arm assembly.

1000 BPI Bearing Design

Recording at 1000 BPI made it necessary to reduce the inner track head-to-disk spacing of the original slider[1] to 125 microinches at a surface velocity of 700 inches per second.

During the development of this slider bearing, three approaches were considered to reduce the film thickness:

(1) To increase the bearing load on the original slider bearings and hence to reduce the clearance between head and disk.

(2) To design a smaller slider.

(3) To retain the original design but to reduce the film thickness by placing vent holes in the slider.

Engineering evaluation of these three approaches showed, however, that approach (1) required a loading of 30 pounds for 10 slider bearings and would cause severe machine problems. Approach (2) required a redesign of the slider bearing with attendant costs in machining and tooling. Approach (3) offered the most advantage from an economic and engineering point of view.

A series of experiments were made to evaluate the performance of a vented slider. The tests were started with a single hole in the center. The desired film thickness was easily obtained by varying the hole size, but the point of minimum film thickness did not coincide with the location of the recording element. Thus the vent hole was moved forward (to reduce the inclination) so that the minimum film thickness would occur in the region of the pole tips.

After a few experiments it became apparent that the specified magnitude and location of the minimum film thickness would be obtained, but the location and size of the hole would interfere with the magnetic element. Hence two vent holes were introduced and were positioned experimentally. The final design specified two 0.070-inch diameter holes in the forward part of the slider. This configuration is shown in Fig. 10, which also shows that the experimental and computed load-spacing curves agree within 10 percent.

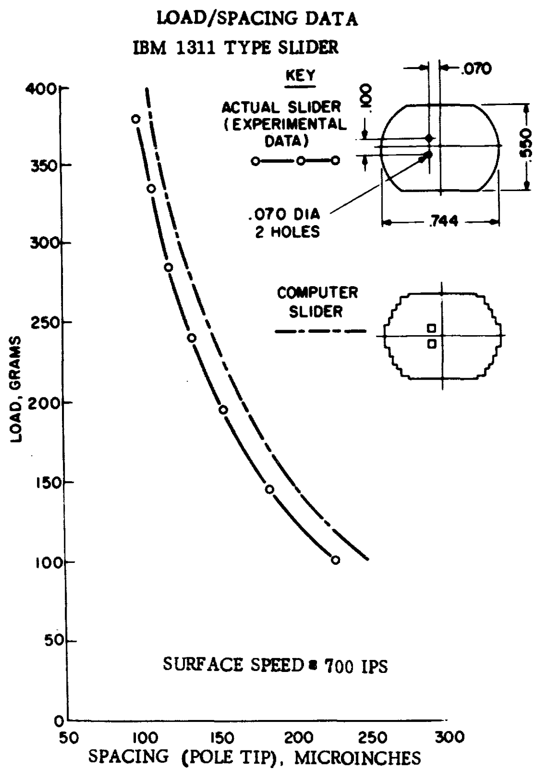


Figure 10. Computed and Experimental Load Spacing Curves.

This slider bearing gives the specified film thickness at the inner track, and the minimum point is in the region of the magnetic element. The dynamic characteristics of this design were superior to the same slider without vent holes because of higher spring rate (slope of the load-film thickness curve at the operating point), better damping due to the smaller film thickness, and a slightly lower mass. Figure 11 indicates how the vent size affects the film thick-

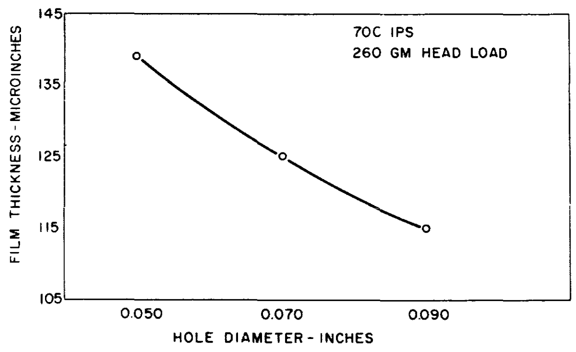


Figure 11. Film Thickness vs. Vent Hole Diameter.

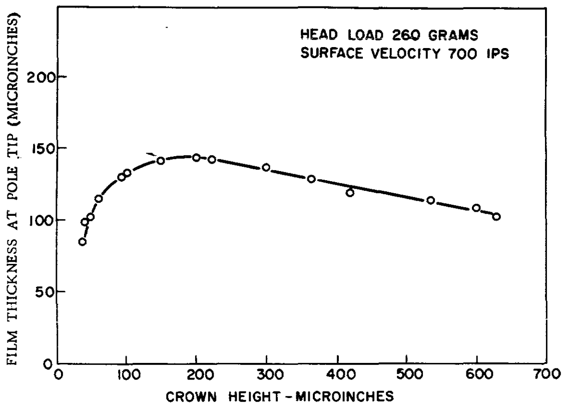


Figure 12. Film Thickness vs. Crown Height.

ness. In the range of the tests, the change in spacing with hole diameter is approximately linear. Figure 12 indicates how the magnitude of the crown height affects the film thickness. Note that the film thickness for a given load decreases rapidly when the crown height drops below 100 microinches. Gas lubricated slider bearings become unreliable at very small crown heights.[1] Figure 13 shows the computed pressure distribution for the slider used in the IBM 1311 Disk Storage Drive.

1000 BPI Suspension System

The high density recording and the interchangeable disk pack feature of the IBM 1311 Disk Storage Drive make the angular orientation of the head relative to the recorded track extremely critical. The line connecting the centers of the read-write and erase probes must be held closely tangent to the track, or a high noise level will result from incomplete track erasure.

The heads in the IBM 1311 are offset on each side of a radial line of the disk. Since the head intersects each track at a different radius, the tangent angle is different at each track. The maximum angular error due to this effect is minimized when the head is oriented so that the error is the same at the inner and outer tracks. On the 1311 this situation occurs when the angular orientation of the head is such that the center-line of the read-write and erase probes is tangent to track 60. It was determined that in order to obtain optimum performance from the heads, the mount would be required to hold the center line of the head pivots parallel to the tangent to track 60 within a maximum angle of 0°24′.

A study of an early four-pivot gimbal suspension system for the slider indicated that the

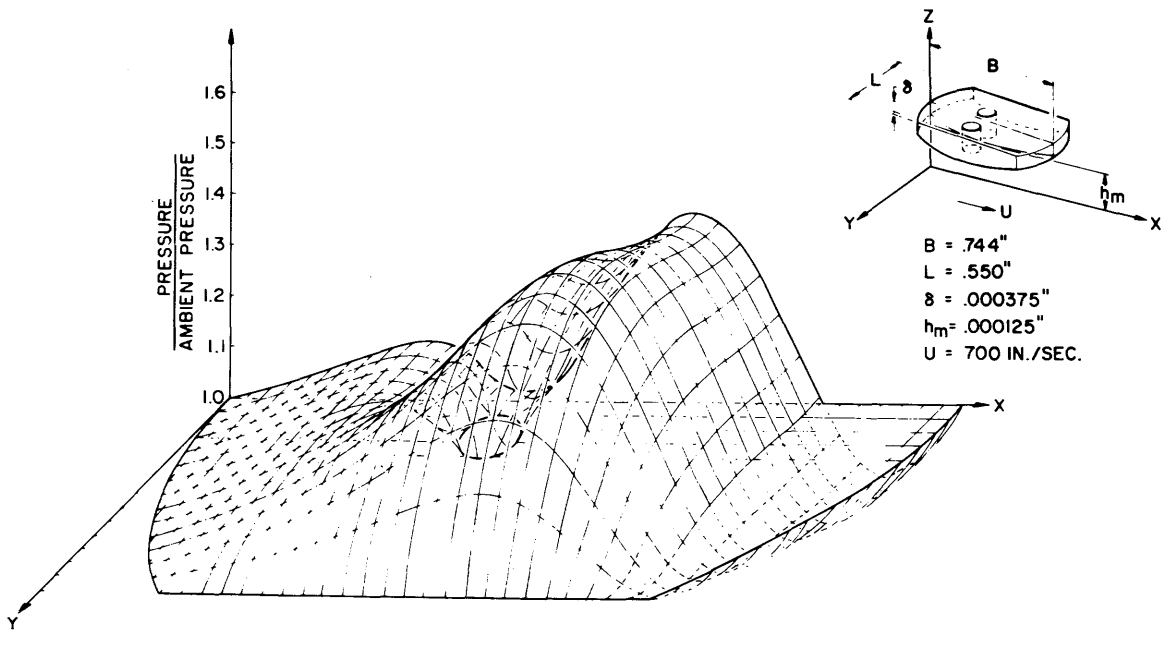


Figure 13. Computed Pressure Distribution for the IBM 1311 Slider.

angular orientation of the 1000 BPI head could not be held within the specified accuracy using this mount unless the manufacturing tolerances were held very closely. Because the resultant cost would be prohibitive, a two-pivot suspension system was investigated.

Since the dynamic performance of a slider bearing is affected by the amount of torque required to overcome the friction in the mounting, it is desirable to keep the pivot torque as low as possible. Also, it is noted that the slider is more sensitive to variations in roll torque than to variations in pitch torque. Therefore, it was decided to eliminate the pivots on the pitch axes and instead use the head supporting spring as a flexure pivot on the pitch axes. This change made it possible to maintain the required angular accuracy at a reasonable cost. Figure 9 shows the spring and head assembly mounted on its supporting arm. This mount holds the head at the required angle within 21 minutes.

After the head and leaf spring assembly is mounted on the supporting arm, the read-write pole tips must be set relative to the locating surfaces on the supporting arm: The required dimensions must be set within ± 0.001 inch in the circumferential direction and ± 0.003 inch in the radial direction. A larger radial tolerance is permitted because the radial position of each head is adjusted after it is installed in the machine. Figure 14 illustrates the critical tolerances associated with the alignment of the head supporting structure.

Mechanical Tolerances of the Head/Disk System

While the nominal spacing of the 1000 BPI slider is 125 microinches at 700 inches per

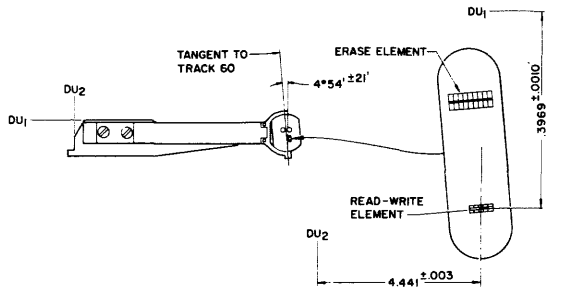


Figure 14. Critical Dimensions for Head Suspension System.

second, this exact film thickness cannot be maintained in the machine due to the tolerances which are allowed on the various components. Variations in the air bearing parameters, deviations of the disk surface from a plane surface, and tolerances in the suspension system all contribute to changes in the film thickness. Although the effects of these parameters are not entirely independent, a summation of the possible worst-case film thickness variations results in a film thickness of 125 ± 43 microinches at the inner track and 162 ± 43 microinches at the outer track.

To summarize, the air bearing and its suspension system were designed so that the head orientation and spacing could be held within the limits required to maintain a satisfactory signal amplitude and low noise level.

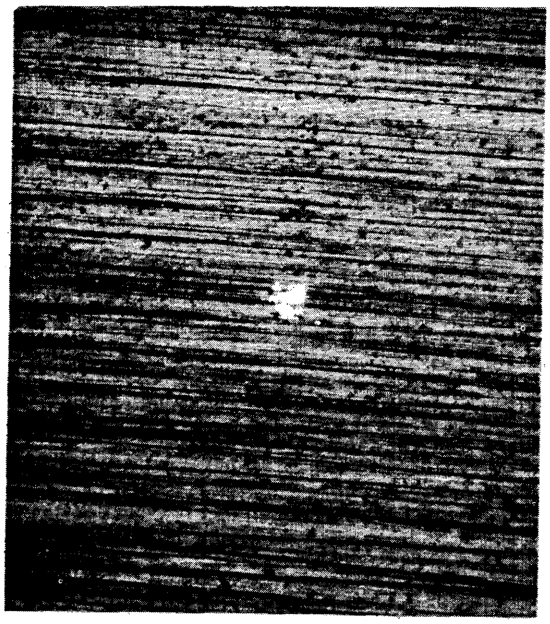
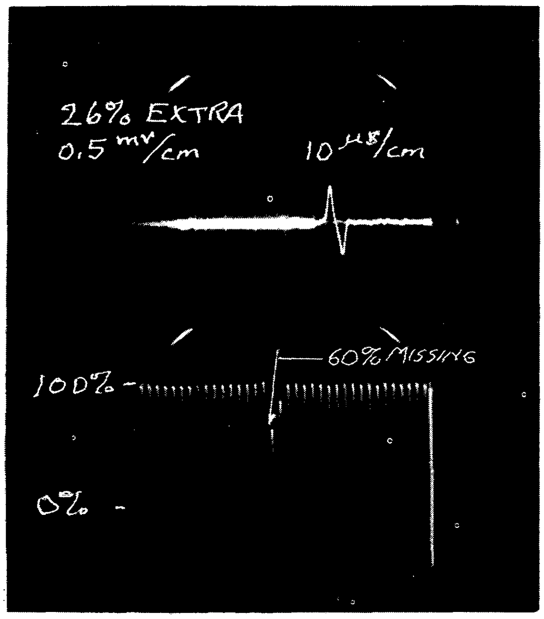
Surface Technology

To this point the disk has been mentioned only briefly. Its mention with respect to the read/write and the air bearing technologies is an indication of its dual function: (1) the provision and maintenance of the stable air film necessary for the slider; (2) the provision of a magnetic recording medium for information storage.

The interchangeability feature of the system produces new problems in both of these areas. The influence of disk coating thickness on signal amplitude and resolution demands close control on the thickness parameter. The dynamic runout of all disks must be held within certain values to insure that the heads on any file would be able to enter and operate in any disk pack. Since contamination becomes a factor of significance, the physical properties of the coating must be controlled.

The 1000 BPI recording aspect of the file imposes its share of problems in critical areas of disk technology. Defects which were too small to produce errors at lower bit densities show up for the first time. The lower air spacings (82 microinches minimum) demanded by 1000 BPI recording require a surface free of projections over 60 microinches high, to insure that there will be no projections capable of scratching the head.

To describe in detail all of the considerations involved in obtaining a disk compatible with

(a) ACTUAL PHYSICAL DEFECT IN DISK COATING (68x)

(b) MAGNETIC RESPONSE TO THE DEFECT

Figure 15. Example of Physical Defect and Resultant Magnetic Response.

the total system is a task which lies beyond the scope of this paper. Several of the more interesting problems, however, are described below.

The coating was one such problem, and continuity and durability were two crucial areas of this problem. With a coating film nominally 200 microinches thick (required to meet the signal amplitude and resolution specifications), a recording density of 1000 bits per inch, and a track width of 0.010 inch, a single bit occupies 2×10^{-9} cubic inches. Under these conditions a small coating defect would produce a single-bit error. An error-causing defect and its magnetic response is shown in Fig. 15. Investigations indicated that one of the major sources of rejected disks was the defect left in the substrate by the inadvertent removal of the intermetallic compounds which are precipitated during the heat treatments required for processing the aluminum substrate. Defect investigation resulted in the development of processes which produce error-free disks.

The maximum thickness is critical because of the influence of film thickness on bit resolution. The relationship between thickness and resolution was found to be linear within the area of our concern. Figure 16 is a resolution measurement (bit pattern: 01001111 . . .) showing the reduction of amplitude which does exist in a worst-case bit pattern. The difference between the amplitude of the isolated bit (peak 1) and that of the lower bit (peak 4) within the

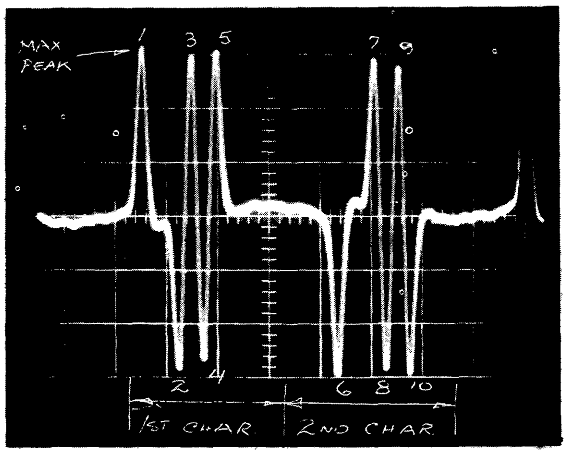


Figure 16. Resolution Measurement Showing Difference Between Amplitudes of Isolated Bit (Peak One) and Lowest Bit (Peak Four).

crowded bit portion (1111) is representative of the resolution. If the film thickness becomes excessive, the amplitude reduction will exceed the allowed 15%. A coating whose thickness vs. output and thickness vs. resolution characteristics were compatible was developed. The resolution properties of this coating are such that the maximum thickness is 240 microinches.

Because interchangeability magnifies the importance of coating durability, a series of tests were developed which would reliably characterize the disk coating.

One is the impact abrasion test, which is representative of the initial head loading occurring in the file. It is typical of the tests devised to sample production disks.

Another test which was devised was a modified Knoop Microhardness indentation test. However, because of the deviation from standard testing procedure the results of the test are termed "Knoop Coating Number" rather than "Knoop Hardness Number." This test included the study of such parameters as cure times and temperatures, and resulted in determination of the optimum amount of indentation resistance required for reliable machine performance.

V. CONTAMINATION

Interchangeability, high bit density, and the mechanical parameters associated with disk storage systems emphasize the contamination problem which exists in any dynamic data storage system. In particular, the presence of small particles creates a potentially destructive situation which could result in an excessively high error rate. This section describes the four approaches which successfully solved the contamination problem associated with the IBM 1311 Disk Storage Drive.

The first approach was to eliminate the sources of the contaminants from the disk pack environment. The second was to prevent contamination of the head-disk area. The third was to develop a contamination-resistant disk coating, as described previously. Finally, steps were taken to design the system so that it could withstand certain types of damage without malfunctioning.

The investigation began by contaminating head to disk air gaps with various particles. It was found that particles with diameters of 0.0015 inch to 0.0035 inch were the most damaging. Particles larger than this did not penetrate the space between the head and disk, and those smaller did no significant damage. Tests also showed that particles softer than annealed iron did not cause serious damage. These findings were extremely valuable when designing areas where wear effects provided the source of contamination. Plastic wear surfaces were used wherever possible so that the wear product of machine operation would be too soft to cause damage. This technique and others reduced the amount of internally generated particles.

In addition to eliminating particle sources, steps were taken to exclude contamination from the air gap between the head and the disk. It was determined that the ability of a particle to enter between the head and the disk was related to the leading edge radius on the slider. Sharper leading edge radii inhibit the particle penetration. This is shown schematically in Fig. 17.

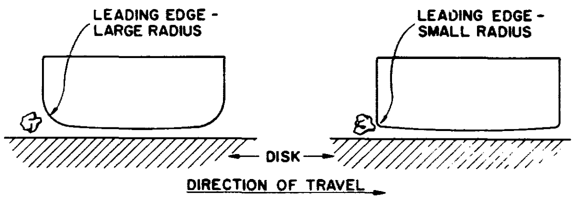


Figure 17. Schematic Diagram Showing Effectiveness of Small Leading Edge Radii in Excluding Particles from the Space Between the Head to Disk.

Air flow studies determined that the rotating pack could be used as an air pump. Originally the pack drew external air and passed it through the head area. Unfiltered air allowed into the file during pack changing was the air used as the bearing film. This condition was eliminated by using the rotating pack to pump filtered air directly over the surfaces of the disks. This effectively pressurized the pack with clean air, thus excluding the unfiltered air. Clean air is provided by venting the hub and spacer rings so that air can be pulled up from the bottom of the pack and passed out onto the disks. The basic air flow diagram is presented in Fig. 18. The ventilated pack design, by providing a laminar flow of clean air for use as an air bear-

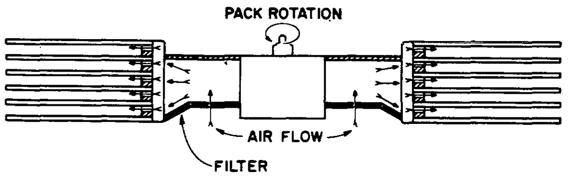


Figure 18. Schematic Drawing of the Disk Pack Cross-Section, Showing the Air Flow Which Distributes Clean Air to All the Disk Surfaces.

ing, prevented possible entry of damaging particles.

At this point, although the machine was operating highly satisfactorily, there still existed occasional random errors which were thought to be caused by electronic malfunctions. The bit configuraton (Fig. 19) of the error was rather peculiar and appeared to be caused by a high-frequency noise spike of unknown origin. This error occurred only at a certain spot on a disk. Microscopic inspection of the disk surface showed the presence of a very small particle in the area of the error. This particle was far too small to create a disk-defect error.

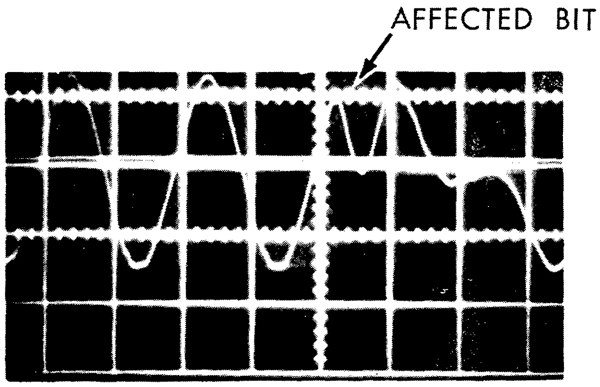


Figure 19. Oscilloscope Trace Showing the Effect of the Capacitive Discharge on the Readback Signal.

Our investigation proved that the cause of the error, immediately following head-selecting operations, was the capacitance discharge through the pole tip laminations to a particle on the disk. .This discharge induced an extraneous signal in the head. The solution to the problem was to ground the lamination.

It should be pointed out that each solution proposed in the preceding paragraphs was verified by actual machine operation. Thus all measures taken to eliminate particulate damage were subjected to extensive tests.

Two types of tests were used to verify the solutions. One was a pack-changing life test, in which machine performance was continuously monitored as packs were repeatedly changed. Another basic test was to operate the file in a purposely contaminated room. The contamination level in the room was maintained at a level far exceeding that of the expected worst case in the field.

The IBM 1311 Disk Storage Drive passed these tests with an excellent safety margin, thus proving that the contamination problem has been solved.

VI. SUMMARY

The requirement for interchangeability demands that many new parameters be considered in the design of the recording system. Such parameters as mechanical and electrical tolerances, and human factors were considered in the design of this recording system. Throughout the design the worst-case limits of each variable were determined and their effects were considered.

The total systems approach integrates the technological advances of many areas. These advances consist of:

—the development of magnetic elements where 1000 BPI (50 TPI) recording has been accomplished with a magnetic configuration which permits reliable reading and writing of information from any 1311 disk pack;

—the development of an air bearing to provide controlled spacing of a nominal 125 micro-inches between the head and disk;

—the provision of a thin, highly contamination-resistant coating approximately 200 micro-inches thick required for high resolution and optimum signal output;

—the development of electronics to deliver the fast symmetrical current waveforms for recording, and to reliably retrieve data over wide signal ranges and under worst-case noise conditions; and finally,

—the solution of the contamination problem associated with the interchanging of disk packs.

ACKNOWLEDGMENTS

The authors wish to acknowledge contributions by their associates in the IBM development laboratory, San Jose. Much of the information pertaining to the 1000 BPI air bearing was based on laboratory investigations by D. H. McClung. The theoretical air bearing design was developed by Dr. T. Tang. Information concerning the magnetic characteristics of the disks was furnished by D. W. Brede. A considerable amount of the work on coating durability was done by L. M. Wilson.

REFERENCE

1. BRUNNER, R. K., HARKER, J. M., HAUGHTON, K. E., and OSTERLUND, A. G., "Experimental Investigation of Pivoted Slider Bearings," Part III of "A Gas Film Lubrication Study," *IBM J. Res. & Dev.*, Vol. 3, No. 3, pp. 260-274; July 1959.

BIBLIOGRAPHY

1. GROSS, W. A., "Gas Film Lubrication," John Wiley & Son, Inc., New York, N. Y.; 1962.
2. HOAGLAND, A. S., and BACON, G. C., "High Density Digital Magnetic Recording Techniques," IRE Trans. on Electronic Computers, Vol. EC-9, No. 1, pp. 2-11; March 1960.
3. MIYATA, J. J., and HARTEL, R. R., "The Recording and Reproduction of Signals on Magnetic Medium Using Saturation-Type Recording," IRE Trans. on Electronic Computers, Vol. EC-8, No. 2, pp. 159-169; June 1959.
4. HOAGLAND, A. S., "Magnetic Recording Head Design," Proc. of Western Joint Computer Conf., pp. 26-31; February 1956.
5. SHEW, L. F., "High Density Magnetic Head Design for Noncontact Recording," IRE Trans. on Electronic Computers, Vol. EC-11, No. 6, pp. 764-773; December 1962.

ENGINEERING DESCRIPTION OF THE BURROUGHS DISK FILE

R. W. Jack, R. G. Groom, and R. A. Gleim
ElectroData M & E Division, Burroughs Corporation
460 Sierra Madre Villa
Pasadena, California

INTRODUCTION

For over eight years Burroughs Corporation has designed and manufactured random access bulk storage devices. The first two such devices produced were the Burroughs 205 and 220 Datafile units. They were magnetic tape storage devices in which a common read/write head was positioned to one of fifty strips of magnetic tape. After head positioning, finding the information in question was the same as searching for a record on a conventional magnetic tape unit.

For the past few years Burroughs has been searching for means to eliminate many of the disadvantages of complex electromechanical magnetic bulk storage devices. Studies have been made of many techniques used for storing large amounts of information capable of rapid access. The studies included the evaluation of other magnetic tape devices and techniques similar to those used in storage devices on the market today. It was soon apparent that if a significant step forward were to be made in the field of large volume random access magnetic storage devices that the following criteria would have to be satisfied:

1) Very short access time.
2) Very high reliability: Maintenance cost and low reliability have been among the main shortcomings of all bulk storage devices except magnetic drums.
3) Practical modularity of storage capacity.

Now let's take each criterion separately and outline some of the characteristics which a magnetic bulk storage file should have to satisfy each criterion.

Very Short Access Time: In most files today access time is a function of two major factors: head positioning time and disk or drum latency time (rotation time). Since head positioning is typically the larger of the two (approximately .03 to .5 second), and if it could be eliminated by substituting a fixed head for each track, a very significant improvement in access time could be achieved. With no positioner, access time then becomes a function of the media rotation. Eliminating head positioning also provides other benefits which are described later in this paper.

Very High Reliability: One of the most serious constraints on adequate reliability of magnetic storage devices of the past and present has been the reliability of the head positioning device. The fact that a head is movable, means that it may be moved not only to the wrong location but also slightly mispositioned. In addition, there are moving parts which are subject to wear and require frequent maintenance attention. If the requirement for moving the head is removed, then one of the major points of unreliability in the bulk storage device is eliminated.

Practical Modularity of Storage Capacity: This must be attainable in the field in a minimum of time and effort and without the necessity of entering the existing storage device and possibly disturbing stored information. In other words, a pluggable module is desired.

The head per track disk file, produced by Burroughs Corporation, meets the above requirements, and the purpose of this paper is to describe some of the technical aspects of this device.

GENERAL FILE CHARACTERISTICS

The Burroughs Disk File is made up of two basic units: a common electronics unit plus one or more storage modules. One common electronics unit may have as many as five storage modules. Each storage module is capable of storing 9.6 million, six bit, alphanumeric characters of information. The storage module is available in three different segment (record) organizations: 96, 240, and 480 character segment lengths. Segment length is a factory option. Total storage capacity of the various combinations of units on the B200 and B5000 Systems are shown in Figure 1.

	MAX ALPHA CHARACTERS PER	MAX SEGMENTS PER	MAX DISKS PER	MAX STORAGE MODULES PER	MAX BULK FILE ELEC. UNITS PER	MAX BULK FILE CONTROL UNITS PER
SEGMENT	96 240 480					
DISK	2,400,000	5,000 480 CHAR. 10,000 240 CHAR. 25,000 96 CHAR.				
MODULE	9,600,000	20,000 480 CHAR. 40,000 240 CHAR. 100,000 96 CHAR.	4			
ELEC. UNIT	48,000,000	100,000 480 CHAR. 200,000 240 CHAR. 500,000 96 CHAR.	20	5		
CONTROL UNIT	480,000,000	1,000,000 480 CHAR. 2,000,000 240 CHAR. 5,000,000 96 CHAR.	200	50	10	
B-200	480,000,000	1,000,000 480 CHAR. 2,000,000 240 CHAR. 5,000,000 96 CHAR.	200	50	10	1
B-5000	960,000,000	2,000,000 480 CHAR. 4,000,000 240 CHAR. 10,000,000 96 CHAR.	400	100	20	2

Figure 1. Storage Capacity for Various Unit Combinations.

The file is a head/track file with fixed, air-bearing heads. Head positioning is not required. There are four disks per storage module with 150 information tracks per disk face. The disks rotate at 1500 RPM, thereby making the average access time to any segment of information 20 milliseconds and the maximum access time

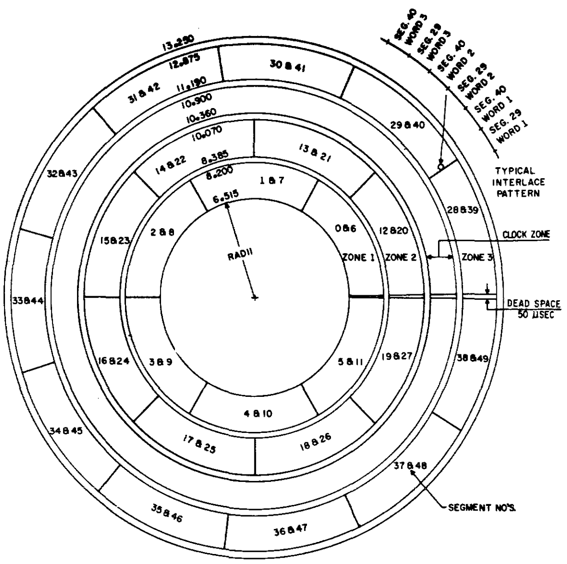


Figure 2. Disk Layout for 480 Character Segment Size.

40 milliseconds. Figure 2 shows the track organization and layout for a 480 character segment file. Each disk face is divided into three zones, each with a fixed clock frequency chosen so that the information packing density of the innermost track of each zone is approximately 1000 bits/inch. Non-return-to-zero recording is used. The clock area is located as shown between zones 2 and 3 and consists of the necessary information and address clock tracks. Each disk face has its own independent set of clocks.

Information is recorded serially by bit in 56 bit words. (See Figure 3.) The 56 bit word contains 8 (six bit) alphanumeric characters plus 1 (six bit) check character and two spacer bits. The check character will be explained later. As previously mentioned, the file is available in any one of three possible segment lengths. Figure 4 tabulates the different segment sizes and gives the number of characters

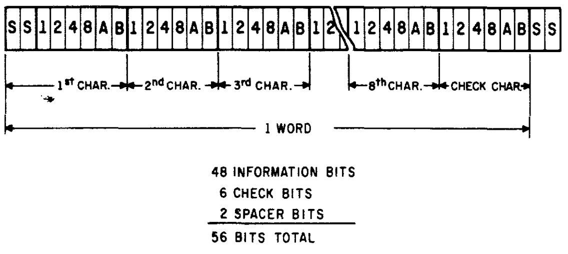


Figure 3. Word Format.

CHARACTERS PER SEGMENT	WORDS PER SEGMENT	SEGMENTS PER STORAGE MODULE	WORDS PER STORAGE MODULE	CHARACTERS PER STORAGE MODULE
96	12	100,000	1,200,000	9,600,000
240	30	40,000	1,200,000	9,600,000
480	60	20,000	1,200,000	9,600,000

Figure 4. Tabulation of Segment Sizes.

and words per segment for each configuration. With the high packing density used, the bit transfer rate from and to the disk is as high as 1.8 million bits/second. This provides a character transfer rate of as much as 264 kilocharacters/second. In order to adapt this transfer rate to the B200 and B5000 Data Processors it is necessary to interlace words from two different segments thereby reducing the maximum effective character transfer rate to the Data Processor to approximately 132 kilocharacters/second. Figure 5 gives the exact transfer rates and packing densities for each frequency zone.

DISK INFORMATION LAYOUT

In determining the best format for a disk file, the systems on which it will be used must be taken into consideration. The statement that a particular storage device has the capability of storing so many million bits means very little unless the device has an information format which is useful with a given Data Processor. Here are a few restrictions which must be taken into consideration in designing such a disk format.

1) Maximum allowable packing density must not be exceeded.
2) Checking bits or characters must be included (this is storage space lost for information).
3) Addressing scheme must be simple for the programmer to use and compatible with the computer system.
4) Transfer rate to the file must be compatible with the rest of the system.

In determining the best layout for the Burroughs Head/Track file, the following restrictions were imposed.

1) Must be compatible with both the B200 and B5000 Systems.
2) Packing density should be approximately 1000 bits/inch.
3) Three segment length options were required (96, 240, 480 characters).
4) Total segments/disk face had to be a decimal number and an integral multiple of 10.
5) Number of tracks/zone had to be decimal and an integral multiple of 10.
6) Since the B5000 uses an 8 character word, the number of characters per segment had to be an integral number of 8 character words.
7) The checking scheme had to be reliable and compatible with Items 1 through 6.

BASIC STORAGE UNIT

Introduction

Thirteen track, air-bearing magnetic head assemblies comprise the basic read/write trans-

ZONE	SEGMENTS/TRACK/ZONE			BITS / TRACK	INFO CHAR / TRACK	WORDS / TRACK	BIT TRANSFER · RATE DFEU – DFCU (MC)	CHAR. TRANSFER RATE DFEU–DFCU (KC)		CHAR. TRANSFER RATE DFCU–DP (KC)	PACKING DENSITY BITS/INCH	
	480 CHAR. SEG.	240 CHAR. SEG.	96 CHAR. SEG.					INCL. CHK. CHARACTER	INFORMATION CHAR. ONLY		MIN	MAX
1	12	24	60	40320	5760	720	1.008	162	144	72	785	988
2	16	32	80	53760	7680	960	1.344	216	192	96	849	1020
3	22	44	110	73920	10560	1320	1.848	297	264	132	912	1050
TOTAL	50	100	250	—	—	—	—	—	—	—	—	—
CLOCK	—	—	—	73920	—	—	—	—	—	—	1025	1079

Figure 5. Transfer Rates and Packing Densities.

ducers. They are floated at an approximate distance of 125 microinches from the disk surface on a laminar layer of air developed by the rotating disk. The disks are 26½ inches in diameter and ⅛ inch thick; they are brass donuts plated with an extremely thin magnetic film.

Storage Disk

Base Material: During the development of the disk, several types of materials were analyzed as possible candidates for the disk substrate material. After investigating many parameters such as availability, stability, size limitations, machinability, and compatibility with the plating process, brass was chosen.

After choosing the substrate material, many months were spent in developing an economical process for providing the desired flatness, run out, and surface roughness. The process involves special flattening, lapping and polishing techniques. After processing, the disk is inspected for the following requirements:

1) Disk run out or wobble must be gradual and must not exceed .005 inch total indicator reading when rotated on a horizontally mounted spindle.
2) Surface flatness must be less than 20 microinches when measured in any 1″ diameter area over the entire disk surface.
3) The average surface roughness must be less than 4 microinches when measured at any point on either disk surface.

Magnetic Characteristics: One criterion for high density recording is the need for a very thin and uniform magnetic coating. To solve this problem, Burroughs has developed several types of magnetic film plating processes which involve both electroplating and electroless plating techniques. Both processes are used in Burroughs current products. The electroplating process was chosen for the disk file. This process plates a magnetic film less than 30 microinches thick with a coercivity of about 500 oersteds and a remanence of approximately 6,000 gauss. This plating thickness is less than 1/16 the thickness of typical oxide coated magnetic tape. After plating, each disk is dynamically balanced and tested for flaws (before assembly) on automatic flaw testing equipment; all inspected disks are free from flaws in active track areas.

Magnetic Head

Figure 6 presents an outline of the head and gives a better indication of its actual shape. The wedge, which gives the head its air-bearing capability, is exaggerated in the drawing and is at an angle less than a degree with respect to the base of the head.

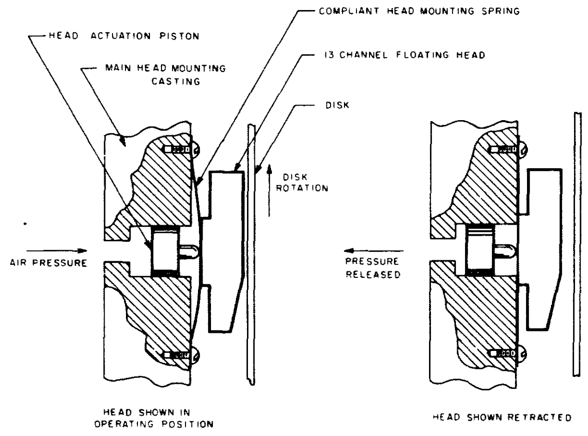


Figure 6. Pictorial Representation of the Head, Its Mounting and Activation Principle.

Extensive measures have been taken to hold head production costs to a minimum. Following assembly, each head is dynamically tested on an automatic performance tester to insure that its read/write characteristics are correct and that it floats at a predetermined gap for a given fixed pressure.

A few words should be said about the air-bearing technique which is the heart of the disk file. It is by using this technique that the close head to medium tolerances can be achieved. The head literally floats on a cushion of air and thus becomes self adjusting in its distance from the disk.

Studies for economical multi-channel heads were started at the Burroughs Research Laboratory seven years ago. These studies included the problem of the flying head shape. One method for producing the angle necessary for flying a head is to lap a spherical shape on the head flying surface. Although this method is good, it does not lend itself to a multi-channel head design since the same constant flying gap must be maintained at each track. These studies had as its objective the determination of the best and most economical materials and a head shape that would be reproducible, economical, and also give the most reliable flying characteristics.

Many months of theoretical studies were made involving the use of both analog and digital computers to determine what effect various head parameters had on flying characteristics and what head shape gave the most stable performance. This work was backed up by many additional man months of empirical testing of actual models. All tests were conducted using distance measuring probes mounted in model heads which were very accurately calibrated so that flying gaps could be monitored during all tests. These techniques have been developed to such an extent that head to disk distance can be measured to a tolerance of $\pm$ 5 microinches.

Head Actuation Principle

As mentioned earlier, to achieve high packing densities it is necessary to float the head in very close proximity to the disk. Since the head to disk gap is a function of the disk surface speed, it is necessary to increase the pressure applied to the head as it moves out on the disk radius. This is difficult to implement on a movable head type disk file since the head is required to operate over a range of varying surface speeds. However, this restriction is overcome with the use of fixed heads.

Figure 6 is a pictorial representation which shows the head mounting arrangement and actuation principle. As can be seen, the head is mounted to a flexible type flat spring which is rigidly mounted to the main head mount casting. The spring provides the facility for the head to align itself during landing and also provides a force for the head to return when the actuator is released. Actuation of the head is accomplished by pushing the head towards the disk by means of an air actuated plunger. Each of the eight disk faces (of the four disks in a storage module) have 7 interlaced pairs of operating head assemblies, making a total of 14 single head assemblies per disk face. Through the facility of a manifold distribution system, one interlaced pair of heads from each disk face (operating in the same pressure zone) is actuated by air which is provided by one of 7 pressure regulators housed in each storage module. Therefore, the air pressure from one regulator actuates 8 interlaced head combinations (16 head assemblies) at one time.

Floating Head Touch Protection System

Contact between the head and recording medium is a problem which exists with all file systems which use floating heads. This type of failure can be catastrophic and permanent damage to the disk and/or head can result. Burroughs has made every effort to make its disk file as free from failure as possible.

To achieve a "touch-free" disk file, all components within the head floating system are designed to be fail safe. The disk speed is monitored with a tachometer which does not permit the heads to actuate unless the disk is rotating at a safe speed. This same circuit will automatically retract the heads if, for any reason, the disk speed falls below a certain speed. Since the heads are air actuated and spring returned, the heads will automatically retract in case of failure in air pressure; power failures of any type automatically release the air pressure, causing the heads to retract.

The final protective device is a "Touch Circuit." This circuit applies a signal between the head body and recording medium and can be adjusted to detect any desired gap; the gap is set for a minimum safe floating gap. If the head comes closer to the disk than this set gap, or the gap is bridged by a foreign particle, this is sensed and the heads are automatically retracted. Provisions have been made for determining which head caused the touch.

Head Interconnection and Selection Method

Packaging of the disk file presented many problems which had to be overcome. For with 96 read/write and 16 clock head assemblies which comprise a 4 disk storage unit, more than 4,000 head leads were required. For obvious reasons conventional head wiring was impractical, and many schemes were studied with the aim of minimizing the number of head leads required. A scheme was ultimately adopted which involved the grouping of all the read/write heads into head banks and then into head sets. One such head set is shown in Figure 7; twelve duplicate head sets comprise a storage unit (Figure 8).

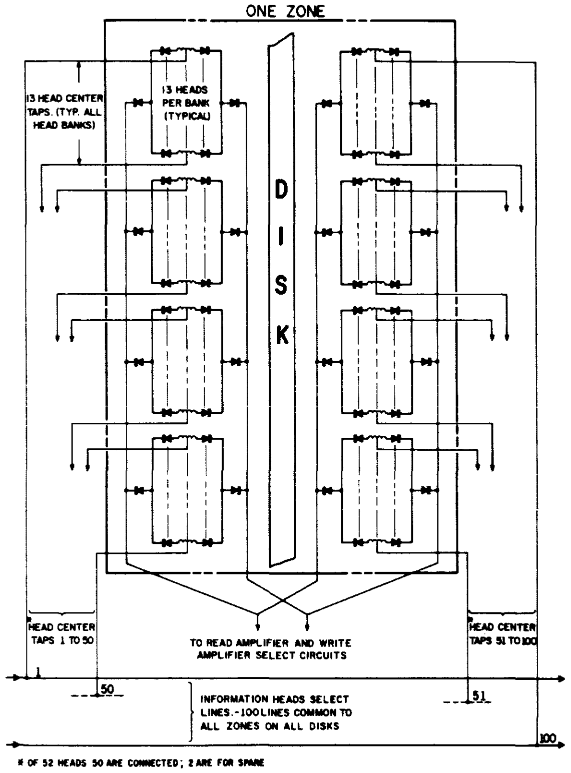

ONE ZONE
13 HEAD CENTER TAPS. (TYP. ALL HEAD BANKS)
13 HEADS PER BANK (TYPICAL)
D I S K
HEAD CENTER TAPS 1 TO 50
TO READ AMPLIFIER AND WRITE AMPLIFIER SELECT CIRCUITS
HEAD CENTER TAPS 51 TO 100
1
50
51
100
INFORMATION HEADS SELECT LINES.-100 LINES COMMON TO ALL ZONES ON ALL DISKS
* OF 52 HEADS 50 ARE CONNECTED; 2 ARE FOR SPARE

Figure 7. Head Set.

As can be seen in Figure 7, one head bank (equivalent to one head assembly) is comprised of thirteen heads which are connected in parallel and are isolated from each other by head switching diodes. For each head set, eight head banks are connected in parallel and are also isolated from each other by head switching diodes. The output from the entire head set goes through the appropriate select circuit to the read or write amplifier, depending upon the mode of operation. Having connected the desired head set to the proper read or write amplifier, all that is required to read or write with any one of the heads in the set is to select the center tap for the desired head.

Normally the center tap lines are held at a bias level with respect to the head set output lines. This bias level is such that all of the head switching diodes are reverse biased and in the non-conducting state, therefore, neither reading nor writing can take place with any head. To select one of the heads, the bias level on the head select line is changed in the direction to forward bias the two diodes associated with the desired head and the two diodes common to the head bank desired.

In the forward biased condition, the head switching diodes may pass current and permit reading or writing with the designated head. All of the other head switching diodes are still reverse biased, therefore, only one head in the head set is selected. Since the 100 head select lines are common to all head sets, one head in each head set will be selected along with the desired head. However, only one read amplifier and write amplifier select circuit is activated, therefore, the output from the undesired head sets will be ignored.

The seemingly extra diodes isolating the head banks reduce the effective capacity of the reverse biased diodes by connecting the diode capacities of the head diodes and head bank diodes in series. The diode capacity, even when reverse biased, allows a small amount of signal from each head to appear at the output. The extra diodes reduce this stray signal to a tolerable level.

Although the scheme described above greatly reduces the number of head leads brought out, a large number of interconnections are still required. To minimize the remaining interconnecting problems, printed circuit boards are used where possible so that production soldering techniques can be employed.

OVERALL LOGICAL DESIGN

Address and information from the data processing system is transmitted in parallel, six bit characters to the Disk File Control Unit (DFCU) in 8 character words. The address is decoded and stored in the Control Unit.

An address consists of seven binary coded decimal digits. Each segment within the five storage modules which may be connected to the

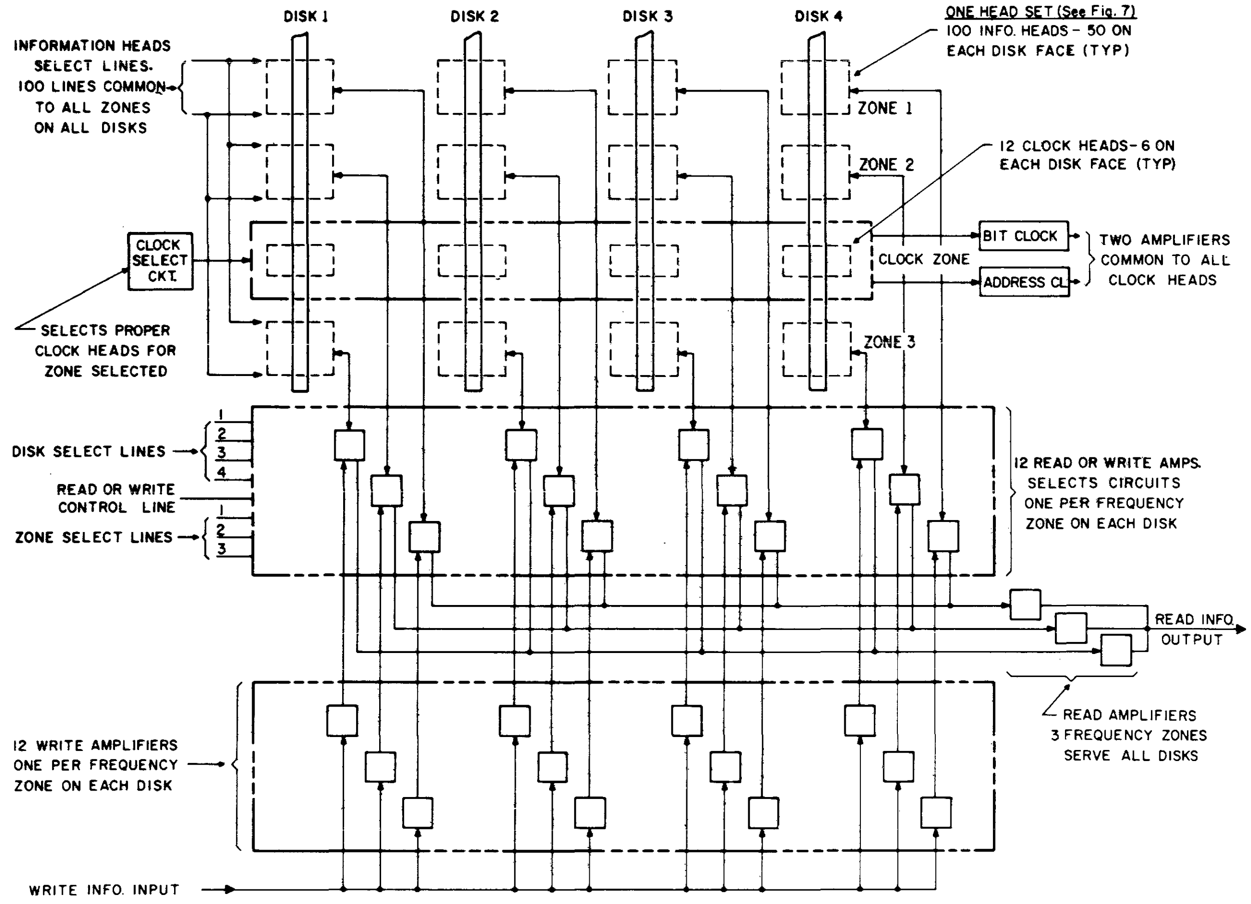

DISK 1
DISK 2
DISK 3
DISK 4
ONE HEAD SET (See Fig. 7)
100 INFO. HEADS - 50 ON EACH DISK FACE (TYP)
INFORMATION HEADS SELECT LINES. 100 LINES COMMON TO ALL ZONES ON ALL DISKS
ZONE 1
ZONE 2
12 CLOCK HEADS-6 ON EACH DISK FACE (TYP)
CLOCK SELECT CKT.
BIT CLOCK
ADDRESS CL
CLOCK ZONE
TWO AMPLIFIERS COMMON TO ALL CLOCK HEADS
SELECTS PROPER CLOCK HEADS FOR ZONE SELECTED
ZONE 3
DISK SELECT LINES
READ OR WRITE CONTROL LINE
ZONE SELECT LINES
12 READ OR WRITE AMPS. SELECTS CIRCUITS ONE PER FREQUENCY ZONE ON EACH DISK
READ INFO. OUTPUT
READ AMPLIFIERS 3 FREQUENCY ZONES SERVE ALL DISKS
12 WRITE AMPLIFIERS ONE PER FREQUENCY ZONE ON EACH DISK
WRITE INFO. INPUT

Figure 8. Block Diagram Showing Association of Read Amplifiers, Write Amplifiers and Select Circuits for Each Storage Unit.

Disk File Electronics Unit (DFEU) is identified by a unique decimal number. Addresses are sequential and no gaps exist in the address sequence. Address content is interpreted differently depending on the segment length option of the Disk File Storage Unit (DFSU) being accessed (see Figure 9).

The most significant digit of the address selects one of ten possible DFEU's. Then it is necessary to select a particular track by selecting one of a maximum of 6,000 data heads in the DFSU's associated with the selected DFEU. Track select levels are transmitted to the DFEU via 15 control lines. The desired head is selected by identifying the following:

1) Disk (1 of 20).
2) Disk Face (1 of 2).
3) Zone of the Disk (1 of 3).
4) Track within the Zone (1 of 50).

Within the DFEU the track select levels are further decoded thereby choosing one DFSU out of 5; one read or write amplifier out of 12; one head out of 1,200; and one set of word, address, and bit clocks out of 24.

Once the proper word clock and address clock heads are selected, it is then possible to complete the addressing. Segment addresses are read from the DFSU and transmitted to the DFEU. These addresses are read serially by bit and shifted into a six bit buffer and shift register located in the DFEU. This address is then compared to the address previously stored in the DFCU. When the sought after address and the read address agree, a read or write cycle is initiated.

A write cycle begins with transmission of the first 6 bit character from the DFCU to the DFEU where it is held in the same buffer as

480 CHARACTER SEGMENT FILE

7	6	5	4 3	2 1
DFEU NUMBER 0 - 9	NOT USED	DISK SET 0 - 9	TRACK NO. 0 - 99 (0-49-DISK 1) (50-99-DISK 2)	SEGMENT NO. 0 - 99 0-49-LEFT FACE 50-99 RIGHT FACE

5000 SEG/DISK 00000-99999 SEG/MAX DFEU

240 CHARACTER SEGMENT FILE

7	6 5	4 3	2 1
DFEU NUMBER 0 - 9	DISK NO. 0 - 19	TRACK NO. 0 - 99 (0-49-L. FACE) (50-99-R. FACE)	SEGMENT NO. 0 - 99

10,000 SEG/DISK 000000-199999 SEG/MAX DFEU

96 CHARACTER SEGMENT FILE

7	6	5 4	3 2 1
DFEU NUMBER 0 - 9	DISK SET NUMBER 0 - 4 4-DISKS PER SET	TRACK NO. 0 - 99 (0-49-DISK 1&2 (50-99-DISK 3&4)	SEGMENT NUMBER (0-249 DISK 1,L.FACE (250-499 DISK 1,R.FACE (500-749 DISK 2,L.FACE (750-999 DISK 2,R.FACE

25,000 SEG/DISK 000000-499999 SEG/MAX DFEU

Figure 9. Address Format.

mentioned above. The write control line is turned on and the stored character is then shifted out of the buffer and recorded serially by bit by the previously selected information read/write head.

This process is repeated for nine characters (eight information and one check character) at which time a word pulse is read from the word clock. This pulse turns off the interlace control circuit and writing ceases. The next word pulse turns the interlace control circuit back on and another nine character word is written. This write-a-word, skip-a-word process continues until the end of the record is reached at which time the action circuitry is shut off and the operation is complete. If more than one record is to be written, the action circuit is held on and the number of records written is tallied in the DFCU. When the correct tally is reached, the action circuit is turned off as before.

The read cycle begins when information is read from the selected information head and shifted into the buffer register in the DFEU. When the register is full, a character pulse is emitted to the DFCU as before and the character stored in the buffer register is transmitted, six bits in parallel to the DFCU, and thence to the data processing system. This process continues until nine characters are read and a word pulse turns off the interlace control circuit. The next word pulse turns the interlace control circuit back on and another nine character word is read. This read-a-word, skip-a-word process continues to the end of the record and is completed as in the write cycle. Multiple records are also handled as in the write cycle.

A checking character (ninth character in each word) is written at the end of each word and is initiated by the DFCU. The checking characters are read back as any other character and are verified by the DFCU.

In summary, information and addresses are conditioned and checked by the DFCU; recognition of the address is made by the DFCU; selection of the proper heads, control of information flow, interlacing of information, and buffering of information is performed by the DFEU; and reading and writing of information is performed by the DFSU.

DISK FILE MODULARITY AND EXPANDING CABINETRY

In the early development phases of the Burroughs Disk File, much study was given to the many types of files. The study included both vertical and horizontal mounted spindles with and without integral drive motors, and also of files with up to 24 disks. It was shown from studies made that a modular capability of approximately 10 million characters would meet most applications. The 4 disk file system, with its 9.6 million character capability, came closest to meeting these requirements.

With the decision to manufacture the smaller (4 disk) file concept, it was also decided to provide an electronic unit which could control a multiple of modular disk units. The electronic unit of the modularity concept being discussed can control up to 5 disk storage modules (Figure 10). In the field, a disk storage module can be added to existing equipment in a time period of less than 4 hours. The installation involves the addition of cabinetry, which provides a type of garage for the portable disk file, and incorporation of the necessary electronic cables and air lines. Removal of a disk file for repair or replacement takes less than 15 minutes. Any file unit can be removed from the system without disrupting the operational ability of the remaining files in the system.

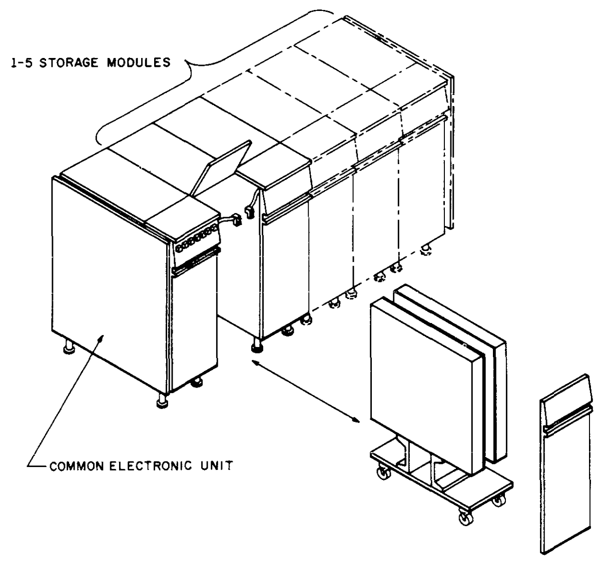


Figure 10. Modular Design Disk File and Expandable Cabinetry.

RELIABILITY

As previously mentioned, perhaps the most serious limitation of past and present magnetic disk storage devices has been the reliability of the head positioning mechanism. Elaborate schemes have been devised to verify head positioning. These are costly and also involve lost time. With any head positioning mechanism there is a finite inaccuracy of the positioner and this inaccuracy is a detriment to unit performance in many areas. Among these is a practical limitation on maximum packing density because of the effect of positioning inaccuracy on read/write system performance. A slight mispositioning during a write command may cause the leaving of a residual track which on a subsequent read becomes a source of noise. The head per track file eliminates these major sources of poor performance. With the fixed head it is possible to begin to utilize to a fuller extent the maximum packing density capabilities of the head and the medium. In the Burroughs head per track file, all head switching is done with solid state electronic circuits.

Maintenance must be considered along with reliability. To achieve a certain reliability, a minimum amount of maintenance must be performed. Tests and analyses of this file have shown that it may be looked upon as a magnetic drum from the point of view of maintenance and reliability. A well designed magnetic drum becomes a very minor part of the maintenance of an overall data processing system.

In any magnetic recording system, the possibility of permanent and temporary read/write errors must be considered. In this file, permanent errors have been eliminated by the requirement that all active tracks on a disk be flaw free when manufactured. In other words, all file addresses are free from permanent read/write errors. However, temporary errors may occur. These may be due to a variety of factors including noise and dirt. Dirt, a primary source of error, has been minimized by operating the disks in a sealed off area. Since temporary errors may occur, an error check code is included which provides an error check character as part of each file word. This check character is so arranged on the disk geometrically, with respect to the information bits it is checking, that the probability of an error going undetected is extremely remote. For an error to be detected which may have occurred during the write process, it is of course necessary to re-read the record just written. To accommodate this, there is a special read check command which allows the segment just written to be re-read and checked by the control unit without tying up computer operations.

Much concern is usually raised concerning the possibility of catastrophic failures and their effect on the file. Perhaps the most catastrophic failure would be the permanent damage of a storage disk. In the Burroughs file, this is practically eliminated by the touch protection system previously described.

Another important feature increasing the reliability of this unit is the fact that each disk face has its own independent set of clocks. In the remote possibility of failure of one or more clocks in a 4 disk storage module, it would be possible to rewrite these clocks from other identical clock tracks remaining in the storage module.

THE FUTURE

We at Burroughs believe that the disk file, described in this paper, is the first of a new generation of random access devices. The file has been tailored in its logical organization to the Burroughs B200 and B5000 Systems. In doing this, it has been necessary to reduce the effective information transfer rate from the file to the system below the intrinsic capabilities of the file. However, it should be noted that this file (with existing circuitry and a slightly different logical organization) would be able, for instance, to communicate with a Data Processor at 1.8 megacycle *word* rate. In other words, this file has future potential which for the first time sees peripheral equipment giving the Data Processor a run for the money.

A MULTIPLE-ACCESS DISC FILE

Irving L. Wieselman and Raymond Stuart-Williams
Data Products Corporation, Culver City, California
and
Donald K. Sampson
Data Products Corporation, St. Paul, Minnesota

INTRODUCTION

During the past two years disc files have been accepted as a reliable and economic form of mass random-access memory. It is now possible to design disc file systems which are functionally and logically optimized to provide efficient operation for a wide range of applications. This is important since at any given computer installation, more than one application is so often required.

The user has a choice of several types of disc files. Each type has properties which are advantageous in some situations and disadvantageous in others. The way in which a disc file is used must be conditioned by its characteristics. At the same time, the way it is designed must be conditioned by the way it is to be used.

The system which is examined here is the Data Products Corporation Model dp/f-5035 DISCfILE* storage system. The design is based on independent positioners for each disc which permit data transfers on two positioners simultaneous with motion on two other positioners. The design philosophy is discussed and the implications of the systems design to the user is evaluated.

SYSTEM DESCRIPTION

General

The dp/f-5035 DISCfILE storage system is a mass random-access memory which combines simultaneous dual-access capabilities with a substantially reduced cost-per-bit. The dual-channel-access permits greatly reduced computation times, particularly in such applications as sorting, real-time random-access and in multi-computer systems. Its capacity in excess of 800 million bits, maximum transfer rate of 1,400,000 bits per second, and effective random-access times between 50 and 100 milliseconds make it particularly suitable for medium-to-large computing systems. In addition, its dual-access features allow simultaneous access by dual computers in a computer hierarchy.

When DISCfILE systems are attached on-line to computers, the total mass memory capacity that is available depends on the design of the computer-DISCfILE interface. Some computer manufacturers have elected to allow as many as four DISCfILE systems to be attached to the computer on one channel. In this way the capacity of a channel could be more than 3 billion bits. In addition, since each DISCfILE system is independent, the number of positioners moving at once is also four times that of a single system. Also, the computer manufacturer could utilize DISCfILES on more than one channel and achieve on-line capacities of the order of 10 billion bits.

The dp/f-5035 system is one of four mass memory systems which were recently announced by the corporation. The others are the dp/f-

* Trademark of Data Products Corporation.

5024, dp/f-5025 and the dp/f-5034. The basic storage module of the 5024 and 5025 systems is 200 million bits, while the 5034 and 5035 systems use modules which can store over 400 million bits per module. The 5025 and 5035 are dual-access systems. All systems attain the same maximum data storage capacity by using several storage modules.

These systems are an outgrowth of the dp/f-5020 storage systems[1] where a maximum capacity of 150 million bits was provided. Over seventy dp/f-5020 systems are in operational use. The new systems make use of the same reliable electromechanical components which were field-proved in the dp/f-5020. Increased bit densities are achieved by new magnetic coaitngs and compatible read/write electronics. The dual-access capability is achieved by taking advantage of the inherent property of an individual positioner for each disc.

Physical Characteristics

Figure 1 is a photograph of a typical dp/f-5035 system. It contains two logic units attached at opposite ends to a file storage module. Each logic unit controls an independent data and access channel. The system may be expanded to maximum capacity by including an additional file storage module as part of the system.

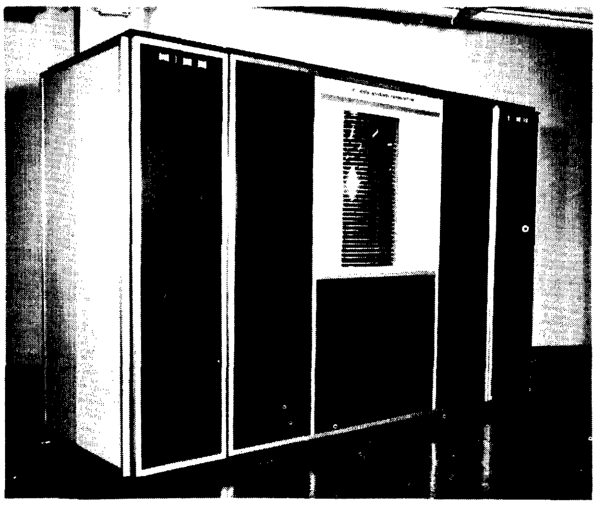


Figure 1. The dp/f-5035 DISCfILE System.

The data is stored as records in closed circular tracks upon the surface of 32 oxide-coated magnesium alloy discs. Each disc has 512 tracks; 256 on the top surface and 256 on the bottom. The discs are 31 inches in diameter and one-eighth of an inch thick. They are mounted on a stiff and precise spindle which is rotated by a 1200 rpm low-slip, three-phase induction motor.

Each disc is accessed by eight read/write heads mounted on a forked arm. This arm is moved and positioned rapidly and accurately by a 64 position magnetic linear positioner. For speed and economy, open-loop positioning is used. The positioners are mounted in two groups around the periphery of a shroud. The shroud is a basic cylindrical structure which supports and protects the rotating discs and provides the precision mounting points and the reacting mass for the linear positioners. The disc spindle bearings are enclosed in the circular end plates of the shroud.

Each file module contains a locked panel which is used to control write-lockout through a system of switches, one for each disc. When a switch on this panel is closed, it prevents writing or erasure from taking place on the associated disc. This allows selected areas of the file to be protected against any form of accidental erasure, similar to the use of a write-lockout ring on a magnetic tape transport.

The dp/f-5035 includes an error prevention, detection, and alarm system. Lights display the nature of the alarm and a signal is sent to the computer indicating that an alarm exists. All essential physical and electrical conditions are checked by this system. The system is similar to that employed in the dp/f-5020 but has been improved by the addition of a write-echo-checking scheme whereby circuits determine that the correct current flowed in the head at the right time.

In a data processing system, the logic units are generally attached to the central processor by a controller. The controller interface utilizes pulses for control signalling as well as data transferring. Additional relay-contact states are carried on other wires. The general interface philosophy is to assign a different pulse line for each operation. Some lines carry signals in both directions when there are no ambiguities. The basic operations transmitted to the 5035 are Select and Seek Record, Read, Write, Read Record Address and Clear Positioner. The 5035 transmits status information to the controller such as File On-Line, Address Received, Ready to transmit selected data, Alarm, and End of Record.

Reliability

The system is designed to operate in a normal data processing environment so that no special control of temperature, humidity or cleanliness is required. The unit needs no warm-up time. The scheduled maintenance required is about 2%, while unscheduled maintenance is rare.

The reliability of the system is determined by its ability to maintain error-free operation. Errors fall into two classes; recoverable errors and unrecoverable errors. When a recoverable error occurs, no data is lost. Most recoverable errors are such that the error may be eliminated by repeating the operation. This recoverable error rate is substantially less than one error for every billion bits of data transferred into or out of the file. In those situations where maintenance is required to eliminate an error condition, no data is lost but only time. Unrecoverable errors occur so infrequently that it is not theoretically possible to define an error rate in a reasonable time period. The system is designed and manufactured so that the recording surface is, and will remain, free from "bad spots."

Data Storage

The data storage surfaces of each disc are divided up into an inner and an outer zone. Each zone is accessed by two heads on the top surface and another two heads on the bottom surface of the disc. Consequently, at any given position, there are four inner zone and four outer zone tracks under the heads. These eight tracks are together known as a "position."

Switching between heads at a position is electronic so that the only delays for random accessing are due to rotational latency. However, all the data stored on eight tracks may be read or written without any accessing delays when accessed sequentially. In the 5035, the data stored at two positions can be transferred simultaneously.

Typical storage capacities are shown in Table I. Capacity is given in alphanumeric characters with an allowance for parity checking. The total number of characters on each

TABLE I

	Number of Records per Position				
	64	100	128	200	256
Alphanumeric Characters per					
Record	528	324	248	144	108
Position	33,792	32,400	31,744	28,800	27,648
Thousands of Records per					
DISCfILE	131	205	262	410	524
System	262	410	524	820	1,048
Millions of alphanumeric characters per					
DISCfILE	69	66	65	59	56
System	138	132	130	118	112

track is fixed. Since each record needs part of the track for addressing and format control, the available space for data is reduced as the number of records on a track is increased. The values listed take account of this effect since they exclude those areas that are unavailable for data storage.

Multiple Access

Access to data is achieved by moving a selected positioner to a selected position. When a positioner is moving to the desired position, it is "seeking." After it has reached its position and reading or writing is taking place, it is "accessing" or transferring data. The dp/f-5035 system has "dual-access" capability, and multiple file module systems can "multiple-seek."

Each logic unit contains the circuits for initiating a seek at one positioner in each file module. It also contains a complete data and access channel. Each file module contains the necessary circuits for it to be seeking simultaneously with two of its positioners. In a complete system with two modules, it is possible to be reading, writing, or reading and writing at two places simultaneously, while two other positioners are seeking or are holding a previously established position. Alternatively, a two-module system can have four positioners seeking at once.

The restrictions imposed upon the system are that two logic units cannot address the same positioner. Further, a logic unit must disengage a positioner which it set. Apart from these, there are no restrictions upon the interconnection of the logic units and the positioners.

The combination of multiple access and multiple seek has a significant effect upon access time. Table II shows typical random-average access times including all delays and 26 milliseconds average latency. The one-file system includes the effect of the dual-access while the two-file system is influenced by both dual-access and multiple-seek.

The transfer rate for the outer zone is approximately 700,000 bits per second and for the inner zone it is 400,000 bits per second. The mean transfer rate using both access channels is nearly 200,000 characters per second. The file can be loaded or dumped at a mean rate of about 150,000 characters per second.

In both scientific and business applications it is a common practice to perform block transfers involving large amounts of data which is often described as a sequential rather than a random mode of operation. In the dp/f-5035, each position stores in excess of 200,000 bits of data. This is more than 4096 words of 50 bits each. Hence the dp/f-5035 can transfer 8192 words in about 400 milliseconds, or an average rate of 50 microseconds per word.

THE COMPONENTS

The Recording Discs

The discs are 31 inches in diameter and one-eighth of an inch thick and are made from a non-inflammable magnesium alloy. They are precision lapped and are then coated with a magnetic oxide mixture. This is bound with a very tough and hard thermosetting plastic. The oxide "paint" is baked at a relatively high

TABLE II

Millions of Characters of Storage Capacity Involved in the Computation	Typical Random-Average Access Time in Milliseconds	
	One-File System	Two-File System
8	70	48
16	75	50
32	80	52
64	95	55
128	—	65

temperature to achieve a sufficiently hard-wearing surface. It is then lapped again with a very fine abrasive to achieve a sufficiently smooth surface. The resulting oxide coat is approximately one mil thick.

Oxide coated discs are used in preference to metallic coatings, even though it is recognized that higher bit densities could easily be obtained by using the latter. This choice is dictated by reliability and manufacturing considerations. To keep losses down, metallic coatings must be thin; hence, they do not have the abrasion-resistant properties of the thermoplastic-oxide coating, so reliability may suffer. Although metallic coatings have been used for drums and small discs, it is difficult to achieve high production rates for such large surface areas as are required in large capacity disc files. It is theoretically possible to develop the technology for metallic coatings but the ability to produce them in large quantities to meet our customers' needs dictated the choice of oxide coating.

Any recording medium contains imperfections, but if the manufacturing process is well controlled it is possible to set an upper limit to the acceptable size of an imperfection. If the track width is made considerably larger than this, there will be no bad spots. This technique is used in the dp/f-5035 since the track width is 0.025 inch.

The Flying Head

The "flying head" pad in the file is a rather simple form of air-lubricated slider-bearing. The term "flying head," which has become commonly used in the industry, has created the impression that the performance of this device is analogous to that of an airplane wing. This, in fact, is not true because the layer of air between the pad and the disc is so small that there is generally not sufficient space for normal aerodynamic effects to take place. In practice, the flying head is operated well below the velocity and pressure at which turbulent flow is to be expected and, for this reason, there is close correlation between experimental and theoretical data.

The technique for measuring the performance of experimental pad configurations follows very closely the methods described in the reference.[2] The basic theory is described in two other references.[3, 4]

The shape of the pad must be sufficiently deep to accommodate the magnetic head or heads contained within it. It must be sufficiently rigid so that it will not distort under the relatively high forces which are applied to it. It must be as light in weight as possible in comparison with the pad surface area upon which it flies.

A simplified diagram of the flying head pad is shown in Figure 2. There are, in fact, two "flying" surfaces joined together by a light but rigid bridge. Each part of the pad contains an erase head and a read/write head. The erase gap is 0.040 inch, the read/write gap is 0.025 inch, and the gaps are separated by 0.065 inch.

It is necessary that the head be forced against the air layer moving with the rotating disc in order to produce the slider air-bearing effect. If a normal spring were employed as the force, then the force exerted would be a function of the relative distance between the head supporting arm and the disc. To overcome this difficulty, compressed air and a "piston" is used to provide the force. A light return spring retracts the heads when the air pressure is removed. This method also provides "fail safe" characteristics should the air supply fail. This use of air pressure allows the relative distance between the arm supporting the head and the disc surface to vary by ± 0.025 inch without affecting the clearance between the disc and the head pad.

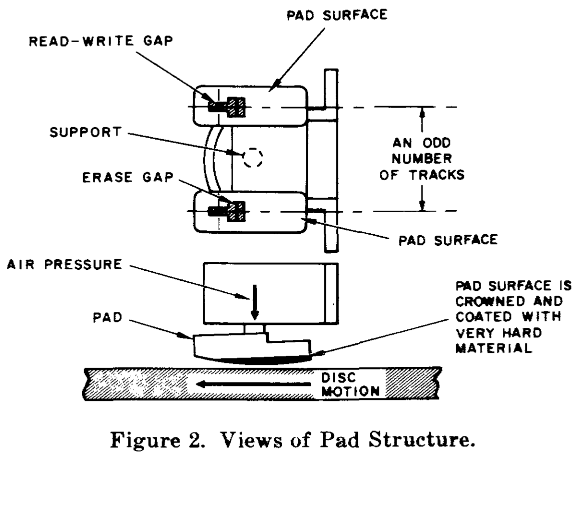


Figure 2. Views of Pad Structure.

In the present system the total indicated runout of the disc surface must be less than 0.020 inch, and the local curvature of the disc such that in a one inch radius there is no more than a 200 microinch change. Disc surface irregularities will look like a forcing function on the head. As a valley is encountered by the head, then the head must accelerate in order to follow the contours of the disc. The force of this acceleration is applied by the air pressure which is pushing on the head through the piston.

The acceleration which the head experiences when it "flies" over a hill of 200 microinches in 0.5 inch is 15 g's. The acceleration which the head can cope with is dependent upon the load on the head supplied by the air pressure and the mass of the head. Figure 3 shows the typical operating region for the head utilized in the system. The ratio of the load on the pad to the weight of the pad is 40 and the head is capable of following accelerations of approximately 40 g's.

The Magnetic Linear Positioner

The linear positioner must move the access arm rapidly, precisely and with the minimum acceleration and deceleration. It is a modern adaptation of principles which have been well known at least since Faraday. A linear positioner performs three functions obtainable by separable elements of the design. First, it is a motor. Second, it is a quantized magnetic ruler. Third, it is a dashpot which controls velocity and provides a "soft" stop.

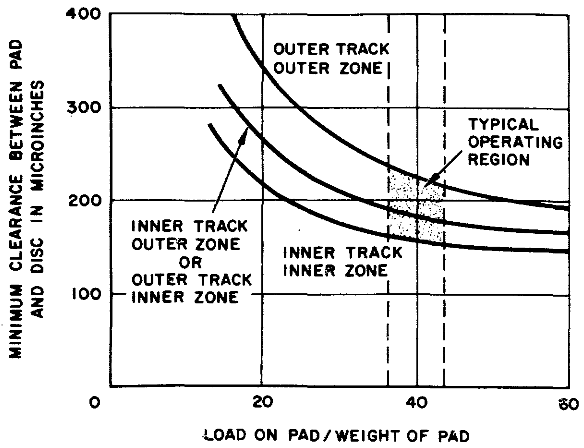


Figure 3. The Characteristics of the Head Pad.

The linear positioner is effectively a D.C. motor turned inside out so that the armature becomes the stator. The device is then cut and flattened out to form a linear motor. The principal components are two wound steel stator assemblies, a permanent magnet armature mounted on a carriage running on rails, and a forked arm carrying the head pads also attached to the carriage.

In order to position the armature, taps are selected on the stator by means of reed relays. A fixed magnetic field is set up which causes the armature to move to the selected position. The configuration of the elements is such that no matter where the armature is initially, it is always forced towards the selected position and then held there by a strong attractive field. Only one set of taps is selected for a given position since the armature is repelled by all the unselected taps and attracted only by the selected taps.

Each of the selectable 64 positions is, in effect, a magnetic "notch." These magnetic "notches" cannot get dirty or become worn as would a mechanical detent. Moreover, the precision of location of the notches cannot change because this is set permanently during the manufacturing process. The notches are only twenty thousandths of an inch wide at the widest point of the V. The restoring force which precisely locates the armature is equivalent to a spring of one hundred pounds per inch stiffness.

The mass of the arm and heads is so small that speed is not very difficult to obtain. The typical precision required demands reasonably careful manufacturing techniques but nothing which forces the state of the art. Control of the acceleration is by far the most difficult thing of all to achieve.

The forces which tend to restore the correct angle of attack of the flying head when in position for reading or writing are very large compared to its mass. When the head pad is moving to a new position, the restoring forces on the head are smaller. Another problem arises in stability since it is difficult to design a configuration where the head support is below the center of gravity. Thus, the forces which may cause the head to tilt and touch the disc must be minimized.

These forces may be minimized by controlling the accelerations during the start and stop of the positioning motion. The desired condition is obtained by a "soft" stop. This is difficult to achieve if a mechanical detent is used to obtain positioning precision since if the detenting mechanism is not well-adjusted or is worn, a final lurch will occur when the arm is stopped. The need for a "soft" stop is also the reason why hydraulic positioning mechanisms must be designed and manufactured most expertly. Such systems usually have a great reserve power and a slight misadjustment can introduce undesirable accelerations.

The final function which the linear positioner performs is to limit and control accelerations and oscillations as the positioner reaches the desired position. This function is performed by the combination of the permanent-magnet-armature and a copper shorting ladder. This represents a very simple and reliable form of viscous damping which produces the desired "soft" stop.

Whenever mechanical motion is utilized, there is an indeterminancy in the positioning obtained. This affects the design considerations of the head geometry and track spacing. When writing new data, the indeterminancy of position prevents the complete erasure of the old data. The unerased data appears as noise bands along the sides of the tracks and in time will introduce errors. This is avoided in the dp/f-5035 by using an erase head in front of the read/write head which is wide enough to clean a band on each side of the track. The band should be at least twice as wide as the maximum possible relative motion between the old track and the head. In the dp/f-5035 the maximum relative displacement is approximately ±0.001 inch and the guard band is 0.006 inch.

Next the effect of indeterminate positioning on track spacing must be considered. Under the worst possible combination of tolerances, there should be little danger of affecting the data in an adjacent track during a writing operation. The center to center track spacing in the dp/f-5035 is 0.0375 inch. Under the worst combination of tolerances, 0.001 inch of the adjacent track would be erased. The reading system is such that it is possible to read satisfactorily even if twenty times this amount is erased.

To insure that the positioner is reliably settled on the desired track before reading and writing can commence, a confirmation procedure is followed. This consists of reading the address portion of the records while the positioner is still moving. When the desired track is reached, the address read will correspond to the selected track address. After the first occurrence of this matching, additional time is taken to examine subsequent addresses to assure that the positioner has settled. This additional time is termed confirmation time. Thus the confirmation procedure assures reliable reading and writing.

OPERATIONAL PERFORMANCE

A disc file can be designed to yield a very high performance in a particular application, but generally such a design is not economic over a range of applications. The optimum economic compromise can be achieved only if software and hardware are studied together during the design of a new unit. The dp/f-5035 is an example of such a design. The characteristics of typical operations must be examined to evaluate how the basic design philosophy of the hardware should affect operations.

Random-Access Operations

The most characteristic operation of a disc file is that of obtaining access to a record or sequence of records located anywhere within the file. This is known as random-access.

If the only operation performed by a disc file were random-access to a record anywhere in the complete disc file, the operations would be slow. A fixed-head file would be the fastest device but, because of rotational latency, it is doubtful if it could perform at higher rates than between 30 and 50 random-access per second. Moving-head files would make from 5 to 20 accesses per second, depending on their design. Since a representative record length is

250 characters, the fixed-head file would transfer only 10,000 characters per second, and the moving-head file would transfer proportionately less.

Fortunately most operations performed by disc files are characterized by random-access to only a portion of the total capacity, and the access often includes sequences of records. These operations have the effect of increasing the data transfer rate with the consequence that system performance is improved.

In the dp/f-5035 the access time to a record is dependent on the location of the record relative to the previously accessed record. The components of access time are illustrated in Figure 4. If the new record is at the same position as

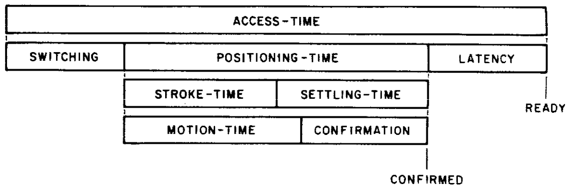


Figure 4. Components of Access Time.

the old, the access time is determined only by latency. If the new record is on a different positioner which had been previously positioned to the desired position, then the access time includes switching time and latency. If the new record is at a new position, then positioning time must also be included since the positioner must move. In the 5035, it is possible to eliminate the switching time when moving to a new position, when that positioner may be cleared previous to its selection.

The typical or average positioning time curves of the linear positioner used in the dp/f-5035 are shown in Figure 5. There are two separate curves because the confirmation time, in order to write, is longer than that required in order to read. These curves include all delays except latency and initial switching time.

The effect of data organization and utilization on random-average access-time is shown in Figure 6. The abscissa is the size of the data file being accessed in the application. The various curves show random-average access times

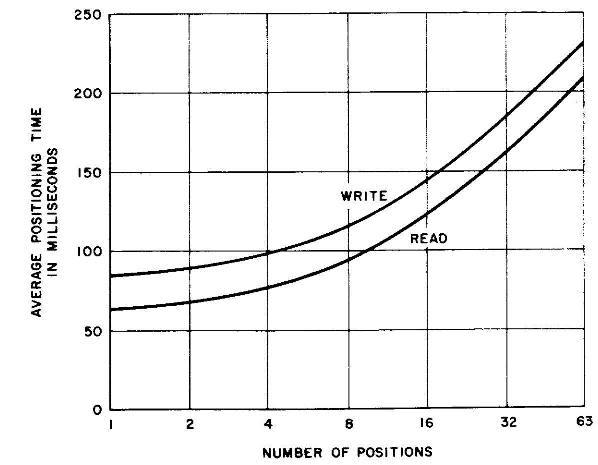


Figure 5. Typical Positioning Time Curves of the Linear Positioner.

for different data organizations. For purposes of simple illustration, all the curves were plotted assuming only one access channel instead of the two channels available in the 5035. The access times for typical data organizations using dual access channels were listed in Table II as a function of the size of the data file.

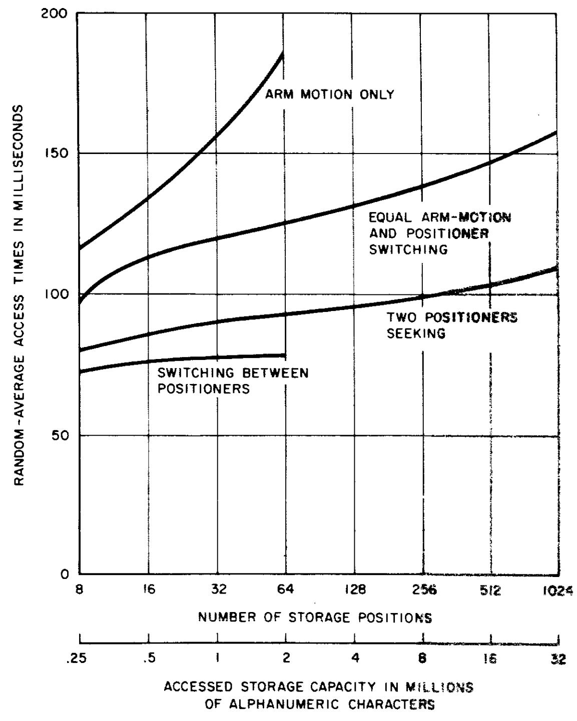


Figure 6. Random-Average Access-Times for a Single Access Channel.

The curves of Figure 6 include all delays and latency and are consequently true random-average access-times. The random-average assumes that in a given area of storage, all positions are equally probable. It also assumes that reading and writing take place an equal number of times, although reading usually takes place more frequently than writing. These curves are therefore biased to a somewhat higher value than would be encountered in practice.

The top curve at the left illustrates the behavior if access is obtained by moving a single positioner. The number corresponding to 64 storage positions is the random-average access-time of the positioner. This is the number normally quoted in specifications since it includes access to all positions. The points on the curve corresponding to 32, 16 and 8 positions demonstrate how rapidly the random-average access-time goes down as the area covered by the access is reduced.

The bottom curve at the left illustrates the behavior if switching between positioners is used without any arm motion. This assumes a single switching matrix and a single data channel. Of course, in the dp/f-5035 much shorter times would be experienced.

When the data is organized to utilize both arm motion and switching between positioners, as is the case for larger capacity data files, the "equal arm motion and positioner switching" curve applies. The random-average access-time is about 130 milliseconds to a typical large file. Much lower values are achieved if the data are organized so that more switching occurs than arm motion.

The last curve illustrates the effect of multiple seek where only one access channel is used, but the system contains two file modules. In this case, the typical access time is less than 100 milliseconds.

These curves illustrate the fact that in any disc file the typical access time is much less than the normally quoted random-average, since random-access to a single record in an entire file is only one of the operations performed in a disc file. In addition, random-access to a lengthy data transmission of 8,192 words reduces the ratio of access time to total operation time.

Examining the raw specifications is insufficient to predict the performance of a moving-head file. Performance will generally be better than one would expect from the specifications. In particular, the inherent low cost of the moving-head file makes it attractive when compared with fixed-head files of similar capacity.

Flexibility through Individual Positioners

The technique of using an individual linear positioner for each disc was adopted in designing the dp/f-5020 because it was believed that it would have significant advantages to the user. There is much more knowledge now regarding the application of disc files, and consequently the technique was re-examined in formulating the design of the present series. There is now no doubt that separate positioners have very real advantages. They increase computational speed and make the DISCfILE flexible and easy to use.

All the more recent disc files have multiple heads which can be switched electronically. This provides access to a moderate amount of data with only latency delays. Typically 20,000 to 100,000 characters can be accessed in this manner.

The advantage of systems which have individual positioners is that when larger data files are used, the programmer has more flexibility in assigning areas of storage to achieve high performance. If a disc file system contains a single positioner and comb, the most efficient organization is to keep the data of a file in cylinders close together to minimize positioning time. However, with disc files which contain individual positioners, positioning may be minimized by organizing data into a cylindrical matrix. Since each positioner may be located in a different position, the optimum area occupied by the data file is determined by the programmer rather than by the characteristics of the disc file.

This concept becomes very significant in many situations. Consider an application where an input must be processed by two data files simultaneously. Each of the data files could be stored in separate areas of the disc file. The two areas together could be considered as a cylindrical matrix. With the dual-access capability, both areas could be accessed simultaneously and hence achieve a high transfer rate. If only one positioner were available, the positioner would have to go back and forth to the two cylinders and spend lost time in motion. Thus two data files may be considered as a fractured cylinder or a cylindrical matrix.

Another example of flexibility is the condition which often arises in files of data. They change their size and shape with time, such as the project or contract or purchase order which persistently remains open long after all around it have been closed. The adaptability of the individual positioners allows for organizing other data around this area, that is data physically separated but contiguous from an access standpoint.

Disc files have become popular for another reason also. A great variety of tasks can be performed utilizing the same storage system. For instance, the disc file could be storing alternate programs in a real-time situation. It could also be processing data against a data file. If rapid access to the program area is required, several positioners could remain in that area. The others could be busy operating on data, and hence the total system operates efficiently.

If the data occupies most or all of the capacity of the disc file, the access is usually not random over the complete file. An example of this is an airline reservation system. Inquiries from a particular city tend to concern flights to or from that city. Also, the file must contain data for many weeks of flights but most inquiries will be limited to a narrow time band of a day or two. When the system takes advantage of such effects, its speed can be substantially greater than would appear from the random-average access-time.

Random or semi-random accesses tend to ease the problems of programming in almost all types of computation. This is particularly true in program storage, compiling, multi-programming, real-time processing and communications systems. In these cases very large transfers or dumps occur. The amount of data involved varies from 512 to 32,000 words. A large portion of the internal storage is changed during a dump or transfer. Very high transfer rates are of primary importance, but short access times are also necessary to locate an area into which a dump can occur, and then to locate the area from which the new data is transferred. In this situation the dual-access and multiple-seek can both be important since dumping and filling can be going on simultaneously. Also, other positioners can be getting ready to continue the operation if a large data transfer is used. It is also common practice to avoid latency by commencing the transfer from the first record which becomes available once the positioner has reached its position.

THE PERFORMANCE-COST CRITERIA

Any component of a data processing system must be evaluated in terms of its operational performance against the overall cost. Performance is defined as how much useful work is done per unit of time. It is a function of the characteristics of the equipment and how well it is utilized. The initial price is only part of the cost factor. The costs of integrating the equipment into the system, programming and maintenance must all be included. For essential memory devices, reliability is an important cost factor because failure can hold up the operation of the entire processing system.

It is almost impossible to evaluate one disc file against another by comparison of characteristics. Disc files, however, can be compared by considering their performance in the expected set of applications. Without knowing the precise applications, comparison of one disc file design with another must be based on rather arbitrary assumptions. If then a figure of merit is developed which includes speed, capacity and cost, the files may be compared on an arbitrary basis.

The Standard Operation Time

A useful but arbitrarily defined tool in evaluating disc files is the concept of a Standard Operation Time. This is the average of a weighted number of different operations. The operations and their weights should be chosen as intelligently as possible to reflect the current experience with regard to disc file utilization.

For the purposes of this paper, Standard Operation Time is based upon the following assumptions:

A. The typical area in which random-accesses occurs has a capacity of five million characters.

B. A typical block transfer or dump involves 2000 words (16,384 characters) of storage.

C. In computing the standard, equal weight is given to the time required to perform the following ten operations:

1. Random-access within a five million character area in order to
   a. Read a record
   b. Write a record
   c. Read and then write the same record
2. Random-access within two randomly placed five million character areas in order to
   a. Read a record
   b. Write a record
   c. Read and then write the same record
3. Random-access within the entire store to a block of 2000 words to
   a. Read from the first available record onwards
   b. Write commencing at a specified record
4. Latency delay only, and then
   a. Read a record
   b. Write a record

Performance, Capacity and Cost

The value of a disc file to a user is directly proportional to its capacity and inversely proportional to its Standard Operation Time. In the following computations, capacity has been expressed in millions of alphanumeric characters and Standard Operation Time in seconds. The cost is necessarily approximate but an estimate has been made in dollars of cost per operating hour. The Index of Value is

$$\frac{\text{Actual Storage Capacity *}}{\text{Standard Operating Time} \times \text{Cost per Hour}}$$

Note that in the same way that we have used a standard set of operations to establish index of value, we have also used a system configuration where only one DISCfILE system is attached to the computer. Other DISCfILE systems configurations, such as two or more systems attached to the computer, would yield different results. For example, two 5024's attached to a computer on a single channel would provide a multiple seek capability. Configurations such as this, or those with multiple systems and multiple channels, might yield a higher index of value.

The manner in which the performance of DISCfILES has improved is demonstrated by Table III.

APPLICATIONS

The dp/f-5035 is intended for use with high performance data processing systems. It is assumed that there are at least 8000 words of internal storage and that operations are in the low microsecond domain. The computer should also have modern input/output capabilities, including at least two channels with versatile interrupts under program control.

The first application area involves the use of the dp/f-5035 system as a multiple secondary store. The programmer may consider the storage as consisting of a number of separate secondary stores. He may select any two positioners of the set for completely overlapped reading or writing and be seeking or holding on two others at the same time.

One of the classical problems in the use of a disc file for secondary storage is how to associate the address in the disc storage with the name of the item in the data file. An anomaly results because the number of addresses in the DISCfILE is small relative to the almost infinite number of names which can be applied to items in the data file. The usual solution is to utilize an algorithm to transform the name of the item into its address. The transformation is not

* Note actual storage capacity is the net usable data storage capacity after allowing for address bits and inter-record gaps for format control.

TABLE III. Storage Capacity in Millions of Alphanumeric Characters, Standard Operating Time, and Index of Value for Various DISCfILE Systems.

DISCfILE System	Number of Files in System	Storage Capacity-Millions Char.	Standard Operating Time Seconds	Index of Value
5020	1	24	.224	12.2
5024	1	32	.186	18.9
	2	64	.146	27.3
5034	1	64	.186	27.1
	2	128	.146	37.8
5025	1	32	.085	30.5
	2	64	.070	47.7
5035	1	64	.085	47.7
	2	128	.070	67.0

always one-to-one, and "chaining" techniques are used to locate items transformed to a common address. Several accesses may be required to find the item.

The obvious solution to the problem is to maintain in primary storage a catalog or index table of the address-name relationships. Unfortunately, the storage requirements could exceed the capacity of the store. A workable solution utilizing the dual-access capabilities of the present system is to combine the techniques; namely, store a transformed catalog in one area and use the algorithm to locate a section of the catalog and then use the catalog to find the unique address. With appropriate overlapping and look-ahead techniques, the table look-up can be done in a time that is never longer than the time required to scan a two-link chain.

The technique of look-ahead is a tempting way of reducing execution time. With the availability of both multiple-access and multiple-seek, the technique of look-ahead may be simply implemented. Positions which are known in advance may be sought so that positioning time may be virtually eliminated. Also, look-ahead may be used when the programmer is not certain of the next address. He can usually make two "guesses" simultaneously and thus improve his probability of guessing right.

Look-ahead can be of considerable value in implementing a one-level store. By this means the programmer would be relieved of the responsibility for keeping track of where information is in the disc file. Wagner[5] believes that there will be breakthroughs in system organizations whereby the user will be able to behave as if he had a one-level store. This has been implemented by utilizing drums in the Atlas Supervisor[6] where all storage is organized into 512-word "pages." In this country several large scientific users have tried a similar approach using disc files with single access capabilities and it is believed that dual-access will provide significant improvements of these systems.

It is always interesting to consider specific examples where various methods of performing a task may be quantitatively compared. Hess[7] analyzed a sorting problem which consisted of sorting, on an 18 character key, 60,000 records of 16 words in an ascending sequence. The input was from 256 word blocks on tape and the output was also placed on tape in the same format. The analysis was performed for a hardware configuration consisting of an IBM 7090, IBM Model VI tape transports, and a two module IBM 1301 disc file. Hess compared the times for three ways of sorting: (1) The disc configuration as a random access device, which took 153.08 minutes; (2) The eight tapes with a tape sort, which took 14.78 minutes; and (3) The disc configuration so as to simulate 20 tapes, which took 13.06 minutes. The importance of the analysis was that the disc file, if properly used, could better the tape system by 12% in execution time.

Recently the same sort was analyzed with a hypothetical configuration in which the hardware was the same, except the dp/f-5035 was used as the disc file. Two major characteristics of the file were employed, its large capacity and its ease of achieving overlap. Using one file unit, it would be possible to simulate a 32 "tape" sort with a 16-way merge, but only 22 "tapes" were actually needed.

The input and ouput phases were, of course, tape limited as they were in the case of the previous analysis. There was an improvement in the output phase since the independently movable positioners permitted look-ahead and advance positioning. The total time for the sort, using the dp/f-5035, was 5.25 minutes; less than half of the best time in Hess' analysis.

When applications programmers first looked at disc storage, one of the most attractive possibilities that immediately came to mind was to use the disc as a secondary storage which could be accessed by two or more computers. A simple illustration of this requirement is the large central computer used primarily for complex processing. Input and output from such a computer is frequently done via tapes prepared by peripheral computers with attached card equipment and printers. Users and managers of these installations visualized that the substitution of the disc for some of the tapes would economically provide operational convenience and improve turn-around time.

A large number of studies were conducted of those disc files which were first made available for this purpose; namely, devices with a single-comb access. When one computer wanted to write on the outer edge of the disc, the other computer frequently would be interested in reading on the inner edge. Since these demands were completely asynchronous and unpredictable, it became apparent that each computer would be waiting an unacceptable time for the comb to be properly positioned to service it, and hence the system was somewhat impractical.

This difficulty is overcome by the dp/f-5035 with its dual-access to independent positioners which allows each of two computers to access the disc file independently at the same time. Note that either computer may be reading or writing without concern for what the other one is doing. The principle involved applies with added force to many military applications requiring access to a large data base.

In addition to two computers accessing a single file, a single computer performing two tasks may be accessing the file. Part of the disc file storage could be used to perform input/output conversion such as transforming from cards to file or file to printer. The other channel of the 5035 could be engaged in other normal processing activities. In the case of an input mode such as data transfer from cards to file, the data stored in the file is available to the computer in a random-access mode at any later time.

There are, of course, many other applications which are not considered in this short analysis. Moreover, specific and quantitative results have not been included since they are beyond the scope of this paper. As the equipment begins to be utilized, new applications will arise and quantitative results will be published. The great flexibility provided by the dual-access and multiple-seek features of the dp/f-5035 DISCfILE, coupled with its high reliability and comparatively low cost make it a valuable tool in mass storage technology.

REFERENCES

1. Donald K. Sampson, "DISCfILE Series dp/f-5020," Data Products Corporation, presented at the Informatics Inc. Disc File Symposium, March 6-7, 1963, Los Angeles, California.
2. R. K. Brunner, J. M. Harker, K. E. Haughton, and A. G. Osterlund, "Experimental Investigation of Pivoted Slider Bearings," IBM Journal of Research and Development, July 1959, Vol. 3, No. 3, p. 260.

3. S. ABRAMOVITZ, "Theory for Slider Bearings with a Convex Pad Surface," Franklin Institute Journal, 1955, No. 25, pp. 221-233.
4. W. A. GROSS, "Gas Film Lubrications," John Wiley and Sons, Inc., 1962.
5. FRANK WAGNER, "Scientific Computing—Three Significant Trends," Datamation, January 1962.
6. T. KILBURN, R. B. PAYNE, and D. J. HOWARTH, "The Atlas Supervisor," Proceedings of the Eastern Joint Computer Conference, Washington, D. C., December 12-14, 1961.
7. HERMAN HESS, "A Comparison of Discs and Tapes," presented at the Informatics Inc. Disc File Symposium, March 6-7, 1963, Los Angeles, California—to be published in the Communications of the ACM.

SYNTACTIC ANALYSIS OF ENGLISH BY COMPUTER–A SURVEY*

Daniel G. Bobrow
Bolt Beranek and Newman Inc.
Cambridge 38, Massachusetts

I. INTRODUCTION

A statement in a spoken language may be regarded as a one-dimensional string of symbols used to communicate an idea from the speaker to a listener. The dimensionality of the statement is limited by the need for presenting words in a single time sequence. However, evidence indicates that most information and ideas are not stored by people in one-dimensional arrays isomorphic to these linear strings. This implies that a speaker must use certain complex information manipulating processes to transform the stored information to a linear output string, and that a listener, in order to "understand" the speaker, must use another set of processes to decode this linear string. In order for communication to take place, the information map of both the listener and the speaker must be approximately the same, at least for the universe of discourse. Most important, the decoding process of the listener must be an approximate inverse of the encoding process of the speaker.

Education in language is, in large part, an attempt to force the language processors of different people into a uniform mold to insure successful communication. A certain preferred class of utterances is defined by an educated native speaker to be "sentences" of his language, and members of this class are the principal (but not only) vehicles of communication in the language. Communication via sentences occurs most often in written discourse, but is far less frequent in the oral mode.

Part of the information processing done in constructing and understanding sentences deals with the meanings of individual symbols (words), and part of this processing deals with words as members of classes of words, and with the structural relationships between classes.

The former is usually known as "semantic" processing and the latter, as "syntactic" processing. It is admittedly very difficult to draw a sharp line between the two, especially when definition of membership in a class of words is dependent upon the meaning of a word. In section II of this paper, we discuss the problems of syntactic classification of words.

Many different methods have been proposed for doing syntactic processing of English sentences, and a number of these methods have been implemented on digital computers. All such processes associate additional structures with the sentences of a language. Some programs demonstrate the generation of grammatical sentences from a set of syntactic rules for English. Other syntactic processors are used as preliminary processors for translation of English sentences to other representations of the same information (e.g., other natural languages, structures within the computer used

* This work was performed with support from the Council on Library Resources.

for information retrieval, etc.). A third set of programs are used in the study of English grammar, and find allowable parsings for a sentence on the basis of the rules given.

Section III of this article contains a survey of those theories of grammar which have served as a basis for syntactic processing by computer. The form of the rules for each grammar and a description of the syntactic structure associated with a sentence by each processor are given; reference is made to computer programs which have been written, and goals and present success of these programs are reviewed. Syntactic analysis programs written only for application to languages other than English are not described, nor any theories of grammar unique to such programs.

II. DETERMINATION OF WORD CLASSES

Conventional school grammars usually divide English words into eight classes. These eight "parts of speech"—noun, pronoun, adjective, verb, adverb, preposition, conjunction, interjection—are supposed to be the "natural" divisions of words into classes, and parsing, i.e., associating a syntactic structure with a sentence, is done in terms of these classes. However, difficulty arises when one tries to determine which words are in what class. For example[†], the usual definition of a noun is that "a noun is the name of a person, place or thing." But blue and red are the names of colors, and yet in expressions such as *the blue tie,* or *the red dress* we do not call *blue* and *red* nouns. In these examples, *blue* and *red* are called "adjectives," because an adjective is a "word that modifies a noun or pronoun." A large part of the difficulty here is that these two definitions are not parallel. The first tries to define the class of a work in terms of its lexical meaning, the second in terms of its function in the sentence.

Modern linguistic practice avoids this type of difficulty by taking the following operational approach. Two words are in the same word class if they appear in the same environments. Thus word classes are co-occurrence equivalence classes.

† These examples are taken from Fries.[13]

As an example of this type of definition of word classes, let us consider the word classes defined by C. C. Fries. He assumes that "all words that could occupy the same 'set of positions' in the patterns of English single free utterances must belong to the same part of speech . . . [we] make certain whether, with each substitution [into his sample frame], the structural meaning is the same as that of our first example or different from it." The frames Fries uses are the sentences:

A. The concert was good (always).

B. The clerk remembered the tax (suddenly).

C. The team went there.

Those words which could fit into the position occupied by *concert* in A, *clerk* and *tax* in B, and *team* in C are called Class 1 items. Class 2 items can replace *was* in A, *remembered* in B and *went* in C. The words in Class 3 can replace *good* in A (with appropriate lexical changes for consistency). However, this does not define Class 3 uniquely, and so frame A was modified, and Class 3 words are defined as only those which can fit into both blanks in:

```
The ____________concert was ____________.
        Class 3                   Class 3
```

Class 4 words must fit into the indicated position in:

```
      Class 3 Class 1 Class 2 Class 3 Class 4
(The) _____   _____   is/was  _____   there.
                                      here
                                      sometimes
```

or in:

```
      Class 1   Class 2          Class 1 Class 4
(The) _____  remembered (the)  _____  clearly
                                        so
                                        repeat-
                                        edly
```

or:

```
            Class 1          Class 2  Class 4
(The) _______________         went    clearly
                                      away
                                      safely
```

There is a large measure of overlap between classes 1, 2, 3 and 4, and the traditionally defined noun, verb, adjective and adverb respectively. Fries keeps the names distinct, however, because, for example, Class 1 both excludes certain words traditionally classified as nouns, and includes words not traditionally classified, as nouns.

In addition to the major word classes, Fries defines some 15 groups of function words. They are the words which serve as structural markers within a sentence. Although in normal text these function words would make up only about 7% of the number of distinct lexical items, the total number of function words would be approximately one third of the total number of words. Unlike most words in the four major classes, the lexical meaning of a function word is not clearly separable from its structural meaning in the sentence. For example, it is not difficult to define or associate appropriate experiences with such words as *horse, computer, run, fall, smooth* or *rapidly,* but conversely for *the, shall, there* (in "there is nothing wrong"), etc., all end in the common suffix "ly". We can tion words must be known as individual items to allow us to determine the structure (and then the meaning) of a sentence.

Having placed large numbers of words in the four major word classes, one notes that words in each class have certain common formal characteristics (this is untrue of the function word group). For example, many of the class 4 words, such as *rapidly, slowly, fairly, badly, etc.,* all end in the common suffix "ly". We can immediately classify "argled" as derived from the verb "argle" by its suffix. There are many other such formal clues to word classification. It is such clues and patterns in English that allow people to parse such sentences as:

"The ollywop stiply argled a golbish flappent."

while knowing the meanings of *none* of the content words.

Most programs which do syntactic analysis of English do not use the formal characteristics and immediate context of a word to determine its part of speech. Most use a dictionary lookup operation to find its possible classifications (*yellow* can be a noun, verb or adjective, etc.) and then resolve ambiguities during the parsing operation.

One program, written by Klein and Simmons[24, 25, 26] at SDC, does compute syntactic classifications for English words. Using a dictionary of suffixes and prefixes, common exceptions, and function words, the program tries to determine a classification for words in a sentence. Most ambiguities are resolved by use of the immediate context. For example, if an unknown word might be an adjective or noun, and is the blank in a context "*the* adjective ______ verb," the program unambiguously labels this word as a noun. The program successfully labelled over 90% of the words in a large text. Computation of word class is fast compared to dictionary lookup for a large vocabulary, but dictionary lookup must be done for the remaining 10% whose type cannot be computed. Klein and Simmons use these computed classifications as input for a parsing program.

Recently, Resnikoff and Dolby[46] of Lockheed reported briefly on another program which computes a part of speech for "any English word, though not always correctly." They report the accuracy on a trial of 150 sentences as "evidently high."

Their criterion of correctness is agreement with *either* the Oxford Universal or Merriam-Webster International Dictionary. The dictionary size and affix lists used were significantly smaller than those used in the Klein-Simmons program. The authors do not use context to resolve ambiguity, or use the computed parts of speech in further linguistic processing, although their program may eventually be used as a preprocessor for a parsing program written at Lockheed.

Only after classification of word types (by computation or lookup) can further syntactic analysis be done. Using only those word classes defined thus far, we discover the following anomaly:

"The boy which stood there was tall."

and

"The bookcase which stood there was tall."

must both be considered grammatical sentences, since the only change from one to the other is a substitution of one member of an equivalence class for another. However, a native speaker of English would call only the second grammatically correct. If we wish our grammar to act as a characteristic function for a language (and accurately specify which strings are sentences and which are non-sentences), then we must divide "nouns" into at least two subclasses, those associated with *which* and those with *who.*

Actually, as pointed out by Francis[12] we can recognize eleven pronoun substitute-groups in present-day English. They are those nouns for which we can substitute (singular-plural, where "/" indicates alternatives):

	Examples
1. he-they	(man, father, brother, monk, king)
2. she-they	(women, mother, sister, nun, queen)
3. it-they	(house, tree, poem, rock, complication)
4. he/she-they	(parent, child, friend, cook, tutor)
5. he/it-they	(bull, ram, rooster, buck, steer)
6. she/it-they	(cow, ewe, hen, doe, heifer)
7. he/she/it-they	(baby, dog, cat, one, other)
8. it/they-they	(group, committee, team, jury, crew)
9. he/she/they-they (or no plural)	(somebody, someone, anybody, person)
10. it (no plural)	(dirt, mathematics, poetry, music, nothing)
11. they (no singular)	(pants, scissors, pliers, clothes, people)

Again these subclasses can be determined in terms of co-occurrence in sentences of the given pronouns and nouns, but the classification system is now more complex.

Still further problems arise when we consider the following pair:

John admires sincerity.
Sincerity admires John.

The first is certainly grammatical, but how shall we classify the second string? We can call it a "grammatical but meaningless sentence". It is well-formed according to our rules of grammar but we can attach no meaning to the string. However, there is a large class of such sentences which would not occur in discourse. A system of analysis is complete only if it provides a means for distinguishing between such meaningless strings and grammatical sentences whose meaning can be understood. Whether this analysis should be done "syntactically", or done in a further "semantic" processing is moot.

One method proposed for including this discrimination in a "syntactic" processor involves further subclassification of the parts of speech thus far defined. For example, a noun would be classified as an abstract or concrete noun ("sincerity" is abstract; "John" is not), a countable or a mass noun ("apple" is a countable noun; "water" is not), etc. A verb could be classified by the type of subject it can have (e.g., an abstract noun cannot be a subject for the verb "admire"). Then "sincerity" is an abstract noun, and "admire" requires a concrete subject. Notice, however, that we have started to define noun subclasses by meaning and we thus enter the shadowy land between syntax and semantics. If our purpose is to process further the parsed sentences, this uncertain division is unimportant, but a theory of syntax based on a non-semantic association of signs with signs must define this line more precisely. Perhaps a distributional basis for defining all necessary subclasses can be found.

III. SYNTACTIC STRUCTURES AND COMPUTER IMPLEMENTATIONS

The processes by which people generate and decode verbal communications are not well understood. Somehow a speaker chooses a sequence of symbols from a repertoire to represent an "idea". He uses these symbols, together with structural markers, in a linear string to communicate to another person. A symbol may be a morpheme, or a word, or even a sequence of words.

In order for a speaker to be understood, he must use his content words within one of a number of preferred sequences separated by appropriate structural markers. A listener then uses this sequence and the structural markers to perform an inverse to the transformation made by the speaker, and decodes the sentence. Only certain preferred sequences of words can be decoded, and one large class of these preferred sequences contains the "sentences" of a language. One of the primary goals of linguistic theory is to provide a grammar, a set of rules, for a language, which will distinguish between the sentences and non-sentences of that language, and simultaneously, associate a useful structure with these linear strings. This associated structure should show the similarities of sentences which are (to a large extent) independent of particular lexical items, and should reflect similarities in the processing needed to understand these sentences.

Many different types of grammars have been proposed for English. We distinguish types of grammars by the form of the grammar rules, and by the type of structure the grammar associates with a word string. Following Chomsky[10], we shall call two grammars *weakly* equivalent if they generate the same set of sentences from the same initial vocabulary, or, analytically, if they classify the same strings as sentences and non-sentences. Two grammars are *strongly* equivalent if there is an isomorphism between the structural diagrams each associates with sentences.

A minimum criterion for any acceptable grammar of English is that it be weakly equivalent to the implicit grammar of a native speaker of English; that is, it should accept as sentences just those the speaker does. However, this implies a different grammar for each speaker, and there is a wide divergence in dialect between a university educated person and a street urchin. Although an uneducated speaker probably will accept more strings as sentences, the standard usually taken for a grammar is weak equivalence to the dialect of the educated person. However, grammars can be developed for both.

The following further criteria are also used as a basis for the acceptability of a grammar:

1. Two sentences such as

   "John hits the ball." and

   "Mary diapers the baby."

   which are considered structurally the same must be associated with the same structural diagram by the grammar.

2. A sentence such as

   "They are flying planes."

   which is ambiguous should be associated with a different structural diagram for each interpretation of the sentence.

3. Chomsky[9] talks of the "explanatory power" of a grammar as a criterion for acceptability. By this he means that an adequate grammar should provide an explanation, for example, of why

   "John hit the ball." and

   "The ball was hit by John."

   are semantically identical although the strings and associated structural diagrams are not similar.

4. All other things being equal, the better of two grammars is the one which is "simpler". However, no standard of simplicity (except conciseness) has ever been agreed upon.

5. A further non-linguistic criterion is the usefulness of the syntactic structure for further processing. Some examples of processing beyond syntactic analysis will be given later in this paper.

Dependency Grammars

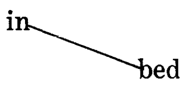
One of the conceptually simplest grammars is the dependency grammar developed by Hays[19, 21]. According to this type of grammar, a sentence is built up from a hierarchy of dependency structures, where all words in the sentence, except for an origin word (usually the main verb), are related to the sentence by a dependence on another word in the sentence.

"The concept of dependency is a broad generalization of syntactic relatedness. Adjectives depend on the nouns they modify; nouns depend on verbs as subjects and objects, and on prepositions as objects; abverbs and auxiliaries depend on main verbs; prepositions depend on words modified by prepositional phrases. Insemantic terms, one occurrence depends on another if the first modifies, or complements the meaning of the second—this includes rules of agreement, government, opposition, etc."

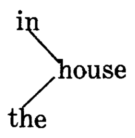
For example, the string "the house" is made up of two elements with *the* dependent on *house*. The article *the* delimits *house*, and is, thus, dependent upon *house* for its meaning in the sentence. (This is a very pragmatic use of the word meaning.) In the phrase "in bed", *bed* is dependent upon the preposition *in* to connect it to the rest of the sentence, and thus depends upon this preposition.

A word can have more than one dependent; in the phrase "boy and girl", both *boy* and *girl* are dependent upon the "governor" of the phrase, the conjunction *and*. Similarly, in the phrase "man bites dog", both *man* and *dog* are dependent upon the verb *bite*, in the type of dependency used by Hays. In the dependency analysis used by Klein and Simmons[24, 26] this is not so.

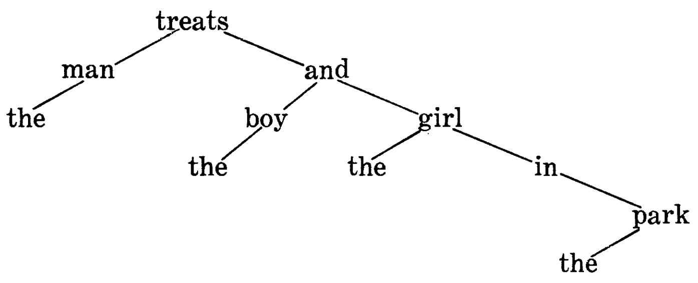
A graphic representation of the syntactic structures associated with some strings by a dependency grammar is shown below.



"in bed"
(a)



"in the house"
(b)



"the man treats the boy and the girl in the park."
(c)

These structures are downward branching trees, with each node of the tree labelled with a word of the string. There is no limit to the number of branches from a node. A word is directly dependent upon any word immediately above it in a tree.

These trees are constructed by investigating all possible connections between words in the initial string. The defining postulate of a dependency grammar[21] is that "two occurrences can be connected only if every intervening occurrence depends directly or indirectly on one or the other of them". Thus, local connections may be made first, and then more distant connections may be tested for validity. This localization assumption is very convenient for computer processing.

Another powerful property of a dependency grammar is the isolation of word order rules and agreement rules. The structure tables for the grammar define allowable sequences of dependencies in terms of word classes. For example, a noun followed by a verb may be in subject-verb relationship. If this word order criterion is met, agreement in number may then be checked.

If, for each successful connection made, the rule which generated the dependency connection is recorded, the *use* of a particular word occurrence in the sentence (e.g., as subject, object, object of preposition, etc.) can be attached to the tree.

Programs Using Dependency Grammars

Several different computer applications have been made using the ideas of dependency grammars. The initial efforts were made by D. G. Hays[21] and his associates at RAND. Much effort by his group has gone into making a dependency grammar for Russian. A number of interesting heuristics are used in this analysis program to find a single preferred parsing of a sentence. First, closest linkages are given preference. Hays proposed making a statistical study of the frequency of occurrence of connections, and when parsing, those connections with maximum weight would be tested first. Another scheme focuses on a specified occurrence that can be independent. Any connection that can be made to this occurrence is checked first. Thus the RAND system tends to make ambiguous nouns into subjects or objects of verbs rather than modifiers of other nouns. Hays states that Garvin calls this the "fulcrum" approach to syntactic analysis.

A parsing program for English utilizing a dependency grammar was written by Hugh Kelly, also of RAND (personal communication). The grammar developed was not extensive, but was sufficient for several applications. One use of this parser was as a preliminary processor of English input for a heuristic compiler of Herbert Simon[50].

Maurice Gross[16] of the MIT Research Laboratory of Electronics wrote a general parsing program for dependency grammars. In one left to right scan, it generates all structures compatible with a given dependency grammar table. No extensive grammar was developed for this program.

The concept of word dependency is also used by Klein and Simmons[24, 27] of SDC in a computer program designed to generate coherent discourse. In general, when a grammar is used to generate sentences, most of the sentences are semantic non-sense. Such non-meaningful but grammatical sentences as "The colorless green ideas sleep furiously." or "John frightens sincerity." are more the rule than the exception. However, Klein and Simmons use a base text of English sentences and observe the dependency relations among words of this text. For example, the adjective *original* may be dependent upon the noun *ideas;* however, in ordinary discourse, *green* will never be connected to *ideas* in this relationship. Thus, following the ordinary grammar restrictions, plus these additional relevance dependency restrictions, the authors are able to generate surprisingly coherent sentences.

Very little work has been published in English about foreign work in syntactic analysis programs for English. Those translations which are available indicate that almost all the Russian work is being done using dependency grammars. For example, Moloshnava[40] talks of using "the known tendency [of words] to combine with other classes of words".

Andreyev[3] describes a dependency analysis which is to be used in translating English into an intermediate language. This intermediate language is then to be translated into Russian. However, none of this has yet been fully implemented on a computer.

Immediate Constituent Grammars

Another type of grammar used to describe the syntax of English is called an immediate constituent grammar. The basic premise of this type of grammar is that contiguous substrings of a sentence are syntactically related. Brackets or labelled brackets are used to demark syntactically significant substrings. These brackets may be non-overlapping, or nested. The sentence is enclosed in the outermost bracket pair. A formal theory of grouping as a basis for analysis is given by Hiz[22]. This type of grammar is called a context-free phrase structure grammar by Chomsky[10].

To illustrate this syntactic structuring let us consider the sentence:

"The man ate the apple."

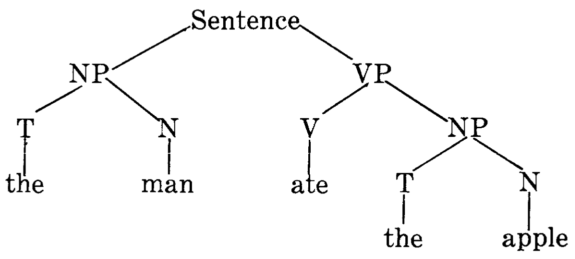
Bracketting the syntactically significant phrases, we get:

((the man) (ate (the apple)))

These unlabelled brackets demark the three principal substructures of the sentence. However, usually when bracketting is done with a phrase structure grammar, the brackets are labelled in some way. For example, we can use "{ }" to enclose a sentence, "[ ]" to enclose a verb phrase, and "( )" to enclose a noun phrase. Then the bracketted sentence would be:

{(the man) [ate (the apple)]}

A more common way to represent the constituent structure of this sentence is with a tree diagram such as that below:



This diagram should be read from bottom to top. It shows that *the* is of word class T (definite article), *man* and *apple* are nouns (N) and *ate* is a verb (V). A *T* and a consecutive *N* combine into the single syntactic unit denoted by *NP* (a noun phrase), and *V* and *NP* form a verb phrase (*VP*). An *NP* followed by a *VP* is a Sentence.

Note that these trees are very unlike those generated by a dependency grammar. The nodes of a dependency grammar tree are labelled with words of the sentence, and the structure shows relations between words. A constituent structure tree has the names of syntactic units as labels for all nodes but the bottom node of the tree. Only at the lowest nodes do the words of the sentences appear. The tree structure shows the combination of syntactic units into higher level constituents.

The rules of these context-free phrase structure grammars are of the following simple form:

$$A_1 + A_2 + \dots A_n \rightarrow B_i$$

$A_1, A_2, \dots, A_n$ and B_i are labels for syntactic units, and this rule is to be interpreted as stating that if a string of syntactic categories $A_1 + A_2 + \dots A_n$ appears in a sentence, the elements of the string may be combined into a single unit labelled B_i. The "+" indicates concatenation. B_i may even be one of the A_1 occurring in the left side of this rule: This is not uncommon in English—for example, if *A* is an adjective and *N* is a noun, then A + N → N is a rule of English; that is, an adjective followed by a noun may be treated as if it were a noun.

The rules given were for the decoding or parsing of English. However, these rules can be easily modified to provide a generation scheme for sentences. The rule given earlier may be rewritten as:

$$B_i \rightarrow A_1 + A_2 + \dots + A_n$$

We now interpret this rule as the instruction, "rewrite B_i as the string $A_1 + A_2 + \dots + A_n$." The initial symbol with which we start is *Sentence* and by successive rewriting we can generate a string of words which is a grammatical sentence (according to our grammar).

Let us illustrate this process by means of the following simple grammar:

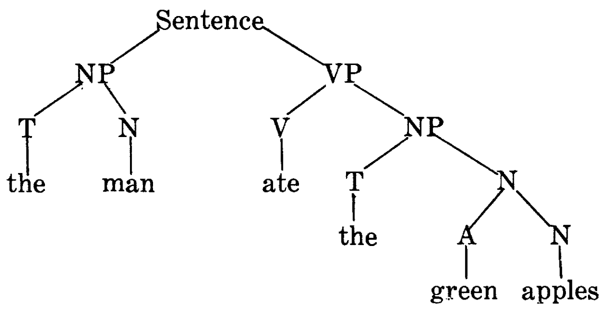
(i)	Sentence → NP + VP
(ii)	NP → T + N
(iii)	N → A + N
(iv)	VP → V + NP
(v)	T → the
(vi)	A → green, red
(vii)	N → man, apples
(viii)	V → ate, hit

Then the following lines are a "derivation" of the sentence "The man ate the green apples". The numbers to the right refer to the rules used in generating this sentence from the rewrite rules.

Sentence	
NP + VP	(i)
T + N + VP	(ii)
the + N + VP	(v)
the + man + VP	(vii)
the + man + V + NP	(iv)
the + man + ate + NP	(viii)
the + man + ate + T + N	(ii)
the + man + ate + the + N	(v)
the + man + ate + the + A + N	(iii)
the + man + ate + the + green + N	(vi)
the + man + ate + the + green + apples	(vii)

The second line of this derivation is formed in accordance with rule (i) by rewriting *Sentence* as NP + VP, and so on.

This derivation can be represented by the following tree structure:



This diagram of course does not indicate in what order the rules of the grammar were used, and many derivations would yield this same tree structure. However, only the information in the diagram is necessary to determine the constituent analysis of the sentence.

A substring is defined to be a "constituent" of a sentence if, from all the words of the substring (and only those words), we can trace back to some single node of the tree. If this node is labelled Q, then we say that this substring is a "constituent of type Q". Note that "the green apples" is a constituent of type *NP*, but "the green", since both these words cannot be traced back to a single node from which *only* they originate, is *not* a constituent of the sentence.

Sometimes it is possible to construct two correct distinct diagrams for the same sentence. Chomsky calls this phenomenon "constructional homonymity". When this occurs, the sentence in question is ambiguous.

These ambiguities arise in two distinct ways. The first type of ambiguity is exemplified by the phrase:

"the boy and girl in the park"

In this string the prepositional phrase *in the park* may be thought of as modifying only the noun *girl*, or the entire noun phrase *boy and girl.* There are two distinct trees corresponding to the two choices for bracketing this string, representing a genuine ambiguity in meaning.

The second type of ambiguity is caused by the frequent occurrence of homographs in English. Two words are homographs if they look identical, but are different parts of speech. For example, the *"run"* in "we run" and the *"run"* in "the run in her stocking" are homographs. In trying to find a parsing for a sentence, all classifications given in the dictionary must be tried for each word. Although sometimes the local context is sufficient to decide which homograph of a word is present, often such homographs cause ambiguous interpretations of a sentence. An interesting example is a short sentence analyzed by Oettinger[42] for which he finds three distinct parsings because of homographs. The sentence is:

"They are flying planes."

In the first parsing, *are* is classified as a copulative, and *flying* is an adjective modifying *planes.* We can bracket this as

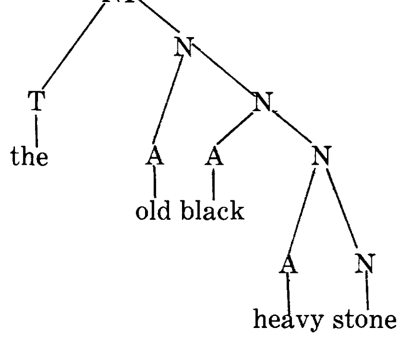
"(they (are (flying planes)))"

For the second parsing, *are* is an auxiliary for the verb *flying,* and *planes* is the object of this verb—this parsing corresponds to the parenthesization

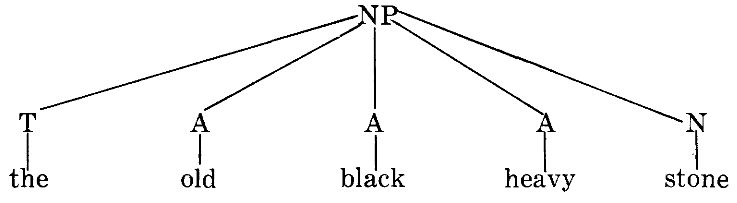
"(they (are flying) planes)".

The third parsing is left as an exercise for the reader.

All the rules given above for this simple phrase structure grammar allowed each constituent to be rewritten as at most two subconstituents. Although such binary rewrite rules are sufficient to link all connected constituents, there is some debate among linguists as to whether the resulting structure adequately represents the structure of the language. In most cases, binary structures are obviously adequate, but where one has a string of coordinate adjectives, such as in "the old, black, heavy, stone", one might like to think that each of the adjectives in this phrase modifies the meaning of just *stone,* rather than *black* modifying the noun phrase *"heavy stone",* etc. The two diagrams below show the difference between the binary structure and the multiple branching coordinate structure:



a. binary structure



b. coordinate structure

For the sake of computational simplicity most computer parsing programs which perform immediate constituent analyses utilize grammars with only binary rewrite rules.

Computer Implementations of Immediate Constituent Grammars

An extensive immediate constituent grammar for English is being developed by Jane Robinson[47] of RAND. All the rules of this grammar are binary combination rules. The grammar uses a complex four-digit coding scheme for the classification of the constituents of English.

Early testing of this grammar was done with a parsing program written at RAND in the list processing computer language IPL-V. Currently, the grammar is being used by, and tested on, a parsing program written for John Cocke of IBM. Cocke's program is much faster than the IPL program since it is written in machine code for the IBM 7090. An efficient algorithm invented by Cocke, and described in an article by Hays[10], is used for generating all possible parsings for a sentence. All two word constituents of the sentence are generated, and stored. Using the constituents of length two, and individual words, all constituents of length three are generated. Similarly, all constituents of length n are generated sequentially from constituent strings of smaller length, until all constituent strings of length s, the length of the sentence, are generated. The generation of constituents is done by a fast hash-code table lookup for pairs of subconstituents.

The time for parsing a short sentence (under ten words) is under one second. All possible parsings for a long sentence of 30 to 40 words can be found in less than one minute. The number of possible parsings grows very rapidly with the length of a sentence. Usually long sentences contain many prepositional phrases, and it is often possible for each such phrase to be a modifier of any of several nouns. Each ambiguity increases the number of parsings multiplicatively.

The output from the RAND parsing program is a printout of the resulting tree. Later versions of the program attach flags to the words indicating in more readable terms the function of each word within the sentence.

A very extensive set of linguistic processing programs has been written at the Linguistics Research Center of the University of Texas. The theory of language propounded by Lehman and Pendegraft[32] is an explication of Morris'[41] theory of signs, with language processing stratified into (at least) three levels. These strata are syntactics, the relationship of signs to signs; semantics, the relationship of signs to the things they denote; and pragmatics, the relationships of signs to their interpretations. Operational programs perform lexical analysis, and then syntactic analysis for any immediate constituent grammar (not restricted to binary constituents). The lexical analysis program acts as a preprocessor for the syntactic analysis program. It accepts as input a string of allographs (i.e., indivisible lexical constants, e.g., letters or strokes) and gives as output a list structure of all segmentations of the input string which includes all items in the input in some acceptable lexical segment. These lexical segments are denoted by "syntactic constants" which are used as parts of the structures constructed by the syntactic processor. Syntactic processing is stochastic in that a probability of occurrence is associated with each syntactic structure. All ambiguities are carried forward in a list structure with no redundancies. At each step in a construction, all possible structures may be used, or only a specified number of highest-probability alternatives.

There is no restriction on the length of an input string, and the Texas system is designed for high volume processing. Corpus maintenance programs in the current system maintain about 850,000 words of English text, 750,000 of German, 3,000 in Chinese and 30,000 of Russian. Immediate constituent grammars for each of these languages are being developed. At present the English grammar consists of approximately 5400 rules. With the addition of semantic rules, pragmatic rules, and interlanguage transfer rules, it is hoped that this system will provide good automatic computer-produced translations.

A parsing program utilizing a simple phrase structure grammar has been written by Klein[24] at SDC. A notable feature of this program is that, as mentioned earlier, the classification of words is not done primarily by dictionary lookup, but by computation.

From the phrase structure parsing certain dependency relationships will be extracted which will be useful in question-answering by the Synthex[20, 21, 37] program. For example, to answer the question "Do worms eat grass?", the following facts are noted about the question. It is an active construction, with *worms* as subject and grass as part of a verb-modifying phrase. These relationships imply certain dependency relationships among *worms, eat* and *grass*. From these, it follows that the second of the following two "information rich" statements:

"Birds eat worms on the grass."

"Most worms usually eat grass."

is the better of two potential answers—though both contain the same "content" words, *worms, eat,* and *grass*.

Klein has recently written a parsing program which simultaneously computes all phrase structure analyses and dependency analyses for an English sentence (personal communication). The dependency relationships found are those needed for his scheme for generation of coherent discourse.

Parker-Rhodes[44] of the Cambridge Laboratory Research Unit has proposed a lattice model for language from which all possible parts of speech, syntactic units and grammar rules may be derived. Not all syntactic structures will appear in all languages. Superimposed upon this immediate constituent grammar is a form of dependency analysis, and items are grouped (from right to left) only if a governer (head of a dependency structure) is found surrounded by an arbitrary number of dependents. The output of the parser is a bracketed structure. A program embodying the Parker-Rhodes procedures for English has been written for the Cambridge University Computer EDSAC II.

L. Dale Green[15] of Electro-optical Systems has written a "multiple path heuristic parsing program." Written in the list processing language, IPL, it is designed to be used as a tool to test any immediate constituent grammar. It recursively generates all legal substructures, cutting off generation when there are more empty nodes in these substructures than items to fill these nodes. Further heuristics to reduce search are planned but have not yet been coded.

James C. Brown[8] of Florida State University has built an utterance generating and resolving device utilizing an immediate constituent grammar. He has developed a small grammar for English and a complete one for Loglan[7]. All his programming was done in COMIT.

Programs to generate grammatical strings from arbitrary immediate constituent grammars are trivial if programmed in a list processing language. This author programmed one in LISP in half an hour, and debugged it in two runs.

Syntax in Backus Notation

There is a straightforward isomorphism between a syntax given in Backus notation[5] and an immediate constituent grammar. Therefore, any of the syntax directed compilers which have been written could act as parsers for English. However, since no grammars for English have been developed in this form for any of these programs, we will not consider them further.

Categorical Grammars

Syntactic analysis performed on the basis of rules of an immediate constituent grammar involves two independent dictionary lookup operations. Parsing operations handle words as members of classes. Therefore, on a first pass each word occcurrence in a sentence is associated with its possible syntactic categories. Then subsequent references to the list of grammar rules are made to determine which adjacent constituents in a string can be combined into higher level constituents. This lookup operation is iterated each time a new category replaces two lower level syntactic markers.

With large vocabulary lists and many grammar rules, these lookups take up a disproportionate amount of time and this time grows rapidly with list size. A computer has only a limited immediate fast memory store, and much necessary material must be placed in slow auxiliary storage, thus further slowing the parsing operation. It would be advantageous if the lookups could be replaced by a computational procedure independent of vocabulary size and number of rules in the grammar. As has been previously mentioned, Simmons and Klein at SDC are using computation instead of dictionary lookup to resolve the problem of word classification for most words.

The second problem, of avoiding a lookup operation for grammar rules, has also been a subject of study. The original work was done by Ajdukiewicz[1] and has been continued by such people as Bar-Hillel[4] and Lambeck[30]. This work is based on the following type of idea (as expressed by Lambeck):

"In classical physics it was possible to decide whether an equation was grammatically correct by comparing the dimensions of the two sides of an equation. These dimensions form an Abelian group generated by L, M, and T, with these quantities admitting fractional exponents. One may ask whether it is similarly possible to assign 'grammatical types' to the words of English in such a way that the grammatical correctness of a sentence can be determined by a computation of this type."

Obviously such a language coding could not be commutative because, for example, the sequence "the boy" is allowable as a syntactic unit in a sentence, and "boy the" is not. However, some codings for parts of speech have been developed which have certain of the properties of dimensions in physics.

Bar-Hillel's "categorical grammar" and Lambeck's "calculus of syntactic types" both use grammatical types denoted by complex symbols which have dimensional properties. Let us illustrate this class coding with an example. Recall that an adjective used as a prenominal modifier has the property that the resulting adjective-noun string can again be treated in the same way as the original noun. Bar-Hillel assigns a noun a grammatical code *n*, and an adjective code $\frac{n}{[n]}$ (this is written as (n/n) in Lambeck's notation). The string has type $\frac{n}{[n]} \cdot n$ (where "." indicates concatenation). Performing a "quasi-arithmetic" cancellation from the right, we compute the code for the string type as $\frac{n}{[n]} \cdot n = n$. As another example, an intransitive verb such as *eats* in "John eats", is given type $\frac{s}{(n)}$. The string "John eats" therefore has type $n \cdot \frac{s}{(n)} = s$. The indicated resulting type is *s*, or sentence, after cancellation.

If the basic grammatical categories are denoted by s, n_1, n_2, . . . , m_1, m_2 . . . then the operator categories of a grammar are denoted by:

$$\frac{s}{(n_1)\ (n_2 \ldots (n_i)\ [m_1] \ldots [m_j] \ldots} ; i + j \geq 1$$

The marker "s" denotes the syntactic type of a string which is a sentence of the language. As indicated, a term enclosed by parentheses, e.g., (n_k), can be cancelled from the left; a term enclosed in brackets, e.g., $[m_k]$, can be cancelled from the right. (Lambeck uses the notation (n s) and (s n) for left and right cancellable terms.)

In a categorical grammar one defines a sub-only two basic categories, *s* and *n*. An example of the complexity of structure types that occur is shown in the figure below, which illustrates the derivation of the syntactic type of the phrase "very large house."

phrase: very large house

$$\text{types:} \quad \frac{\frac{n}{[n]}}{\frac{n}{[n]}} \qquad \frac{n}{[n]} \cdot n \rightarrow \frac{n}{[n]} \cdot n \rightarrow n$$

Note that this algebra is not associative. The derivation shown starts by combining the two left hand terms first to get a derived type of $\frac{n}{[n]}$ for the substring "very large". Then the right hand cancellation is made yielding the grammatical type *n* for the entire string. If, in attempting to compute the type of this substring, the right hand pair were combined first, we would be left to find the type of a string with the two constituent codes:

$$\frac{\frac{n}{[n]}}{\left[\frac{n}{[n]}\right]} \cdot n$$

But this pair is not further reducible. Thus, the pairing must go the other way if this substring is to have a single derived category.

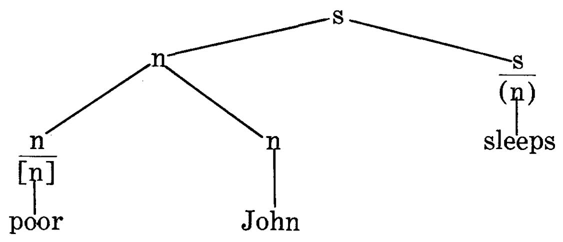
A derivation which leads to a derived category which is a *single* operator category or a single basic category is called a proper derivation. A derivation is a sequence of lines in which each succeeding line differs from the one immediately preceding it in that exactly one pair of adjacent elements has been combined to form a new derived category. A parsing for a sentence is any proper derivation whose terminal symbol is *s*.

The figure below shows the only proper derivation of the simple sentence "Poor John sleeps".

$$\begin{array}{ccccc}
\text{Poor} & & \text{John} & & \text{sleeps} \\
\frac{n}{[n]} & \cdot & n & & \frac{s}{(n)} \\
 & n & \cdot & & \frac{s}{(n)} \\
 & & s & &
\end{array}$$

In a categorical grammar one defines a substring *t* to be a constituent of a sentence *s* if in a proper derivation of *s* there is included a proper derivation of *t*. This definition is equivalent to our earlier definition of constituent in terms of nodes of a tree.

The derivation given can be represented by a tree:



Each member of a substring which is a constituent can be traced back to a single node. Note the following phenomenon which is not immediately obvious. The boundaries of a constituent are very dependent upon the context of the substring.

For example, "John sleeps" has an immediate derivation to a single category marker, *s;* however, in the context "Poor John sleeps" there is no proper derivation for this sentence in which "John sleeps" is reduced to a single constituent.

The mathematical problems implicit in dealing with strings of markers of the type shown have been further investigated by Bar-Hillel[5] and his associates, by Lambeck and by R. P. Mitchell[39] of the Lockheed Corporation. Lambeck has even extended a method first proposed by Gentzen for an intutionist propositional calculus to give a decision procedure for this algebra as a deductive system. However, very little has been done to develop an extensive grammar of this form for English.

Computer Implementation of a Lambeck Algebra

A categorical grammar, or Lambeck algebra has implicit in its markers for the word classes the combination rules for the grammar. Only one program that I know of has been written to take advantage of this method of avoiding an iterated grammar rule lookup. This program was written by Glenn Manacher of the Bell Telephone Laboratories in Murray Hill, New Jersey (personal communication).

The program was written in the COMIT programming language, and at present can parse sentences which do not contain conjunctions, commas, appositive clauses, or elliptical constructions. However, Manacher believes that these restrictions are not inherent in this type of grammar.

The program is a multipass scanner which makes local linkages first, and recursively builds up the parsed structure. Some ten basic categories are used, and the operator categories for the grammar are defined in terms of these basic categories.

The grammar and program successfully handle such complex constructions as those in "he wants a man to see the car work". The output of the program is a parenthesization of the sentence to show the constituent structure and a list indicating the use of each word within the sentence, e.g., "object of the verb 'work' ", etc.

Predictive Syntactic Analysis

Predictive syntactic analysis is based upon an immediate constituent grammar of a very restricted form. The restrictions are associated with the order in which words in the input string are scanned during analysis. In most immediate constituent parsers many passes are made over the input string, and substructures internal to the string are often determined before constituents containing the initial words are considered. A predictive parser analyzes a sentence in one left to right scan through the words. Before describing the structure of the syntactic rules in terms of an immediate constituent grammar, let us first consider the mechanisms used in predictive analysis.

When a person reads the word "the", he expects that later in the same sentence he will find a noun delimited by this occurrence of "the". Similarly, he would predict from the appearance of an initial noun phrase the later occurrence of a verb phrase (if he were expecting to read a sentence). Predictive analysis works in a similar manner. An initial prediction is made that a sentence is to be scanned. From the initial word in the sentence, and this prediction, new, more detailed predictions are made of the expected sentence structure. For example, if the first word in the sentence were "they", the grammar table states that with an initial pronoun the prediction of a sentence, *S*, may be replaced by predictions of a predicate and then a period, or by a prediction of successively, an adjective phrase, a predicate, and then a period—or by seven other sets of predictions.

One set of these predictions at a time is placed on a push-down list, with the prediction which must be satisfied first at the top of this list. A push-down list, as you know, is a list structure which has the property that the last item placed in the list will be the first item retrievable from the list. It is similar to a dish stack in a cafeteria in which the last dish placed on top of the stack must be the first taken from the stack.

As each successive word of a sentence is scanned, its syntactic type, s_j, and the topmost prediction in the stack, p_i, are compared. It may happen that this prediction p_i can be completely satisfied by a member of the class s_j. For example, if the prediction is "noun phrase", the proper noun Tom completely satisfies this prediction, and "noun phrase" would be removed from the top of the stack. If not, new predictions compatible with s_j and p_i are generated, and these new predictions are placed on top of the push-down stack. If no new predictions can be made, this implies that the set of earlier predictions was incorrect and an alternative path must be tried. If the terminal punctuation mark of the sentence is reached and all the predictions made have been satisfied, then this set of predictions represents a parsing for the sentence. If some predictions remain unsatisfied, or there are no more predictions in the stack, but words remain, then this line for parsing has failed. If all sets of predictions fail, the sentence is ungrammatical.

Each set of predictions for a word class marker, s_j, and a top prediction, p_i, is a form of an immediate constituent binary rewrite rule for the predicted structure p_i. The two constituents of p_i are always of the following restricted form. The left element is always a terminal marker (in fact, s_j). This element can only be rewritten to give a single symbol, i.e., a word of the sentence. The right subconstituent of p_i is a complex symbol—a list of further predictions. Gross[16] has shown that predictive grammars are weakly equivalent to the class of unrestricted immediate constituent grammars. His paper also contains a proof of the weak equivalence of immediate constituent grammars and both dependency and categorical grammars. The original proof of equivalence for dependency grammars was done by Gaiffman at RAND. Weak equivalence of two grammars, you will recall, means that both grammars recognize exactly the same set of strings as the set of grammatical sentences (or equivalently, generate the same strings).

Computer Implementation of Predictive Syntactic Analysis

Computer implementations are based on work first done by Ida Rhodes[45] for a Russian parsing program. The most extensive grammar for English for predictive analysis by computer has been developed at Harvard by Kuno and Oettinger[29, 42]. By making use of a set of "prediction pools" (i.e., a number of push-down lists), they have been able to obtain all the parsings of syntactically ambiguous sentences very efficiently:

"Branchings caused by homography and by multiple functions of a given word class are followed in a systematic loop-free sequence in which each partial path is traversed only once. Different paths that reach the last word in a sentence correspond to acceptable syntactic structures of the sentence."

The program is written in machine code for an IBM 7090, and it has, for example, found the four possible parsings for a 23-word sentence in 1.2 minutes.

Several heuristics are used to increase the efficiency of the program. For example, if at any time the sum of the minimum number of words to fulfill all the predictions in the push-down list exceed the number of words left in the sentence, this set of predictions is immediately discarded. This technique is extraneous to the theory of predictive analysis, but does not change the class of sentences which will be parsed. Another cutoff measure used is the depth of nesting of a predicted structure. Evidence given by Miller[38] indicates that people cannot understand sentences with a depth of nesting greater than about seven. Thus, structures whose pool of predictions imply a much greater depth are discarded on the assumption that such depths are rarely if ever reached by well-formed sentences. Theoretically, but not practically, this changes the set of parsable sentences for the program.

Kuno and Oettinger use a coding trick in the grammar to obtain greater efficiency. Some predictions are optionally droppable so that two separate parallel paths need not be followed; for example, the prediction of an adverb immediately after a verb is droppable since the structure of the remainder of the sentence is essentially independent of such an occurrence. Using such a coding, the initial segment of the sentence need not be analyzed twice, once for the prediction set with, and once for the set without the adverb. Another favorable feature of this program is complete separation of the rules of the grammar from the active program. The active program need not be changed when a change is made to a grammar rule. As mentioned previously, this program finds all possible parsings of a given sentence compatible with its grammar.

Robert Lindsay[33, 34, 35] has also written a parsing program based upon predictive analysis techniques. His program finds just one parsing for a sentence, even for ambiguous sentences, and not always the "correct" parsing, i.e., the one that would be understood by a person. However, the parsing found usually contains sufficiently correct substructures for the additional "semantic" processing done by Lindsay's program.

Lindsay was interested in the problem of extracting certain information from text, and being able to answer certain questions on the basis of this information. The universe of discourse for the questions was the relationship of one member of a family to another (e.g., uncle, brother-in-law, etc.). The input text could contain extraneous information, i.e.,

"Tom bought his daughter Jane a candy bar."

After a complete syntactic analysis, including linking the "his" in this sentence to "Tom", the noun phrases dealing with family relationships are examined and the information is stored on a representation of a family tree. Questions about relationships are answered by finding the points on the tree representing the individuals, and then using this tree structure, computing the relationship corresponding to the path through the family tree between these points.

This predictive syntactic analysis program served adequately for the further processing of the sentences that was desired. It had, however, the following restrictions. The language was limited to basic English. This is a system extracted from normal English by C. K. Ogden[43] which contains approximately 1500 vocabulary items, and eliminates some syntactic complexities of the language. However, the words are chosen sufficiently carefully (they are not the 1500 most common words of English but the most useful) so that most thoughts expressible in unrestricted English can be expressed in this dialect. None of the simplifications involved are contradictions of standard English grammar or semantics.

In addition to this restriction to basic English, Lindsay's parsing program cannot handle some appositive phrases (i.e., those with no punctuation indicating their right end), quotations, interjections, or certain special constructions frequently used with questions. However, Lindsay is continuing work on the problem of syntactic analysis by computer, and may extend the scope of his program.

Another parsing technique closely related to predictive analysis is that implemented on an IBM 7090 for Sydney Lamb at Berkeley (personal communication). The program works with any binary non-discontinuous immediate constituent grammar with the following additional information. In each pair of "tactic codes" (syntactic names), one of the pair is marked as presupposing the other. For example, in the construction $A + \mathrm{N} \rightarrow \mathrm{N}$ the adjective A presupposes the noun; since the adjective occurs first we say that the A predicts N. If the tactic code which is the presupposer is the second member of a pair, we say it "retrodicts" the other tactic code. Retrodictions are related to the "hindsight" technique of Rhodes[45], and the predictions to predictive analysis. However, these presuppositions are strictly local in nature. Using these presuppositions, the program determines all possible analyses in one left to right scan.

Program efficiency is gained by grouping all construction formulas for one presupposer. The addresses in the grammar table correspond to syntactic codes, and the entries are computer instructions to be executed. Thus, very fast parsing is obtained. English and Russian grammars are being developed for this program as one portion of a much larger computer language processing system.

Phrase Structure Grammars with Discontinuous Constituents

The phrase structure grammars we have considered thus far have allowed only contiguous elements of a string to be syntactically related. In generating a sentence, a single syntactic constituent may be replaced by two continguous subconstituents (for *binary* immediate constituent grammars). Rewrite rules of the form $\mathrm{A} \rightarrow \mathrm{B} + \mathrm{C}$ are sufficient to generate most syntactic structures found in English. However, consider the following sentence:

"He called her up."

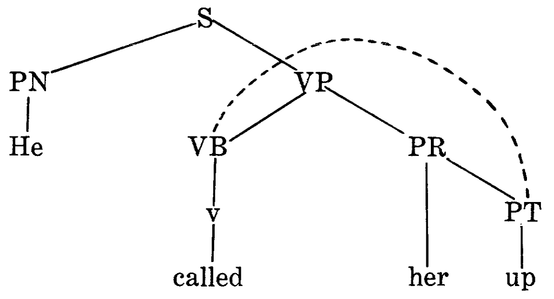
The word "up" is part of the verb structure, and intuitively we think of "call up" as one constituent. However, to account for the syntactic connection between *call* and *up*, and to restrict the rewrite rules to contiguous subconstituents, a large number of rules would have to be added to the grammar—one for each type of element that might appear between *call* and *up*.

To provide a concise notation to describe this type of discontinuous syntactic form which appears in many languages, Professor Yngve[53] of MIT added another rule format to simple phrase structure grammars. Rules in this format are called binary discontinuous rewrite rules.

These rules are written in the form:

$$\mathrm{A} \rightarrow \mathrm{B} : \mathrm{C}$$

The interpretation for such a rewrite rule is as follows. If, in the generation of a sentence we have a string XAY, where Y is a single syntactic marker, then we may rewrite this string, using the rewrite rule above, as XBYC. In general, when using the rule shown, A is replaced by B, and C is inserted in the string to the right of B, but separated from B by exactly one constituent marker. This rewrite rule is undefined if A is the right hand element of the string. Shown below is a set of rules used to generate the sentence, "He called her up". On the left is a structural diagram for the sentence.



$$\begin{aligned} S &\rightarrow PN + VP \\ VP &\rightarrow VB + PR \\ VB &\rightarrow V : PT \\ PN &\rightarrow \text{he, her} \ldots \\ V &\rightarrow \text{called} \ldots \\ PT &\rightarrow \text{up} \end{aligned}$$

Note that the structural diagram is no longer a simple tree structure, and therefore relationships within the sentence cannot be indicated just by bracketing. Thus, this type of grammar is not strongly equivalent to the immediate constituent grammars discussed previously. However, Mathews[37] has shown that the two types of grammar are weakly equivalent.

Computer Implementations of Discontinuous Constituent Grammars

One of the earliest computer programs utilizing a discontinuous constituent grammar was a program written by Yngve[54] for the generation of English sentences. A grammar was written which was sufficient to generate the first ten sentences of a children's story book. Then, using these rules and the vocabulary in this text, 100 sentences were generated randomly. The sentences generated were "for the most part quite grammatical, though of course, non-sensical." Here again is the question of when semantic non-sense impinges upon grammatical correctness.

Another program for generation of English sentences from a discontinuous constituent grammar was written in the COMIT programming language by B. K. Rankin at the National Bureau of Standards. The grammar used was one in a series of languages which are subsets of English, and which are to be used for information retrieval. Rankin and Kirsch intend eventually to be able to generate sentences which describe pictorial information, and answer questions in English about previously stored pictorial and textual information (personal communication).

Donald Cohen (personal communication), also of the National Bureau of Standards, has invented an algorithm and written a program which will recognize and analyze any sentence according to the rules of an arbitrary discontinuous constituent grammar. The result of the analysis is essentially a nested listing of the order in which rewrite rules could have been applied to generate the sentence, and the results of applying these rewrite rules.

The program gains efficiency by making a preliminary examination of the sentence to determine which rules can possibly "cover" which elements. Then only those pairs of elements of which both can be covered by one rule need to be considered as candidates for combination into a single constituent.

A parsing program for a discontinuous constituent grammar, with a "self-organizing heuristic program" has been written by Kenneth Knowlton[28]. Knowlton used a grammar which contained some extreme overgeneralizations. He then provided a training sequence for his parsing program. He showed it correct hand-parsings for sentences. The program abstracted probable partial structures from within these sentences. Using this information the first parsing found by the program for a sentence was most often the "correct parsing", although the overgeneralizations implied by the rules of the grammar would have allowed many incorrect parsings. The program attempted to parse 300 sentences written in basic English. It used at each attempt its experience on previous sentences and the hand-parsed correct answer given for these sentences. A limit effort was imposed for the parsing of each sentence, and the attempt to parse a sentence was abandoned if the effort expended exceeded this level. Of the approximately 150 sentences for which a parsing was found within the time limit, about

90% were parsed correctly. The results indicate that learning can be successfully applied to the problem of finding a good parsing for a sentence.

An extensive discontinuous constituent grammar for English was written by G. Harman at MIT. This grammar has some special properties which are related to questions that arise in connection with transformational grammars. Therefore, we shall postpone discussion of this effort until after an exposition of transformational grammars.

Transformational Grammars

All the syntactic theories thus far described are weakly equivalent to what Chomsky calls a context free phrase structure grammar. Chomsky[10] raises several objections to such restricted grammars, and proposes additional types of rules to be added to phrase structure grammars.

First, he proposes that the form of the rewrite rules be generalized to ZXW → ZYW. Z and W are the context of the single symbol X, and Y may be a string of one or more symbols. Z and W may be null, but in general, the set of elements Y that may be substituted for X is dependent upon Z and W, the context of X. A grammar with such rules is called a context sensitive phrase structure grammar.

To illustrate the necessity for such rules in the generation of English sentences, consider the generation of the two parallel sentences "he runs" and "they run". The generation for both proceeds from the initial symbol *S* to the string PR + VI (PR = pronoun, VI = intransitive verb). Then if PR is rewritten as "he", VI must be rewritten as "runs" and not "run", to be grammatical. Similarly, if PR is rewritten as "they", "run" for VI is a necessary substitution for grammaticalness. Of course, two separate rules could be used which would specify either both pronoun and verb as singular, or both as plural. However, one context sensitive rewrite rule would express this more concisely.

Another more serious objection to phrase structure grammars is the mathematical limitation on the type of strings producible by such grammars. Chomsky[11] has shown that strings of indefinite length of the form abca′b′c′, in which a′ is dependent upon a, b′ upon b, etc., cannot be produced by a phrase structure grammar. An example in English, pointed out by Bar-Hillel, is the construction involving "respectively", e.g., "Tom, Jane and Dan are a man, woman and programmer, respectively."

Although theoretically such statements can be extended indefinitely, there is some question as to how long a sentence of this type can be understood by a person. Certainly the finiteness of his available memory—even allowing him to use paper—imposes some limit. However, such limitations need not be made explicitly in the grammar.

Another linguistic objection raised by Chomsky is that the strong intuitive relationship between two such sentences as "The man drives the car" and "The car is driven by the man" is not explained in any way by a phrase structure grammar. Similarly, "the dog is barking" and "the barking dog" should be related within a "good" grammar, according to Chomsky.

As a solution to these problems, Chomsky has proposed an additional set of rules beyond the usual phrase structure rewrite rules. After the generation of sentences by a phrase structure grammar, Chomsky proposes that there be transformation rules which can transform one sentence into another sentence of the language. For example, one such transformation would take a sentence in the active voice *(the man drives the car)*, and transform it into a sentence in the passive voice *(the car is driven by the man)*. In addition, some transformations may apply to pairs of sentences, and combine the pair into one sentence. One such transformation would transform the two sentences "the boy stole my wallet", and "the boy ran away", into the complex sentence "the boy who ran away stole my wallet".

Such transformations, as Chomsky[9] points out, have as domain and range, sets of "P-markers", that is, tree structures associated with strings by a phrase structure grammar. They are not defined on terminal strings. In addition to specifying how a terminal string is to be changed, a transformation must specify what the "derived" P-marker of the new sentence is.

With the introduction of transformation rules, the basic phrase structure grammar can be simpler. Only a very simple set of "kernel sentences" of a language need be generated. All other complex sentences can be generated by use of transformation rules on sets of these kernel sentences. In addition, if certain "semantic" restrictions are to be included in the grammar (i.e., *frightens* may have *John* as an object, but not *sincerity*), these restrictions need be listed only once. For example, restrictions on the generation of objects for active verbs need not be repeated for the passive construction, since it is taken from a fully developed sentence in the active voice. These restrictions will be carried over implicitly. On the other hand, for a simple phrase structure grammar, such restrictions would have to be listed explicitly for both voices.

Computer Implementation of Transformational Grammars

Walker and Bartlett[52] of the Mitre Corporation are developing a parsing program based on a transformational grammar for use with an information retrieval system. Analysis of a sentence is performed by generation of possible strings of constituents as suggested by Mathews[36]. Each of the terminal strings generated is matched against the input sentence. When a match is achieved, a parsing has been found, and the parsing routine has determined the kernel sentence of the input string. The program also records the transformations used to embed the kernel sentences in the input string. To limit the large search space of possible kernels and transformations that might be generated, certain heuristics are used to find plausible transformations. For example, if a sentence ends in a question mark, then it is certain that at some point the question transformation was used.

Walker and Bartlett give an example of the types of kernels to be found in analysis of a question for the sentence "What are the airfields in Ohio having runways longer than two miles." The kernel sentences found are:

1. X's are the airfields.
2. The airfields are in Ohio.
3. The airfields have runways.
4. The runways are long.
5. Two miles are long.

Using the information about how the kernel sentences are transformationally related, these kernels will be translated into another language which is information retrieval oriented. This portion of the program has not yet been implemented, and thus the value of this type of syntactic analysis as a preliminary processor has not been determined for this task.

Janet Andress[2] of IBM is constructing a general purpose program to facilitate study of Chomsky type generative grammars. Given a set of context sensitive phrase structure rules, and a set of transformation rules, the program generates exactly one structure of each syntactic type possible within the grammar. The linguist may insert these rules in a very compact notation. At present only the phrase structure rule generation portion has been programmed but Andress expects eventually to be able to handle all of Lees'[31] transformational grammar.

A Phrase Structure Grammar with Complex Constituents

As mentioned earlier, Harman[17] has written a generative grammar without transformation rules which overcomes most of the difficulties associated with a discontinuous constituent phrase structure grammar. Additional power is introduced into the grammar by the use of complex symbols for syntactic markers. The rewrite rules of the grammar are written in terms of these new notational abbreviations. An example of a complex syntactic marker that might be used is:

"SENT/SUBJ ABSTR, OBJ ANIM".

This marker may be interpreted as a phrase marker for a sentence which has an abstract subject and an animate object. This notation is based upon facilities provided in the COMIT programming language, and thus is easily manipulable on a computer.

The rewrite rules allow either the top, or the subscripts (after the "/") to be rewritten independently, and certain properties (i.e., subscripts) may be carried to a subconstituent or discarded.

One of the nice properties of this type of grammar is that "semantic" restrictions can be accounted for at a high level, instead of when specifying the terminal string. In addition, both passive and active constructions are generated from one sentence specification, thus indicating the close relationship of these forms. This compares favorably with this same property of transformational grammars.

Harman based his grammar on the most comprehensive transformational grammar of English he could find[‡], and claims that his program will generate roughly the same set of sentences as the transformational grammar will. The two grammars are of approximately the same length, and because the phrase structure grammar has an unordered set of rules (this is untrue of a transformational grammar), it may be somewhat easier to write. Thus, the use of this complex subscript notation appears to overcome the principal difficulties which led to the introduction of transformational grammars. These two types of grammars are neither strongly nor weakly equivalent. However, it remains to be seen which type of analysis and which structures are most fruitful for further use on the computer. The additional generative power of transformational grammars should be kept in mind, but such facility may never be needed in practice.

String Transformation Grammars

Zellig Harris and his associates at the University of Pennsylvania have developed a method of syntactic analysis which is intermediate between constituent analysis and transformational analysis. Harris gives, in his book[18], a very lucid account of his work which we summarize briefly below.

The basic assumption underlying the string analysis of a sentence is that the sentence has one "center" which is an elementary sentence, and which represents the basic structure of the sentence. Additional words within the sentence are adjuncts to these basic words, or to structures within the sentence. Analysis consists of identifying that portion of the input string which is the center of the sentence, and adjoining the remaining words, in segments, to the proper elements of the sentence. For example, Harris gives the following anlaysis:

> "Today, automatic trucks from the factory which we just visited carry coal up the sharp incline."

trucks carry coal is the center, elementary sentence

today is an adjunct to the left of the elementary sentence

automatic is an adjunct to the right of trucks

just is an adjunct to the left of visited
etc.

In the analysis each word is replaced by a marker for its syntactic category. When several constituents can be strung together such that this pluri-constituent can be replaced by a marker of a constituent within it, then this endocentric construction (i.e., one expanded from an elementary category by adjoining) can be split into this head and its adjuncts. Iterating over all segments of the input string one finally obtains the center of the string.

Later work by Joshi[23] tends to make the results of a string analysis more like the results of a Chomsky transformational analysis. A sentence is resolved into a number of kernel sentences such that each main verb in the sentence is part of its own kernel. Some phrases with implicit verbs are also decomposed. An example of this latter type of decomposition is given in the analysis of "the violinist arrived late" into "N_1 arrived late", and "N_1 plays the violin". However, these kernels are identified only from the string, not from the structure of the associated syntactic tree, as in a Chomsky transformational analysis.

Computer Implementations of String Transformational Grammars

Each of the methods of syntactic analysis developed by Harris and his associates have been designed with computer implementation in mind. The method reported in Harris's book was implemented on the UNIVAC I computer, and was run successfully in 1959. The transformational decomposition program described by Joshi was never completed.

‡ An improved version of Lees'[31] transformational grammar.

The general arrangement of the work done by the UNIVAC program was as follows:

1. Dictionary lookup of each word, and replacement of the word by its category mark or marks.
2. Resolution, where possible, of multiple category marks for single words by use of local context.
3. Multiple scans through the string—some passes from the left, some from the right. Each scan tries to segment the sentence into "first-order strings".

For example, to find noun phrases a scan is made from right to left. Whenever a noun is found, a noun phrase bracket is opened (on the right). The scan continues to the left, accepting all words which can be part of this phrase. When a left delimiter is found, such as the occurrence of an article, the phrase is closed and the scan is continued. Additional scans are made until no more groupings into first-order strings can be made. This string of symbols (zero and first-order) is then checked for well-formedness (against a set of standard patterns). All alternative segmentations are checked and all the resultant successful parsings are given. This UNIVAC program will analyze a very large class of English sentences. Space limitations within the UNIVAC were, in general, the reason for the omission of those sentence types not included. The same general methods would work for almost all sentences.

Naomi Sager[48], another associate of Harris at the University of Pennsylvania, has directed the programming of a predictive procedure for string analysis. At each position all possible extensions of the center string are predicted, plus right adjuncts of the current category, left adjuncts of the next category, etc. The procedure is analogous to phrase structure predictive analysis. The program is written in IPL, and recursively finds all possible string analyses of a sentence.

Another syntactic analyzer based upon Harris's methods was that used in the Baseball[14] program written at Lincoln Laboratory. This program accepted a restricted class of English questions about a limited universe of discourse —baseball statistics—and answered these questions. The syntactic analysis portion of this program performed essentially the first three steps of the procedure given above for Harris' program. Then, assuming well formedness, the segmented string was converted into a standard set of questions—a specification list—which was then used to obtain the answer to the question asked. Thus, in this case all the syntactic analysis necessary for the desired further processing was a phrase structure grouping.

Householder and Thorne[51] use some of Harris' techniques for bracketing syntactic units, but the result of their analysis is a string in a different order than the input string—a linearized dependency order. Much of the processing that their program does is described by a sequence of ad hoc rules for unambiguously determining the word class of English homographs, and defining clause boundaries. The output of the syntactic analysis is a list of the kernel sentences of the input sentence. They hope to use this output for a question answerer.

CONCLUSION

This paper has reviewed a wide variety of theories of grammar and modes of output resulting from many types of syntactic analysis programs. Without a further goal than syntactic analysis for its own sake, we are limited to judging these programs by some arbitrary non-operational criteria of elegance, explanatory power and simplicity. A sufficient proof of the goodness of any of these theories lies only in its usefulness for further processing. Until a method of syntactic analysis provides, for example, a means for mechanizing translation cf natural language, processing of a natural language input to answer questions, or a means of generating some truly coherent discourse, the relative merit of each grammar will remain moot.

REFERENCES

1. AJDUKIEWICZ, "Die Syntaktische Konnexitat"; *Studia Philosophica*; Vol. 1 (1935), 1-27.
2. ANDRESS, JANET, "A Program for the Study of Generative Grammars", IBM Research Paper RC-820, November 9, 1962.

3. ANDREYEV, N. D., "Linguistic Aspects of Translation", *Proceedings of 9th International Congress of Linguistics,* 1962, Cambridge, Massachusetts.
4. BAR-HILLEL, Y., "A Quasi-Arithmetical Notation for Syntactic Description", *Language,* Vol. 29, No. 1, January-March 1953.
5. BAR-HILLEL, Y.; GAIFFMAN, C.; and SHAMIR, E.; "On Categorical and Phrase Structure Grammars", Research Council of Israel, Vol. 9F, No. 1, June 1960.
6. BACKUS, J. W., "The Syntax and Semantics of the Proposed International Algebraic Language of the Zurich ACM-GAMM Conference", *ICIP,* Paris, June 1959.
7. BROWN, JAMES C., "Loglan", *Scientific American,* June 1960.
8. BROWN, JAMES C., "Simulation Model-Making and the Problem of System-Controlled Human Behavior", *Faculty Conference on Computer Applications,* Florida State University, Tallahassee, Florida, April 1962.
9. CHOMSKY, N., "On the Notion 'Rule of Grammar' ", *Proceedings of the Symposium in Applied Mathematics,* Vol. 12, p. 6.
10. CHOMSKY, N., *Syntactic Structures,* Mouton and Company, S-Gravenhage, 1957.
11. CHOMSKY, N., "Three Models for the Description of Language", *IRE Trans. on Information Theory,* Vol. IT-2, September 1956.
12. FRANCIS, W., *The Structure of American English,* Ronald Press, New York, 1958.
13. FRIES, C. C., *Structure of English,* Harcourt Brace and World, 1952.
14. GREEN, B. F.; WOLF, A. K.; CHOMSKY, C.; and LAUGHERY, K.; "Baseball: An Automatic Question-Answerer", *Proceedings of the Western Joint Computer Conference,* Los Angeles, California, May 1961.
15. GREEN, L. D., "A Multiple Path Heuristic Parsing Program", Electro-optical Systems, Inc., Pasadena, California, 1963.
16. GROSS, M., "On the Equivalence of Models of Language Used in the Fields of Mechanical Translation and Information Retrieval", Mechanical Translation Group Memo, MIT, 1962.
17. HARMAN, G. H., and V. H. YNGVE, "Generative Grammars without Transformation Rules", MIT RLEQPR #68, 1962.
18. HARRIS, Z. S., *String Analysis of Sentence Structure,* Moulton and Company, The Hague, 1962.
19. HAYS, D. G., "Basic Principles and Technical Variation in Sentence-Structure Determination", RAND P-1984, May 1960.
20. HAYS, D. G., *Computer Applications in the Behavorial Sciences* (Chapter 17), Edited by H. Borko, Prentice-Hall, 1962.
21. HAYS, D. G., "Grouping and Dependency Theories", *Proceedings of the National Symposium on Machine Translation,* Edited by H. P. Edmundson, Prentice-Hall, 1961.
22. HIZ, H., "Steps Toward Grammatical Recognition", *International Conference for Standards on a Common Language for Machine Searching and Translation,* Cleveland, 1959.
23. JOSHI, A., "A Procedure for a Transformational Decomposition of English Sentences", TDAP 42, U. of Pennsylvania, 1962.
24. KLEIN, S., "Automatic Decoding of Written English", Ph.D. Thesis, Univ. of Calif. Berkeley), 1963.
25. KLEIN, S., and R. F. SIMMONS, "Automated Analysis and Coding of English Grammar for Information Processing Systems", SDC Report, SP-490, August 1961.
26. KLEIN, S., and R. F. SIMMONS, "A Computational Approach to Grammatical Coding of English Words", *Journal of the ACM,* July 1963.
27. KLEIN, S., and R. F. SIMMONS, "Syntactic Dependence and the Computer Generation of Coherent Discourse", *Mechanical Translation* (in press).
28. KNOWLTON, K., "Sentence Parsing with a Self-Organizing Heuristic Program", Doctoral Dissertation, MIT, August 1962.
29. KUNO, S., and A. G. OETTINGER, "Syntactic Structure and Ambiguity of English",

Proceedings of the International Federation of Information Processing Congress, 1963.

30. LAMBECK, J., "On the Calculus of Syntactic Types", *Proceedings of Symposia in Applied Mathematics*, Vol. XII, 1961.

31. LEES, R. B., "The Grammar of English Nominalizations", Part II, *Int. J. Amer. Ling.*, Vol. 61. 26, No. 3.

32. LEHMAN, W. P., and E. D. PENDEGRAFT, "Machine Language Translation Study Report #16", Linguistics Research Center, University of Texas, Austin, Texas, June 1963.

33. LINDSAY, R. K., "Inferential Memory as the Basis of Machines Which Understand Natural Language", p. 217-233, *Computers and Thought*, McGraw Hill, 1963 (in press).

34. LINDSAY, R. K., "The Reading Machine Problem", Unpublished Doctoral Dissertation, Carnegie Institute of Technology, 1960.

35. LINDSAY, R. K., "Toward the Development of Machines Which Comprehend", University of Texas, 1961.

36. MATHEWS, G. H., "Analysis by Synthesis of Sentences in a Natural Language", *First International Conference on Machine Translation and Applied Language Analysis*, HMSO, London, 1962.

37. MATHEWS, G. D., "Discontinuous One Way Grammars", MIT RLEQPR #68, pp. 193-194, 1962.

38. MILLER, G. A., "The Magical Number Seven-plus-or-minus-two", *Psychological Review*, Vol. 63, 1956.

39. MITCHELL, R. P., "A Note on Categorical Grammars", *Proceedings of the First International Congress on Machine Translation of Languages and Applied Language Analysis*, 1961.

40. MOLOSHNAVA, T. N., "An Algorithm for Translating from the English to the Russian". Translation in *Foreign Developments in Machine Translation and Information Processing*, No. 11, USSR.

41. MORRIS, C. W., "Foundation of the Theory of Signs", *International Encyclopedia of Unified Science*, Vol. 1, No. 2, University of Chicago Press, Chicago, 1955.

42. OETTINGER, A., and S. KUNO, "Multiple-Path Syntactic Analyzer", *Proceedings of the International Federation of Information Processing Congress*, 1962.

43. OGDEN, C. K., "A System of Basic English", Harcourt-Brace, 1934.

44. PARKER-RHODES, A. F., "A New Model of Syntactic Description", *First International Conference on Machine Translation and Applied Language Analysis*, HMSO, London, 1962.

45. RHODES, IDA, "A New Approach to the Mechanical Syntactic Analysis of Russian", *Mechanical Translation*, Vol. 6, November 1961.

46. RESNIKOFF, H., and J. L. DOLBY, "Automatic Determination of Parts of Speech of English Words", *Proceedings of the IEEE*, July 1963.

47. ROBINSON, JANE J., "Preliminary Codes and Rules for Automatic Parsing of English", RAND RM3339, December 1962.

48. SAGER, NAOMI, "A Procedure for Syntactic Analysis", *Proceedings of the IFIP Congress*, Munich, 1962.

49. SIMMONS, R. F., KLEIN, S., and MCCONLOGUE, K.; "Co-Occurrence and Dependency Logic for Answering English Questions", SDC Report SP-1155, April 1963.

50. SIMON, H., "Experiments with a Heuristic Compiler", *JACM* (in press).

51. THORNE, J. P., "Automatic Language Analysis", Indiana University, RADC-TDR-63-11, ASTIA Document No. 297381, December 1962.

52. WALKER, D. E., and J. M. BARTLETT, "The Structure of Languages for Man and Computer: Problems in Formalization", *First Congress on the Information Sciences*, 1962.

53. YNGVE, V. H., "A Model and an Hypothesis for Language Structure", *Proceedings of the American Philosophical Society*, Vol. 104, No. 5, 1960, p. 444.

54. YNGVE, V. H., "Random Generation of English Sentences", Memo 1961-4, Machine Translation Group, RLE, MIT, 1961.

THE COMPUTER-STORED THESAURUS AND ITS USE IN CONCEPT PROCESSING*

Clayton A. Shepherd
Manager, Information Retrieval
Federal Government Marketing
UNIVAC
Division of Sperry Rand Corporation
Washington, D. C.

INTRODUCTION

As the state of the art of computer development has advanced, the documentalist has turned more and more to mechanization for solutions to many of his problems. The process of retrieval of information, especially the searching of coded indexes prior to the selection of the document images themselves, has captured the lion's share of applications effort. The retrieval phase, however, often poses relatively minor difficulties; the most crucial point in the documentation process—the one which may contribute more than any other to its over-all success or failure—is that of indexing the documents prior to their entry into the search files.[1] The time-honored computer principle of "GIGO," garbage in-garbage out, could apply nowhere better than in the indexing process. Because of the crucial nature of this function, it usually proves to be the most expensive operation of the information center, the presence of a goodly assortment of data processing equipment notwithstanding.

The Indexing Problem

Unfortunately, efforts to mechanize the indexing process have usually met with limited success. Many factors contribute to the inherent difficulty the computer undergoes in attempting to "understand" the documents being processed. Statistical indexing techniques ameliorate the situation only slightly.[2] Such approaches are largely ineffectual primarily because the author of a given document is likely to be dealing in concepts, and is using various words in various combinations throughout the paper only in order to express those concepts effectively. Maron[3] errs in saying that words and sentences are but one step removed from the things and events they describe. Rather, they are *two* steps removed: the factor, so elusive to the indexer, which exists between the real word and the written representation thereof, is the *ideational concept* which exists in the mind of the writer and which is conveyed to the reader through the medium of words on paper. It is thus the task of the indexer, whether man or machine, to assign terms to a given document

* Work on this project was done in cooperation with the Institute for Advancement of Medical Communication, and was supported in part by a grant from the National Institutes of Health.

on a concept basis rather than merely lifting words from the sentences without regard to content.

The indexing problem, then, is one of conveying to the retrieval system the various concepts which have been expressed by the author, in as accurate and complete a manner as possible. One method of accomplishing this goal is that of utilizing a staff of very highly trained indexers, each so competent in his own technical field that he is able to comprehend the precise ideational content of the document and thereby index it adequately . . . an obviously expensive proposition, albeit an effective one.

An alternative, one that presents far less expense, and yet insures that the intellectual content of the documents is thoroughly understood, is that of relegating the indexing process to the authors themselves. Certainly a highly structured relationship must exist between the documentation center and the author group for such a system to operate effectively; however, in controlled situations, such as that in which various research laboratories are responsible to a central office, author-indexing could be utilized to excellent advantage. More and more the call is heard for a sharpening of the author's responsibility in coping with the information problem.[4]

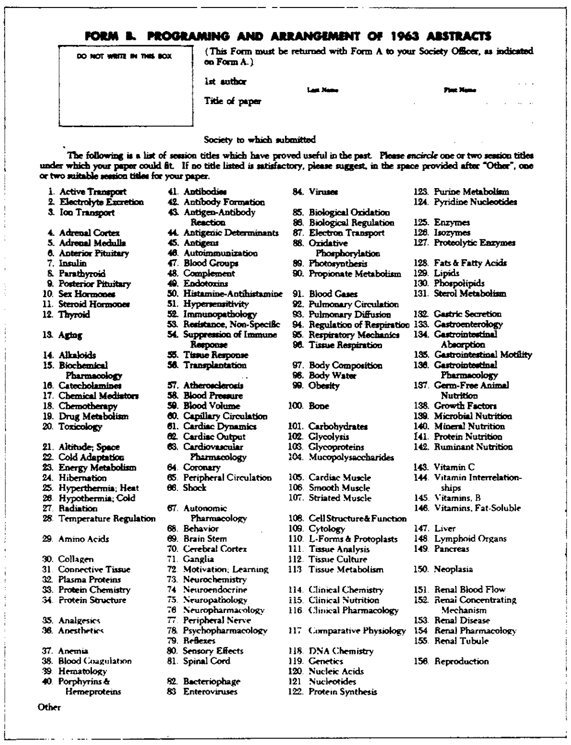
The present paper deals with such an effort, in which the authors contributing papers to a large scientific conference submitted both *specific indexing terms* and *conceptual statements*, which were supplied to the computer for analysis, cross-referencing, and final production in the form of a printed index. Analysis of the specific index terms presented little problem; however, processing the natural language concepts demanded novel techniques. In this respect the use by the computer of a specialized thesaurus showed itself to be quite successful and may prove to be an effective operational tool in the future in coping with the indexing problem.

The Federation Project

The Federation of American Societies for Experimental Biology is an organization made up of six separate professional groups, all vitally concerned with the biological sciences, which bonded together primarily because of a closely overlapping interest. The Federation holds its annual conference in the spring, at which time between two and three thousand papers are presented. Prior to the meeting, *Federation Proceedings* is distributed to all members; this publication contains both the abstracts of presented papers and a subject index. The Federation first approached UNIVAC in the fall of 1959 to investigate the feasibility of generating the index by means of a computer as well as of accomplishing several other tasks; the results of this pilot effort have been previously reported.[5]

For the 1963 meeting, however, an important experimental innovation was conceived. As before, the authors of the papers selected one or more terms which they felt applied to their reports from a list compiled by specialists in the fields represented by the Federation members. As before, they were given the opportunity of adding any concepts which they felt were pertinent, but which did not appear on the standard list. Blanks were furnished for this purpose at the bottom of the form provided (see Figure 1). Processing of the authors' standard choices was accomplished by using the corresponding numerical codes as input and then converting them to their aphanumeric representation by ordinary table look-up methods, also in a manner similar to previous years.

Many minor departures from prior processing methods were incorporated. The most important difference, however, between the processing of the 1963 index and indexes of former years was in the manner of handling the natural language concepts provided by the authors themselves. It was in this area that the computer thesaurus was put to work; whereas previously these statements had been analyzed, ultimately cross-referenced and entered into the index by trained personnel, now they were processed by machine with the use of the thesaurus.



FORM B. PROGRAMING AND ARRANGEMENT OF 1963 ABSTRACTS

DO NOT WRITE IN THIS BOX

(This Form must be returned with Form A to your Society Officer, as indicated on Form A.)

1st author
Last Name First Name

Title of paper

Society to which submitted

The following is a list of session titles which have proved useful in the past. Please *encircle* one or two session titles under which your paper could fit. If no title listed is satisfactory, please suggest, in the space provided after "Other", one or two suitable session titles for your paper.

1. Active Transport	41. Antibodies	84. Viruses	123. Purine Metabolism
2. Electrolyte Excretion	42. Antibody Formation		124. Pyridine Nucleotides
3. Ion Transport	43. Antigen-Antibody	85. Biological Oxidation	
	Reaction	86. Biological Regulation	125. Enzymes
4. Adrenal Cortex	44. Antigenic Determinants	87. Electron Transport	126. Isozymes
5. Adrenal Medulla	45. Antigens	88. Oxidative	127. Proteolytic Enzymes
6. Anterior Pituitary	46. Autoimmunization	Phosphorylation	
7. Insulin	47. Blood Groups	89. Photosynthesis	128. Fats & Fatty Acids
8. Parathyroid	48. Complement	90. Propionate Metabolism	129. Lipids
9. Posterior Pituitary	49. Endotoxins		130. Phospolipids
10. Sex Hormones	50. Histamine-Antihistamine	91. Blood Gases	131. Sterol Metabolism
11. Steroid Hormones	51. Hypersensitivity	92. Pulmonary Circulation	
12. Thyroid	52. Immunopathology	93. Pulmonary Diffusion	132. Gastric Secretion
	53. Resistance, Non-Specific	94. Regulation of Respiration	133. Gastroenterology
13. Aging	54. Suppression of Immune	95. Respiratory Mechanics	134. Gastrointestinal
	Response	96. Tissue Respiration	Absorption
14. Alkaloids	55. Tissue Response		135. Gastrointestinal Motility
15. Biochemical	56. Transplantation	97. Body Composition	136. Gastrointestinal
Pharmacology		98. Body Water	Pharmacology
16. Catecholamines	57. Atherosclerosis	99. Obesity	137. Germ-Free Animal
17. Chemical Mediators	58. Blood Pressure		Nutrition
18. Chemotherapy	59. Blood Volume	100. Bone	138. Growth Factors
19. Drug Metabolism	60. Capillary Circulation		139. Microbial Nutrition
20. Toxicology	61. Cardiac Dynamics	101. Carbohydrates	140. Mineral Nutrition
	62. Cardiac Output	102. Glycolysis	141. Protein Nutrition
21. Altitude; Space	63. Cardiovascular	103. Glycoproteins	142. Ruminant Nutrition
22. Cold Adaptation	Pharmacology	104. Mucopolysaccharides	
23. Energy Metabolism	64. Coronary		143. Vitamin C
24. Hibernation	65. Peripheral Circulation	105. Cardiac Muscle	144. Vitamin Interrelation-
25. Hyperthermia; Heat	66. Shock	106. Smooth Muscle	ships
26. Hypothermia; Cold		107. Striated Muscle	145. Vitamins, B
27. Radiation	67. Autonomic		146. Vitamins, Fat-Soluble
28. Temperature Regulation	Pharmacology	108. Cell Structure & Function	
	68. Behavior	109. Cytology	147. Liver
29. Amino Acids	69. Brain Stem	110. L-Forms & Protoplasts	148. Lymphoid Organs
	70. Cerebral Cortex	111. Tissue Analysis	149. Pancreas
30. Collagen	71. Ganglia	112. Tissue Culture	
31. Connective Tissue	72. Motivation; Learning	113 Tissue Metabolism	150. Neoplasia
32. Plasma Proteins	73. Neurochemistry		
33. Protein Chemistry	74 Neuroendocrine	114. Clinical Chemistry	151. Renal Blood Flow
34. Protein Structure	75. Neuropathology	115. Clinical Nutrition	152. Renai Concentrating
	76 Neuropharmacology	116. Clinical Pharmacology	Mechanism
35. Analgesics	77. Peripheral Nerve		153. Renal Disease
36. Anesthetics	78. Psychopharmacology	117 Comparative Physiology	154 Renal Pharmacology
	79. Reflexes		155. Renal Tubule
37. Anemia	80. Sensory Effects	118. DNA Chemistry	
38. Blood Coagulation	81. Spinal Cord	119. Genetics	156. Reproduction
39. Hematology		120. Nucleic Acids	
40. Porphyrins &	82. Bacteriophage	121 Nucleotides	
Hemeproteins	83 Enteroviruses	122. Protein Synthesis	

Other

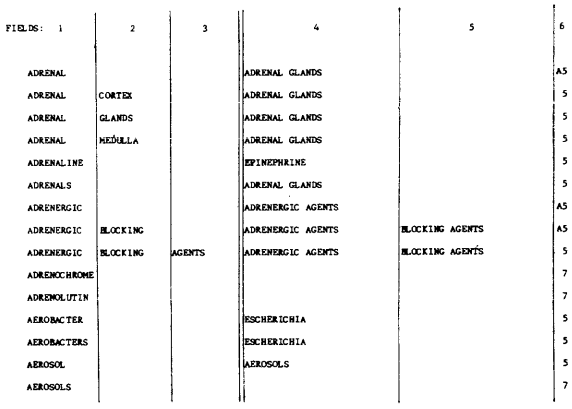
Figure 1. Form B, Provided for Author Response.

The Machine Thesaurus

As a purely experimental device, the thesaurus in its initial form was of necessity but an approximation. Its content was structured in terms of the collection of statements written in by authors in previous years. A projection of the subject matter was made, hoping that the field had not changed too radically, and recognizing that even the most accurate appraisal would fail to anticipate all areas of subject material.

Since the theoretical base of the experiment was the use of the thesaurus in concept processing, the computer was programmed to accept, wherever possible, the entire statements written in. By examination of previous years' forms, it was found that few statements were likely to exceed three valid words—that is, those remaining after the words comprising the statements were processed against a "throwaway" or delete list, and words not contributing to the conceptual meaning of the statements were eliminated. For this reason, the thesaurus was designed to contain a maximum of three valid words in the left-most, or "look-up," portion.

Further, each of the words comprising the look-up portion was limited by practical considerations to twelve characters, the length of one computer word on the UNIVAC® I Computer. This restriction caused no difficulty, since it was found in previous years that twelve characters were sufficient in every case to insure unique recognition. Thus, each of the words or combinations of words anticipated in the authors' statements were entered in the left three fields of the thesaurus (see Figure 2).



FIELDS: 1	2	3	4	5	6
ADRENAL			ADRENAL GLANDS		A5
ADRENAL	CORTEX		ADRENAL GLANDS		5
ADRENAL	GLANDS		ADRENAL GLANDS		5
ADRENAL	MEDULLA		ADRENAL GLANDS		5
ADRENALINE			EPINEPHRINE		5
ADRENALS			ADRENAL GLANDS		5
ADRENERGIC			ADRENERGIC AGENTS		A5
ADRENERGIC	BLOCKING		ADRENERGIC AGENTS	BLOCKING AGENTS	A5
ADRENERGIC	BLOCKING	AGENTS	ADRENERGIC AGENTS	BLOCKING AGENTS	5
ADRENOCHROME					7
ADRENOLUTIN					7
AEROBACTER			ESCHERICHIA		5
AEROBACTERS			ESCHERICHIA		5
AEROSOL			AEROSOLS		5
AEROSOLS					7

Figure 2. Sample Segment of Machine Thesaurus.

Field 4 contained, where applicable, the valid equivalent of the concept as it was to appear in the index, with Field 5 being allocated for an additional term.

Field 6 was coded according to the action the computer was to take upon encountering a match between the author's statement and the thesaurus entry. Code 5 indicated that the word or words of the statement were to be replaced by the contents of Field 4 and, if applicable, Field 5. In many cases this meant merely contracting the separate words of the statement into a readable phrase acceptable as an index entry. The third entry in Figure 2 is an example.

More importantly, this code was used to cause uniform indexing of various forms of a given concept, including cases in which the phrase used was on too specific a level to become an entry of the index. Thus, "adrenal cortex" and "adrenal medulla" would both be transformed into "adrenal glands." Code 7 indicated that the concept appearing in the look-up portion was in itself a valid indexing term, needing no modification. In addition, a further option was incorporated into the system which represented an extension of Code 7, although it was not utilized in the present experiment. Provision was made to recognize Code 6 as indicating that the term in the look-up portion is valid, but the document represented by that term should *also* be indexed under the terms appearing in Fields 4 and 5. Use of this provision would allow more complete indexing in a collection of larger size or where greater flexibility is desired.

Since each phrase was to be processed as a conceptual entity where possible, the "A," or "association," code in Field 6 indicated to the computer that a given term found in the thesaurus was immediately followed by more specific terms using the first word or the first and second words of the phrase in question. Since each concept processed against the thesaurus contained a maximum of three words, a match could possibly occur on all three words. If this was not the case, a match was made at the highest level possible, and the remainder of the phrase was retained for further processing. For example, if the concept "adrenergic blocking agents" were encountered, a match would first be made on "adrenergic," the association code would be found, a test and match would occur on the second word, and the association code found at the second level. A third level match would occur, and the appropriate action would be taken—the substitution with the two more precise concepts. In the case of "adrenal disorders," however, although a match would be encountered at the first level, none would be found at the second level. "Adrenal" would become "adrenal glands," and "disorders" would be retained for reprocessing, at which time it would be found in the thesaurus as an acceptable term. Thus, one of the most flexible features of the thesaurus was its ability to recognize and process the cases in which two concepts were in fact present in a single statement.

Certainly not every phrase to be written in by the authors could be anticipated. Provision was therefore made for "educating" the thesaurus by examining printer listings indicating cases in which a match was found at no level. The thesaurus was continually undergoing a process of amendment, new concepts being added as processing progressed, and the terms replaced as input to be accepted or modified. A flow diagram of the thesaurus process is shown in Figure 3*.

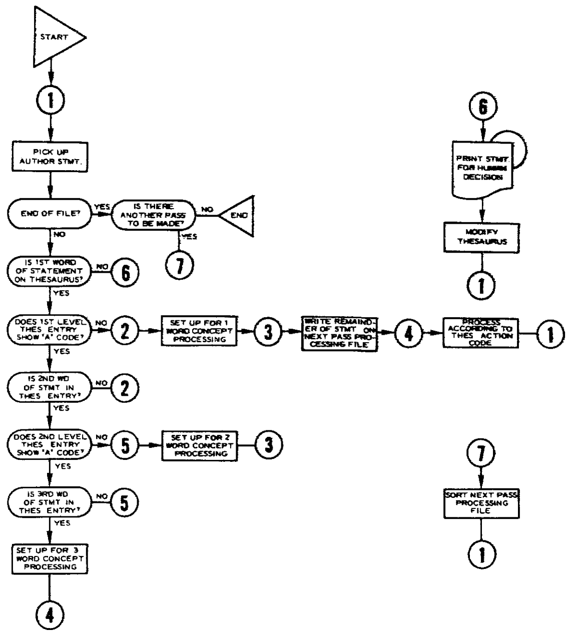

START
1
PICK UP AUTHOR STMT.
END OF FILE?
YES
IS THERE ANOTHER PASS TO BE MADE?
NO
END
YES
7
NO
IS 1ST WORD OF STATEMENT ON THESAURUS?
NO
6
YES
DOES 1ST LEVEL THES ENTRY SHOW 'A' CODE?
NO
2
SET UP FOR 1 WORD CONCEPT PROCESSING
3
WRITE REMAINDER OF STMT ON NEXT PASS PROCESSING FILE
4
PROCESS ACCORDING TO THES ACTION CODE
1
YES
IS 2ND WD OF STMT IN THES ENTRY?
NO
2
YES
DOES 2ND LEVEL THES ENTRY SHOW 'A' CODE?
NO
5
SET UP FOR 2 WORD CONCEPT PROCESSING
3
YES
IS 3RD WD OF STMT IN THES ENTRY?
NO
5
YES
SET UP FOR 3 WORD CONCEPT PROCESSING
4
6
PRINT STMT FOR HUMAN DECISION
MODIFY THESAURUS
1
7
SORT NEXT PASS PROCESSING FILE
1

Figure 3. Flow Diagram of Thesaurus Processor.

Processing for Index Production

The thesaurus phase, although the newest and most interesting of the processes, was only one of many necessary steps in the transformation of the punched input data into a complete, unified index. Figure 4 illustrates the various stages in the over-all process. One of the first tasks was to eliminate the nonsignificant words from the author statements. This was done by compiling a

* The author gratefully acknowledges the valuable participation of Mr. Paul Montgomery, of the UNIVAC Data Processing Center, New York, whose effort both in the areas of programming and supervision of processing was indispensable.

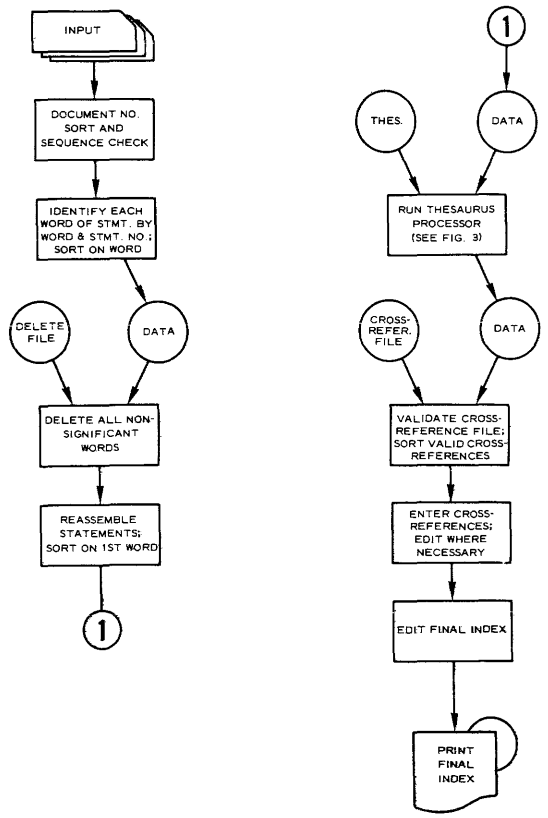

INPUT
DOCUMENT NO. SORT AND SEQUENCE CHECK
IDENTIFY EACH WORD OF STMT. BY WORD & STMT. NO.; SORT ON WORD
DELETE FILE
DATA
DELETE ALL NON-SIGNIFICANT WORDS
REASSEMBLE STATEMENTS; SORT ON 1ST WORD
1
THES.
DATA
RUN THESAURUS PROCESSOR (SEE FIG. 3)
CROSS-REFER. FILE
DATA
VALIDATE CROSS-REFERENCE FILE; SORT VALID CROSS-REFERENCES
ENTER CROSS-REFERENCES; EDIT WHERE NECESSARY
EDIT FINAL INDEX
PRINT FINAL INDEX

Figure 4. Flow Diagram of System.

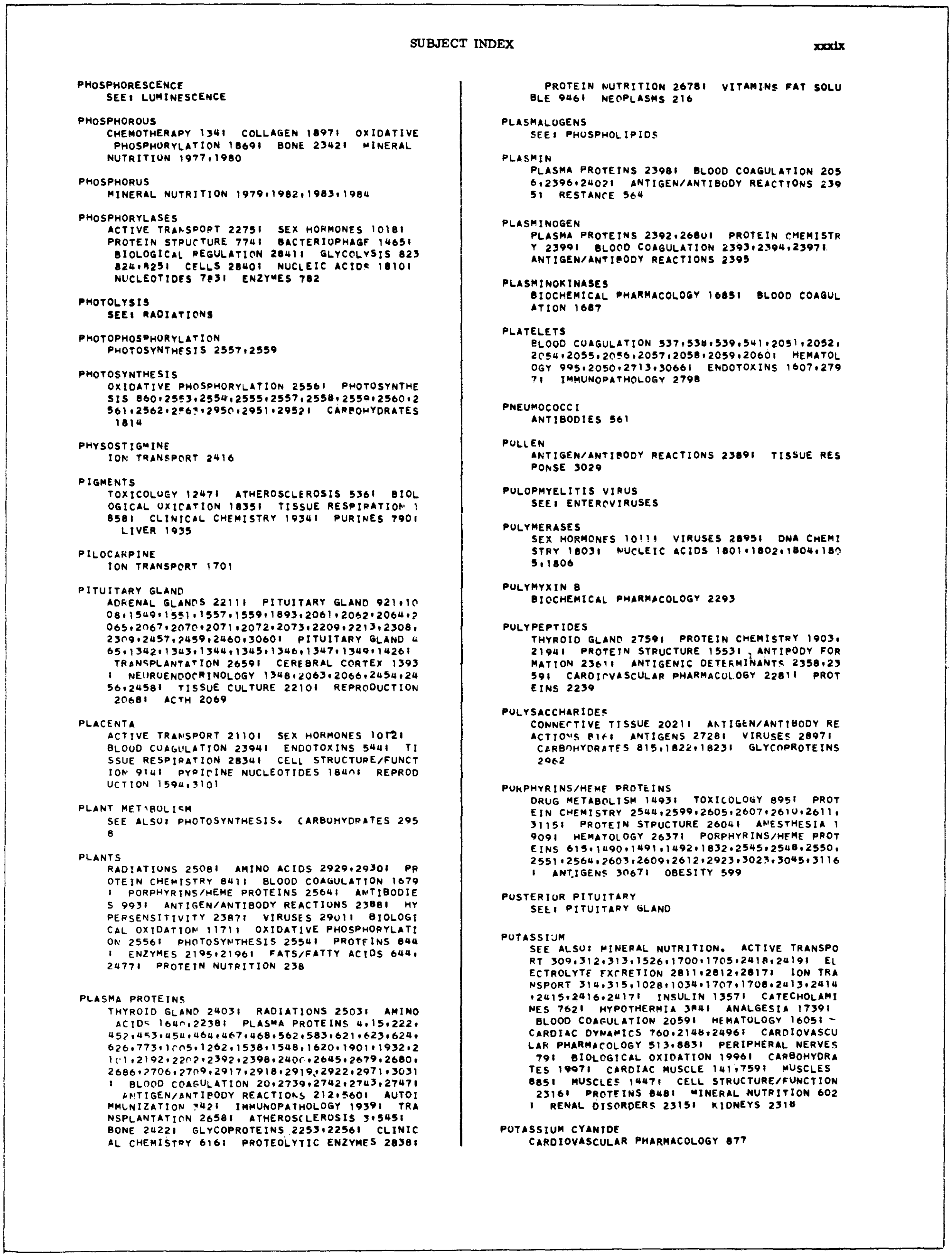
special table of such words, as complete as was possible based on prior years' information, against which each word of input was compared. Even so, a few were missed, but came to light as mismatches in the thesaurus phase and were eliminated.

Following the thesaurus phase, the index information, now completely expanded and conceptualized, was cross-referenced. Another special table had been devised for this purpose; each concept which might require cross-referencing (here, again, it was impossible to predict precisely which would and which would not be needed) was listed, together with a second concept to which the first would refer. Thus, two elements existed for each dummy entry: one, a reference "from"; the other, a reference "to." A typical entry in the table was "vasodilators—(see) (see also) cardiovascular pharmacology." The first task was to determine, in the case of each entry, whether any documents were indexed under the "to," or right-hand term; if not, a cross-reference to that term in the index would naturally be useless. Otherwise, the dummy entry was accepted as applicable and was retained. The second task was to insert the dummy entries into the list of index information and, in doing so, to determine whether the particular cross-reference should read "see" or "see also." Such a choice was dictated by the presence or absence of papers indexed under the "from," or left-hand term. Thus, in the above example, if first it were determined that references were in fact listed under "cardiovascular pharmacology," we must learn if entries exist under "vasodilators." If this is the case, "see also" is to be inserted in the cross-reference, since the word "see" alone would imply that no documents are to be found dealing with vasodilators.

Finally, the completely expanded and cross-referenced index was edited and printed out for photo-offset reproduction (although plans had been tentatively made for production of paper tape coded to drive Photon printers). A sample of the completed index is shown in Figure 5. Examples of cross-references may be noted. Each term under which one or more papers were indexed appears in alphabetic order, with the designations for all pertinent papers listed thereunder. A specific paper might be listed as "Radiations 2508." The number refers to the number of the article—abstracts are arranged serially in the *Proceedings*. The term preceding the number informs the user of the main subject of the paper, so that he may judge the context within which his subject of interest is discussed.

SUMMARY

An experiment in lightening the indexing problem in document processing has been discussed. The use of the mechanized thesaurus appears to be promising, especially combined with preliminary indexing operations such as was accomplished by the authors themselves. It offers the distinct advantage of enabling the user of the index to retrieve in terms of a complete conceptual representation of the document collection, without the burden of



SUBJECT INDEX xxxix

```
PHOSPHORESCENCE
    SEE; LUMINESCENCE

PHOSPHOROUS
    CHEMOTHERAPY 134;  COLLAGEN 1897;  OXIDATIVE
     PHOSPHORYLATION 1869;  BONE 2342;  MINERAL
    NUTRITION 1977,1980

PHOSPHORUS
    MINERAL NUTRITION 1979,1982,1983,1984

PHOSPHORYLASES
    ACTIVE TRANSPORT 2275;  SEX HORMONES 1018;
    PROTEIN STRUCTURE 774;  BACTERIOPHAGF 1465;
     BIOLOGICAL REGULATION 2841;  GLYCOLYSIS 823
     824,825;  CELLS 2840;  NUCLEIC ACIDS 1810;
     NUCLEOTIDES 783;  ENZYMES 782

PHOTOLYSIS
    SEE; RADIATIONS

PHOTOPHOSPHORYLATION
    PHOTOSYNTHESIS 2557,2559

PHOTOSYNTHESIS
    OXIDATIVE PHOSPHORYLATION 2556;  PHOTOSYNTHE
    SIS 860,2553,2554,2555,2557,2558,2559,2560,2
    561,2562,2563,2950,2951,2952;  CARBOHYDRATES
     1814

PHYSOSTIGMINE
    ION TRANSPORT 2416

PIGMENTS
    TOXICOLOGY 1247;  ATHEROSCLEROSIS 536;  BIOL
    OGICAL OXICATION 1835;  TISSUE RESPIRATION 1
    858;  CLINICAL CHEMISTRY 1934;  PURINES 790;
      LIVER 1935

PILOCARPINE
    ION TRANSPORT 1701

PITUITARY GLAND
    ADRENAL GLANDS 2211;  PITUITARY GLAND 921,10
    08,1549,1551,1557,1559,1893,2061,2062,2064,2
    065,2067,2070,2071,2072,2073,2209,2213,2308,
    2309,2457,2459,2460,3060;  PITUITARY GLAND 4
    65,1342,1343,1344,1345,1346,1347,1349,1426;
     TRANSPLANTATION 2659;  CEREBRAL CORTEX 1393
    ;  NEURUENDOCRINOLOGY 1348,2063,2066,2454,24
    56,2458;  TISSUE CULTURE 2210;  REPRODUCTION
     2068;  ACTH 2069

PLACENTA
    ACTIVE TRANSPORT 2110;  SEX HORMONES 1012;
    BLOOD COAGULATION 2394;  ENDOTOXINS 544;  TI
    SSUE RESPIRATION 2834;  CELL STRUCTURE/FUNCT
    ION 914;  PYRIDINE NUCLEOTIDES 1840;  REPROD
    UCTION 1594,3101

PLANT METABOLISM
    SEE ALSO; PHOTOSYNTHESIS.  CARBOHYDRATES 295
    8

PLANTS
    RADIATIONS 2508;  AMINO ACIDS 2929,2930;  PR
    OTEIN CHEMISTRY 841;  BLOOD COAGULATION 1679
    ;  PORPHYRINS/HEME PROTEINS 2564;  ANTIBODIE
    S 993;  ANTIGEN/ANTIBODY REACTIONS 2388;  HY
    PERSENSITIVITY 2387;  VIRUSES 2901;  BIOLOGI
    CAL OXIDATION 1171;  OXIDATIVE PHOSPHORYLATI
    ON 2556;  PHOTOSYNTHESIS 2554;  PROTEINS 844
    ;  ENZYMES 2195,2196;  FATS/FATTY ACIDS 644,
    2477;  PROTEIN NUTRITION 238

PLASMA PROTEINS
    THYROID GLAND 2403;  RADIATIONS 2503;  AMINO
     ACIDS 1640,2238;  PLASMA PROTEINS 4,15,222,
    452,453,454,464,467,468,562,583,621,623,624,
    626,773,1005,1262,1538,1548,1620,1901,1932,2
    101,2192,2202,2392,2398,2400,2645,2679,2680,
    2686,2706,2709,2917,2918,2919,2922,2971,3031
    ;  BLOOD COAGULATION 20,2739,2742,2743,2747;
      ANTIGEN/ANTIBODY REACTIONS 212,560;  AUTOI
    MMUNIZATION 342;  IMMUNOPATHOLOGY 1939;  TRA
    NSPLANTATION 2658;  ATHEROSCLEROSIS 3,545;
    BONE 2422;  GLYCOPROTEINS 2253,2256;  CLINIC
    AL CHEMISTRY 616;  PROTEOLYTIC ENZYMES 2838;
      PROTEIN NUTRITION 2678;  VITAMINS FAT SOLU
    BLE 946;  NEOPLASMS 216

PLASMALOGENS
    SEE; PHOSPHOLIPIDS

PLASMIN
    PLASMA PROTEINS 2398;  BLOOD COAGULATION 205
    6,2396,2402;  ANTIGEN/ANTIBODY REACTIONS 239
    5;  RESTANCE 564

PLASMINOGEN
    PLASMA PROTEINS 2392,2680;  PROTEIN CHEMISTR
    Y 2399;  BLOOD COAGULATION 2393,2394,2397;
    ANTIGEN/ANTIBODY REACTIONS 2395

PLASMINOKINASES
    BIOCHEMICAL PHARMACOLOGY 1685;  BLOOD COAGUL
    ATION 1687

PLATELETS
    BLOOD COAGULATION 537,538,539,541,2051,2052,
    2054,2055,2056,2057,2058,2059,2060;  HEMATOL
    OGY 995,2050,2713,3066;  ENDOTOXINS 1607,279
    7;  IMMUNOPATHOLOGY 2798

PNEUMOCOCCI
    ANTIBODIES 561

POLLEN
    ANTIGEN/ANTIBODY REACTIONS 2389;  TISSUE RES
    PONSE 3029

PULOPMYELITIS VIRUS
    SEE; ENTEROVIRUSES

POLYMERASES
    SEX HORMONES 1011;  VIRUSES 2895;  DNA CHEMI
    STRY 1803;  NUCLEIC ACIDS 1801,1802,1804,180
    5,1806

POLYMYXIN B
    BIOCHEMICAL PHARMACOLOGY 2293

POLYPEPTIDES
    THYROID GLAND 2759;  PROTEIN CHEMISTRY 1903,
    2194;  PROTEIN STRUCTURE 1553;  ANTIBODY FOR
    MATION 2361;  ANTIGENIC DETERMINANTS 2358,23
    59;  CARDIOVASCULAR PHARMACOLOGY 2281;  PROT
    EINS 2239

POLYSACCHARIDES
    CONNECTIVE TISSUE 2021;  ANTIGEN/ANTIBODY RE
    ACTIONS 816;  ANTIGENS 2728;  VIRUSES 2897;
     CARBOHYDRATES 815,1822,1823;  GLYCOPROTEINS
     2962

PORPHYRINS/HEME PROTEINS
    DRUG METABOLISM 1493;  TOXICOLOGY 895;  PROT
    EIN CHEMISTRY 2544,2599,2605,2607,2610,2611,
    3115;  PROTEIN STRUCTURE 2604;  ANESTHESIA 1
    909;  HEMATOLOGY 2637;  PORPHYRINS/HEME PROT
    EINS 615,1490,1491,1492,1832,2545,2548,2550,
    2551,2564,2603,2609,2612,2923,3023,3045,3116
    ;  ANTIGENS 3067;  OBESITY 599

POSTERIOR PITUITARY
    SEE; PITUITARY GLAND

POTASSIUM
    SEE ALSO; MINERAL NUTRITION.  ACTIVE TRANSPO
    RT 309,312,313,1526,1700,1705,2418,2419;  EL
    ECTROLYTE EXCRETION 2811,2812,2817;  ION TRA
    NSPORT 314,315,1028,1034,1707,1708,2413,2414
    ,2415,2416,2417;  INSULIN 1357;  CATECHOLAMI
    NES 762;  HYPOTHERMIA 384;  ANALGESIA 1739;
     BLOOD COAGULATION 2059;  HEMATOLOGY 1605;
    CARDIAC DYNAMICS 760,2148,2496;  CARDIOVASCU
    LAR PHARMACOLOGY 513,883;  PERIPHERAL NERVES
     79;  BIOLOGICAL OXIDATION 1996;  CARBOHYDRA
    TES 1997;  CARDIAC MUSCLE 141,759;  MUSCLES
    885;  MUSCLES 1447;  CELL STRUCTURE/FUNCTION
     2316;  PROTEINS 848;  MINERAL NUTRITION 602
    ;  RENAL DISORDERS 2315;  KIDNEYS 2318

POTASSIUM CYANIDE
    CARDIOVASCULAR PHARMACOLOGY 877
```

Figure 5. Sample Page from Final Index.

performing deep indexing by highly trained personnel.

The need for subject specialists, however, has not been entirely obviated. Luhn[6] correctly states that the preparation of any special dictionary or thesaurus demands complete familiarity with the field in question. This was certainly found to be true; construction of the thesaurus used in the present experiment depended not only on study of previous years' materials, but on their being completely understood. Once again, the essence of successfully representing the subject matter was the *concept*, which, when incorporated into the thesaurus, enabled it to fulfill its purpose.

The study, although oriented toward the production of a printed index, has demonstrated a method by which several types of retrieval files may be generated. Certainly, preparation of an index to be searched not by humans but by the computer could be accomplished by such a process. The ability to provide standardized indexing terms automatically, together with the advantage of complete cross-referencing, would result in a high degree of retrieval efficiency.

BIBLIOGRAPHY

1. BAXENDALE, PHYLLIS B. "An Empirical Model for Machine Indexing." *Machine Indexing: Progress and Problems,* Center for Technology and Administration, The American University, 1961, pp. 207-218.
2. SALTON, GERARD. "Some Experiments in the Generation of Word and Document Associations." *AFIPS Conference Proceedings,* Vol. 22, 1962, pp. 234-250.
3. MARON, M. E. "Automatic Indexing: An Experimental Inquiry," *Journal of the Association for Computing Machinery,* Vol. 8, No. 3, 1961, pp. 404-417.
4. WEINBERG, ALVIN M. "Scientific Communication." *International Science and Technology,* No. 16, April, 1963, pp. 65-74.
5. SCHULTZ, C. K., and SHEPHERD, CLAYTON A. "The 1960 Federation Meeting: Scheduling a Meeting and Preparing an Index by Computer." *Federation Proceedings,* Vol. 19, No. 2, 1960, pp. 682-699.
6. LUHN, H. P. "Potentialities of Auto-Encoding of Scientific Literature." IBM Research Report RC-101, 1959.

SYNTACTIC STRUCTURE AND AMBIGUITY OF ENGLISH*

Susumu Kuno and Anthony G. Oettinger
Computation Laboratory of Harvard University
Cambridge, Massachusetts

1. INTRODUCTION

This paper is in two parts. The first (Section 2) gives an evaluation of the performance of the multiple-path syntactic analyzer to date, with emphasis on the nature and the consequences of syntactic ambiguities in English sentences and suggestions for the refinement of the grammar. The remainder of the paper is concerned with certain concrete implications of the theoretical description of multiple-path predictive analysis provided by recent work of Evey[4, 5] and Greibach[6, 8]. A modification of the form of the current grammar is proposed which should yield a new grammar with additional intuitive appeal, a simplified version of the present analysis program, and sentence structure descriptions in the form of a generalized parenthesis-free notation readily interpretable as a tree.

The basic technique of multiple-path predictive analysis has been described previously (Kuno and Oettinger[12, 13]). The grammar and other details of the operating system are given in full in two recent reports (Kuno[10, 11]).

The grammar is essentially a set of directed productions as defined by Greibach[6, 8]. A directed production is written as (P, c) → c $P_1 \ldots P_k$ where c is a terminal symbol (syntactic word class) and the P's are intermediate symbols (predictions). Each prediction stands for a syntactic structure ascribed by the grammar to a string of the language, such as "S" (sentence), "VP" (predicate), "SP" (subject phrase), "PD" (period), etc. A syntactic role indicator is adjoined to each production to describe the role played by the word class c when fulfilling the prediction P. For example, (S, prn) → prn VP PD, (*SV*) indicates that a sentence may be initiated ("S" is an initial symbol) by a prn (personal pronoun in the nominative case) serving as subject of a predicate verb (*SV*), and that the pronoun should be followed by a predicate ("VP") and a period ("PD").

For any given English sentence the analyzer, now in operation on Harvard's IBM 7090, produces explicitly all parsings of the sentence implicit in the current version of the grammar, which has been designed to accept as well-formed most sentences that appear or may appear in scientific papers.

The analyzer, based on a predictive technique originally proposed by Rhodes[17], is abstractly characterized as a directed production analyzer or dpa (Greibach[6]). Every dpa is the inverse of a context-free phase structure generator (psg) in a standard form with productions P → c $P_1 \ldots P_k$. It is an inverse in the sense that the dpa will accept as well-formed precisely those strings generated by the psg. Since Greibach has shown that for every psg (in the sense of Chomsky) there is a psg in standard form which generates precisely the same set of strings, every psg has a dpa as an inverse, and the intuitively evolved multiple-path predictive analyzer therefore turns out to have even

* This work has been supported in part by the National Science Foundation under Grant G-24833.

greater generality and esthetic appeal than was originally hoped for.

The mechanism of analysis may be characterized as a non-deterministic pushdown store transducer. According to results of Chomsky[2] and Evey[4], the set of all languages that can be either accepted or generated by this class of machines is precisely the set of all context-free phrase structure languages. Earlier conjectures of Oettinger[15] regarding the role of pushdown stores in syntactic analysis are thus confirmed and, although other mechanisms have been suggested (Matthews[14], Sakai[19]) or implemented (Robinson[18]) there is now good reason for regarding the pushdown store transducer as a "natural" device and not merely as a convenient programming trick.

Conceptually, the analyzer operates as follows. The topmost prediction P (intermediate symbol) in a prediction pool (pushdown store) is used to form a couple (P, c) with the word class c of the word being scanned. If there is no production in the grammar with couple (P, c), the pool is abandoned. Otherwise, the symbol "P" is deleted from the pool, as many copies of the pool are made as there are productions with couple (P, c), the elements P_1 . . . P_k of each production are loaded into the corresponding pool, the process moves to the next word and continues with each of the new pools in turn. The process is initiated with a single pool containing only the initial symbol "S"; it yields an acceptable structure for a sentence whenever a period (or equivalent) is reached and the pool is empty after removal of the prediction of the period; it terminates when all pools have been abandoned or have led to acceptable structures. Since a given word may belong to more than one syntactic word class, means for cycling through the possible word class combinations must be superimposed on this basic non-deterministic pushdown store machine, but this adds no essential features or complications.

Each distinct sentence structure is displayed both as a list of couples (P, c) consistent with the characterization of the system as a directed production analyzer and in a more conventional tree form related to its characterization as the inverse of a phrase structure generator.

2. THE OUTPUT OF THE ANALYZER

2.1 The application of the analyzer to English text has, on the whole, yielded results that are encouraging in the sense that intuitively satisfactory and semantically acceptable structures are produced for a wide range of sentences. Where a sentence is commonly regarded as inherently ambiguous (e.g., "They are flying planes."), the analyzer produces several structures each reflecting one of the distinct interpretations.

There has been, to date, no difficulty in extending the grammar to yield acceptable analyses for sentences rejected by earlier versions, and no major difficulties are anticipated on this score in the future. Catastrophic increase in the size of the grammar seems unlikely; in fact, the current grammar of 2100 rules is descended from an earlier version with 3500 rules with some increase in power on the way.

To be sure, certain common "idiomatic" structures are still maltreated owing to the absence of idiom tables. These have been deliberately omitted to resist the temptation toward excessive *ad hoc* use of such tables to handle apparently difficult constructions that, after some thought, turn out to be amenable to clear-cut systematic treatment within the frame-work of a dpa. Certain rare types of linked structures (e.g., such strings as abcd . . . abcd . . .) known to be beyond the scope of context-free phrase structure grammars must eventually be accounted for either by introducing the equivalent of less restrictive productions (thereby significantly deviating from pushdown store techniques) or by some *ad hoc* truncating technique (thereby sacrificing some conceptual elegance for the sake of a sound engineering solution). These and other sins of omission are not, however, of prime concern to us today.

The most serious problem for the immediate future is the matter of ambiguity. A sentence is ambiguous relative to a given dpa (psg) if that sentence is analyzed (generated) by the dpa (psg) in more than one way. Dealing with ambiguity is hard for both formal and psychological reasons.

Formally, there is a class of unpleasant theoretical results that tell us that the ambiguity problem is recursively unsolvable for context-free languages even of greatly restricted generality (Chomsky and Schützenberger[3], Greibach[7]), i.e., no general algorithm can be found for determining whether or not a given dpa (psg) will analyze (generate) some sentence in more than one way. The outlook for practically interesting decidable subsets is dim, and so experimental search for special solutions in special cases is our only recourse.

In a grammar that purports to describe a natural language, the question is not so much the existence of ambiguity but, worse yet, matching the ambiguity of the grammar to that observed in the language. From this point of view, there are three types of ambiguities: those that should be in the grammar because they are seen in the language, those that should not but are readily eliminated, and the rest. Obviously, the first two types cause no trouble. The elimination of the second type usually corresponds to an enlargement of the precincts of syntax at the expense of what otherwise would be regarded as semantics.

It is, however, a major problem to classify an ambiguity. Is it there because the grammar is at fault? Or are we unhappy with it merely because our mind is fixed on one plausible interpretation to the exclusion of others? At this stage one's disciplined inclination is to answer yes to the first question. Consider, however, the following sentence: "People who apply for marriage licenses wearing shorts or pedal pushers will be denied licenses."† Silly but clear, isn't it? But have you thought that "*People* who apply . . . *or pedal pushers* . . ." could be denied licenses? Dope pushers would be! Or perhaps it is "*People* who *apply* for . . . *or* (who) *pedal* pushers . . ." ? People do pedal bicycles. Are they wearing shorts, or are they applying for shorts that happen to be wearing marriage licenses? Will they *be denied* licenses? Or will they be *denied licenses*? There are more which the current grammar relentlessly exhibits.

Less frivolous cases will now be considered. Space permits only a sampling of both good and bad. Details may be found in Kuno[11] or run your own; grammar, dictionary and program are available to responsible investigators.

2.2 The first example to be considered will be a clear-cut one. It will serve primarily to illustrate various features of the analyzer and its output and to demonstrate that there are well-behaved English sentences that are properly treated by the analyzer. Two additional examples will then be used to exhibit ambiguities of the second and third type.

Figure 1 is a fragment of the grammar table. The argument pairs are couples (P, c). The new predictions (NEW PREDS) are right-hand sides $P_1 \ldots P_k$ of directed productions (P, c) $\rightarrow$ c $P_1 \ldots P_k$. Thus the rule entry of 7X, MMM-3 corresponds to a directed production (7X, mmm) $\rightarrow$ mmm XD MC. As mentioned earlier, the syntactic role indicator (SR) partly specifies the role c plays when fulfilling the prediction P. The role is completely specified by the syntactic role indicator in conjunction with indices (e.g., "A" of "XD-A" in 7X, MMM-3) associated with predictions. The agreement test indicator (AGREE TEST) introduces an apparent deviation from a strict pushdown transducer, but Greibach (Section 2.3)[8] has shown that it functions purely as an abbreviation technique without altering the fundamental nature of the grammar and analyzer. The structural and shift codes (STRUCT, SHIFT CD) are used by an editing program to turn the output of the dpa into a tree representation.

Definitions of a few of the 133 word classes (terminal symbols) presently used in the grammar are given in Fig. 2. A list of all 82 current predictions (intermediate symbols) is given in Fig. 3.

Sentence 1 is "The increase in flow stress was attributed to vacancies, which have appreciable mobility at — 72". Figure 4 shows the word class codes associated by the English dictionary with each word in this sentence. "S", "P", "C" and "Y" as the fourth character denote singular, plural, common, and subjunctive, respectively.

The unique analysis produced for this sentence is shown in Fig. 5. In any analysis, a single word class (SWC) together with a mne-

† For this and several other valuable test sentences we are indebted to Professor F. W. Harwood of the University of Tasmania who challenged our ability to deal with them.

ARGUMENT PAIR	SR	AGREE TEST	NEW PREDS	MNEMONIC DESCRIPTIONS OF PREDICTIONS	STRUCT, SHIFT CD	ENGLISH EXAMPLES
					***	LANGUAGE
7X,GT1-0	YY	00100		SUBJECT MASTER	SA	PROCESSING
			7X-X	SUBJECT MASTER	0 S	MECHANISMS
						WILL BE NEEDED
- - - -	- -	- - -	- - -	- - - - - - - - - - - -	- - - -	
7X,MMM-0	YY	10010		SUBJECT MASTER	S	TRANSLATION
						WILL BE NEEDED
7X,MMM-1	YY	10010		SUBJECT MASTER	S	TRANSLATION
			AP-	POST-POSITIONAL ADJ	1 SPM	PERFORMED
						(AUTOMATICALLY)
						WILL BE NEEDED
7X,MMM-2	YY	10010		SUBJECT MASTER	S	TRANSLATION
			AC-	ADJECTIVE CLAUSE	1 S7S	WHICH
					(S7V)	IS PERFORMED
						(AUTOMATICALLY)
						WILL BE NEEDED
7X,MMM-3	YY	00001		SUBJECT MASTER	S	ANALYSIS
			XC-A	(A) AND (B)	0 +	AND
			MC-X	NOUN SUBJECT	0 S	SYNTHESIS
						WILL BE NEEDED
7X,MMM-4	YY	00001		SUBJECT MASTER	S	ANALYZERS
			CN-A	COMMA	0 ,	,
			MC-X	NOUN SUBJECT	0 S	TRANSFORMERS
			XC-A	(A,B,) AND (C)	0 +	AND
			MC-X	NOUN SUBJECT	0 S	SYNTHESIZERS
						WILL BE NEEDED
7X,MMM-5	YY	10010		SUBJECT MASTER	S	ANALYZERS
			CN-A	COMMA	0 ,	,
						(AUTOMATIC)
			1C-X	SUBJECT	0 S	ANALYZERS
			CN-A	COMMA	0 ,	,
						WILL BE NEEDED
- - - -	- -	- - -	- - -	- - - - - - - - - - - -	- - - -	
7X,NOU-0	YY	00100		SUBJECT MASTER	SA	TRANSLATION
			7X-X	SUBJECT MASTER	0 S	PROGRAM
						WILL BE NEEDED
- - - -	- -	- - -	- - -	- - - - - - - - - - - -	- - - -	
					***	SPACE
7X,NUM-0	YY	00100		SUBJECT MASTER	SA	COMMUNICATIONS*
					(SA)	GREAT
			4X-X	MODIFIED SUBJECT	0 S	DIFFICULTIES
						ARE TO BE
						CONSIDERED

Figure 1. Fragment of Grammar Table.

AAA	common features of ADJ, ADK, ADM, ADN, ADO, and ART
AAB	common features of ADK and ADN
ADJ	(a) adjectives which can be modified by "very" such as "beautiful", "red", etc. "Many, much, few, little" are excluded from this class. (b) adjectives in the superlative degree, excluding "most" and "least". Ex. "prettiest, best, worst".
ADK	adjectives in the comparative degree: "older, better".
ADL	"very, only, same" as adjectives.
ADM	"most" and "least".
ADN	"more" and "less".
ADO	"many, much, few, little".
ADP	"such".
ART	ART is for noun-phrase introducers such as definite and indefinite articles ("the, a"), demonstrative adjectives ("this, that, these, those"), possessive pronouns ("my, your"), pro-adjectives ("another, any") and titles ("Dr.").
AUX	auxiliary verbs: "will, shall, can, may, do, does, etc.".
AV1	usual adverbs: "quickly, fast".
AV2	adverbs homographic with prepositions: "He came in.".
BE1	finite forms of "be" as a complete intransitive verb: "He is well.", "He moved that the problem be up for discussion.".
BE2	finite forms of "be" as a copula which has to be followed by a noun complement or by an adjective complement: "He is tall.", "I am a student.".
BE3	finite forms of "be" as an auxiliary verb for the progressive form, passive voice, or be-to-form: "He is running.", "He was killed.", "He is to come here.".
BI1	the basic form of BE1: "be".
BI2	the basic form of BE2: "be".
BI3	the basic form of BE3: "be".
CCO	non-subjunctive adverbial clause introducers: "before, after, since".
CMA	comma, semicolon, colon, dash, and parenthesis.
CO1	noun clause introducing conjunctions "that", "if" and "whether".
II1	the basic form of VI1.
II3	the basic form of VI3.
IT1	the basic form of VT1.
IT2	the basic form of VT2.
IT6	the basic form of VT6.
IT7	the basic form of VT7.

Figure 2. Fragment of Class Definitions.

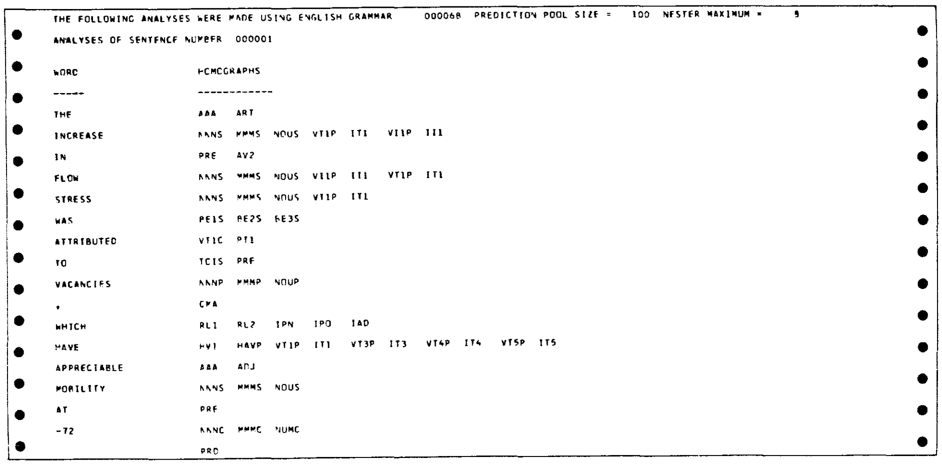
Prediction	Mnemonic Description	Prediction	Mnemonic Description
1X	SUBJECT	IO	INTERROG PRN ACC
33	AS-CLAUSE	IQ	INTERROG PRN COMPL
4X	MODIFIED SUBJECT	IX	COMPLETE VI
7X	SUBJECT MASTER	LB	RELATIVE PRONOUN ACC
88	THAN-CLAUSE	MX	NOUN SUBJECT
A1	ATTRIBUTIVE ADJ	N2	OBJECT
A2	DISCONTINUOUS ADJ	N3	NOUN COMPLEMENT
AC	ADJECTIVE CLAUSE	N5	MODIFIED OBJECT
AI	ADJECTIVE	N6	MODIFIED COMPLEMENT
AP	POST-POSITIONAL ADJ	N8	OBJECT MASTER
AR	ARTICLE	N9	COMPLEMENT MASTER
B1	INFINITE VT1	NC	NOUN CLAUSE
BV	INFINITE VERB	ND	NOUN CL WITH NO OBJ
BW	INF VERB WITH NO OBJ	NE	CONDITIONAL NOUN CLAUSE
BX	INF COMPLETE VI	NQ	NOUN OBJECT
BY	INFINITE COPULA	PA	PARTICIPLE
C2	ADVERB CLAUSE CONJ	PB	PART WITH NO OBJ
C3	AS (OF COMPARISON)	PD	PERIOD
C8	THAN (OF COMPARISON)	PF	PERFECT PARTICIPLE
CM	COMMA, AND, OR	PG	PERF PART WITH NO OBJ
CN	COMMA	PH	PERF PARTICIPLE VI
CX	COPULA	PI	PERF PART COPULA
DA	ADVERB	PJ	PERF PART BE1
DB	ADVERB AFTER BE1	Q1	PERF PARTICIPLE VT1
DC	THERE, HERE	QU	QUESTION MARK
DM	DUMMY PREDICTION	R1	PARTICIPLE VT1
DN	ADVERBIAL NOUN PHR	RR	PARTICIPLE VI
DP	PREPOSITIONAL PHR	RS	PRES PART COPULA
DQ	PREPOSITION	SE	SENTENCE
EX	BE2 (COPULA)	SF	DECLAR CL WITH NO OBJ
FX	BE3 (AUXILIARY)	SG	DECLARATIVE CLAUSE
G1	GERUND OF VT1	SH	CONDITIONAL DECLAR CL
GR	GERUND	TX	SIMPLE OBJ VT
HX	HAV3 (TENSE AUX)	UX	AUXILIARY VERB
I1	TO-INFIN VT1	VX	PREDICATE
ID	INTERROG ADVERB	WX	PREDICATE WITH NO OBJ
IF	TO-INFINITIVE	XC	(A,B,) AND (C)
IG	TO-INFIN WITH NO OBJ	XD	(A) AND (B)
IH	TO-INFIN COMPLETE VI	ZC	(A,B,) AND (C) (DROP)
II	TO-INFIN COPULA	ZD	(A) AND (B) (DROP)
IN	INTERROG PRN SUBJECT	ZM	COMMA, AND, OR (DROP)

Figure 3. List of Predictions.

monic interpretation (SWC CODE) is selected among those originally given as in Fig. 4. Classes mmm or nnn and aaa or aab account for features common to several noun and adjective classes respectively, and have been introduced explicitly to achieve certain practical economies. However, only the parent class appears in this column of the analysis output. For example, although the rule (4X, mmm) accounts for "increase" as "nou", it is "nou" which appears as SWC in Fig. 5.

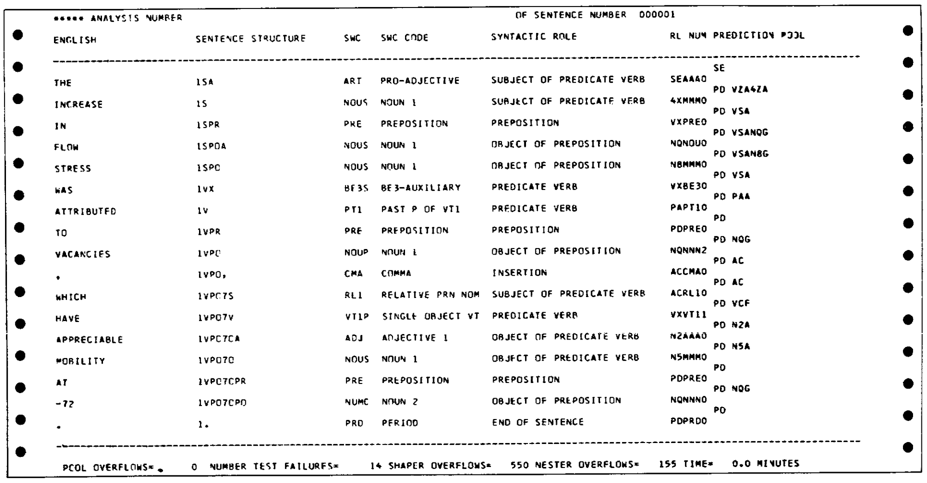
The data in the "SYNTACTIC ROLE" column of Fig. 5 give a rough idea of the role of each word in the sentence. The syntactically and semantically acceptable sentence structure produced by the analyzer is exhibited in more explicit detail by the tree in Fig. 6. This tree is based in an obvious way on the data in the "SENTENCE STRUCTURE" column of Fig. 5; the latter format, which is easier to lay out on a standard printer than a tree, is produced by an editing program from the dpa output which will be described shortly. The structure symbols used in both representations are defined in Fig. 7. The tree representation, which is both intuitively appealing and useful in certain applications, has features of both phrase structure and dependency trees (Hays[9]); its nature is examined more closely by Greibach (Section 3)[8], and in Section 3 of this paper.

The heart of the output, corresponding to the output of a dpa, is given in the columns "RL NUM" (Rule Number) and "PREDICTION POOL" of Fig. 5. Before the processing of "flow", the pushdown store holds | PD- | VS-A | NQ-G |. The couple (NQ, nou) specifies the rule that accepted "flow" as nou used attributively. The right-hand element of the corresponding production (NQ, nou) →



```
THE FOLLOWING ANALYSES WERE MADE USING ENGLISH GRAMMAR     000068  PREDICTION POOL SIZE =   100  NESTER MAXIMUM =    5
ANALYSES OF SENTENCE NUMBER  000001

WORD                  HOMOGRAPHS
-----                 ------------
THE                   AAA   ART
INCREASE              NNNS  MMMS  NOUS  VT1P  IT1   VI1P  II1
IN                    PRE   AV2
FLOW                  NNNS  MMMS  NOUS  VI1P  II1   VT1P  IT1
STRESS                NNNS  MMMS  NOUS  VT1P  IT1
WAS                   BE1S  BE2S  BE3S
ATTRIBUTED            VT1C  PT1
TO                    TOIS  PRF
VACANCIES             NNNP  MMMP  NOUP
,                     CMA
WHICH                 RL1   RL2   IPN   IPO   IAD
HAVE                  HV1   HAVP  VT1P  IT1   VT3P  IT3   VT4P  IT4   VT5P  IT5
APPRECIABLE           AAA   ADJ
MOBILITY              NNNS  MMMS  NOUS
AT                    PRE
-72                   NNNC  MMMC  NUMC
                      PRD
```

Figure 4. Coding of Sentence 1.



```
***** ANALYSIS NUMBER                                  OF SENTENCE NUMBER  000001
ENGLISH         SENTENCE STRUCTURE   SWC   SWC CODE           SYNTACTIC ROLE              RL NUM PREDICTION POOL
-----------------------------------------------------------------------------------------------------------
                                                                                                 SE
THE             1SA                  ART   PRO-ADJECTIVE      SUBJECT OF PREDICATE VERB   SEAAA0
                                                                                                 PD VZA4ZA
INCREASE        1S                   NOUS  NOUN 1             SUBJECT OF PREDICATE VERB   4XMMM0
                                                                                                 PD VSA
IN              1SPR                 PRE   PREPOSITION        PREPOSITION                 VXPRE0
                                                                                                 PD VSANQG
FLOW            1SPOA                NOUS  NOUN 1             OBJECT OF PREPOSITION       NQNOU0
                                                                                                 PD VSAN8G
STRESS          1SPO                 NOUS  NOUN 1             OBJECT OF PREPOSITION       N8MMM0
                                                                                                 PD VSA
WAS             1VX                  BE3S  BE3-AUXILIARY      PREDICATE VERB              VXBE30
                                                                                                 PD PAA
ATTRIBUTED      1V                   PT1   PAST P OF VT1      PREDICATE VERB              PAPT10
                                                                                                 PD
TO              1VPR                 PRE   PREPOSITION        PREPOSITION                 PDPRE0
                                                                                                 PD NQG
VACANCIES       1VPO                 NOUP  NOUN 1             OBJECT OF PREPOSITION       NQNNN2
                                                                                                 PD AC
,               1VPO,                CMA   COMMA              INSERTION                   ACCMA0
                                                                                                 PD AC
WHICH           1VPO7S               RL1   RELATIVE PRN NOM   SUBJECT OF PREDICATE VERB   ACRL10
                                                                                                 PD VCF
HAVE            1VPO7V               VT1P  SINGLE OBJECT VT   PREDICATE VERB              VXVT11
                                                                                                 PD N2A
APPRECIABLE     1VPO7CA              ADJ   ADJECTIVE 1        OBJECT OF PREDICATE VERB    N2AAA0
                                                                                                 PD N5A
MOBILITY        1VPO7O               NOUS  NOUN 1             OBJECT OF PREDICATE VERB    N5MMM0
                                                                                                 PD
AT              1VPO7CPR             PRE   PREPOSITION        PREPOSITION                 PDPRE0
                                                                                                 PD NQG
-72             1VPO7CPO             NUMC  NOUN 2             OBJECT OF PREPOSITION       NQNNN0
                                                                                                 PD
.               1.                   PRD   PERIOD             END OF SENTENCE             PDPRD0
-----------------------------------------------------------------------------------------------------------
 PCOL OVERFLOWS= .    0  NUMBER TEST FAILURES=    14 SHAPER OVERFLOWS=   550 NESTER OVERFLOWS=   155 TIME=   0.0 MINUTES
```

Figure 5. Analysis of Sentence 1.

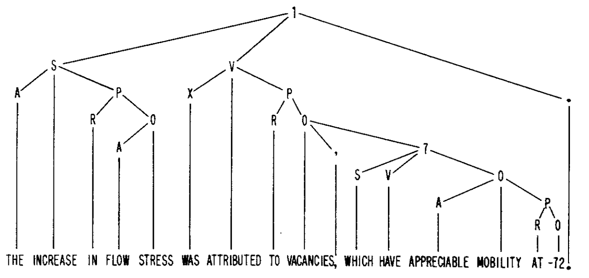

1
S
V
.
A
P
X
P
R
O
R
O
A
7
S
V
O
A
P
R
O
THE INCREASE IN FLOW STRESS WAS ATTRIBUTED TO VACANCIES, WHICH HAVE APPRECIABLE MOBILITY AT -72.

Figure 6. Tree for Sentence 1.

1	declarative	S	subject
2	interrogative	V	verb
3	imperative	O	object
4	subject clause	C	complement
5	object clause	D	adverb
6	complement clause	P	phrase
7	adjective clause	A	attributive
8	adverbial clause	M	participle
		G	gerund
		X	auxiliary verb
		R	phrase or clause introducer (preposition or conjunction)
		E	adverbial noun phrase
		.	period
		,	comma
		+	and/or/but
		=	question mark

Figure 7. Structure Symbols.

nou N8 replaces NQ-G in the pushdown store yielding $\boxed{\text{PD-}\ |\ \text{VS-A}\ |\ \text{N8-G}}$ as a new state for the subsequent processing of "stress".

The dpa itself treats certain adverbs and prepositional phrases as "floating" structures, since little is understood as yet about reliable ways of relating them to the structures they modify. This is reflected, for example, by the fact that, although the VS prediction accepts "in" as a (floating) preposition, it is restored to the pushdown store by the production (VX, pre) $\rightarrow$ pre NQ VX‡. In the editing process, however, certain experimental assumptions have been made. Thus, for example, "in flow stress" and "to vacancies" are provisionally connected to "increase" and to "attributed" respectively. The matter turned out all right in this case, but later examples will show that this success, regrettably, is not universal, and many interesting open questions remain. The under-

‡ This one generic production serves to handle not only VS ("S" for singular) but also VP ("P" for plural) and similar variants denoted by "X".

lying dpa output readily lends itself to experimentation with a variety of potentially useful or elegant representations.

The actual analyzer is not in fact rigorously a dpa. For one thing, it is truncated in a way that reduces it to what any operating machine is for all practical purposes, namely, a finite state machine. Moreover, certain departures are made for the sake of operating economy from the straightforward specification of productions and from strict pushdown store operation. As mentioned regarding the agreement test, Greibach has shown that these departures have no theoretical significance.

So-called "droppable" predictions (Greibach[8], Section 2.3) are another case in point. Their introduction to condense certain pairs of productions into one led to the elimination of 900 productions from the grammar table and a speed-up of machine operation by a factor of 2.5.

The final line of Fig. 5 tells about various tests made during the analysis. The program is written so that only a certain maximum number of predictions (100 in the current version) can be stored in the prediction pool at one time. The maximum must be large enough to allow the pool to accommodate a great many pairs of predictions which are droppable. It was sufficient for the analysis of Sentence 1, as shown by "POOL OVERFLOWS = 0".

The number of otherwise successful lookups in the grammar table which were discarded because they failed the agreement test between the fulfilled prediction (e.g., of a 3rd person singular verb) and the processed syntactic word class (e.g., VT1*P*) is given as "Number Test Failures". Fourteen paths were discontinued with the help of the agreement test, as is shown by "NUMBER TEST FAILURES = 14". These paths probably pertained to the interpretation of "flow" or "stress" as predicate verb of subject "increase" ("The increase in flow stresses . . .).

"Shaper Overflows" indicates the number of paths that were discontinued because of the shaper test, which eliminates pools such that the number of words remaining to be processed is less than the minimum number needed to fulfill the remaining predictions in the pool. There were 550 such instances in the analysis of Sentence 1.

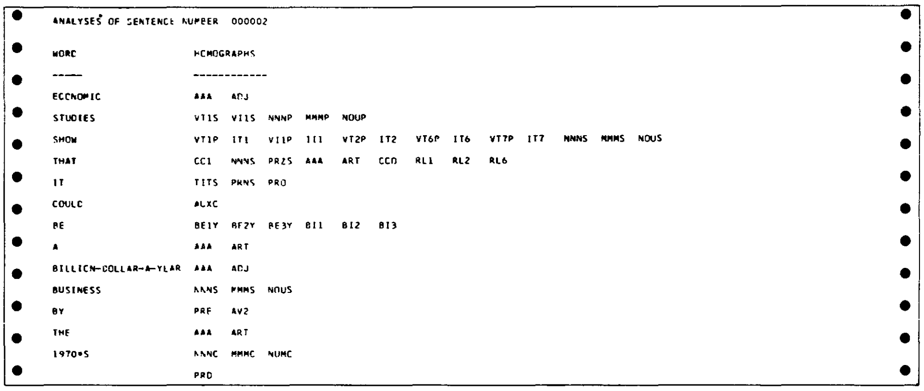
"Nester Overflows" indicates the number of paths that were discontinued because of the nesting test, a comparison of the number of non-droppable predictions in the prediction pool against an allowable maximum each time a new pool is formed. This test effectively truncates the dpa.

Since the number of non-droppable predictions in an active pool corresponds roughly to the depth of nesting of the next word class to be processed on the assumption that all the droppable predictions will eventually be dropped, and in line with the hypothesis of Yngve[20] that English sentences usually do not have a depth of nesting greater than about seven, it is expected that a small finite maximum number of predictions will suffice for the processing of well-formed sentences from natural habitats. At any stage of the analysis of a sentence, therefore, any prediction pool containing more than the maximum number of predictions can be discarded on the assumption that it predicts a depth of nesting never reached by well-formed sentences.

The maximum number was originally set at 12 in order to gain confidence that legitimate paths would be discontinued on this basis only very rarely, if at all. It turned out that no legitimate analysis had a prediction pool which contained more than *six* non-dropped predictions at any stage of the analysis. It is therefore reasonably improbable that a legitimate analysis will be lost because of the nesting test. In case of serious doubt, the maximum can be readily raised albeit at an unpleasant price in machine time. The version of the program with which Sentence 1 was processed had the maximum depth of nesting set to 8. 155 paths were discontinued due to the nesting test. Experiments to test the effect of limiting the maximum extent of self-embedding are also under way.

The analysis of the sentence took less than 0.1 minutes.

2.3 Figure 8 shows the word class coding of Sentence 2, which reads "Economic studies show that it could be a billion-dollar-a-year



```
ANALYSES OF SENTENCE NUMBER  000002

WORC                    HCMOGRAPHS
-----                   ------------
ECCNOMIC                AAA   ADJ
STUDIES                 VT1S  VI1S  NNNP  MMMP  NOUP
SHOW                    VT1P  IT1   VI1P  II1   VT2P  IT2   VT6P  IT6   VT7P  IT7   NNNS  MMMS  NOUS
THAT                    CC1   NNNS  PRZS  AAA   ART   CCO   RL1   RL2   RL6
IT                      TITS  PRNS  PRO
COULC                   AUXC
BE                      BE1Y  BE2Y  BE3Y  BI1   BI2   BI3
A                       AAA   ART
BILLICN-COLLAR-A-YLAR   AAA   ADJ
BUSINESS                NNNS  MMMS  NOUS
BY                      PRE   AV2
THE                     AAA   ART
1970*S                  NNNC  MMMC  NUMC
                        PRD
```

Figure 8. Coding of Sentence 2.

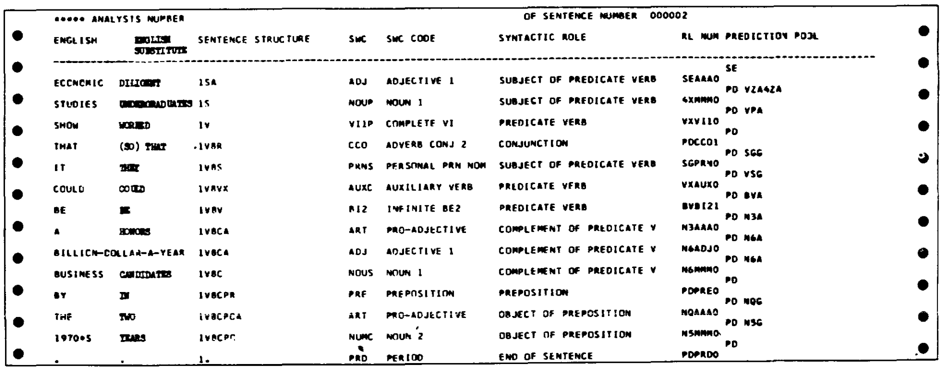
business by the 1970's.". Four distinct analyses were obtained for this sentence, mainly due to the interpretations of "show" and of "that". It turns out in this case that all but one can be eliminated by appropriate modifications of the grammar.

Analysis No. 1 (Fig. 9) treats "that" as a conjunction (CCO) which introduces an adverbial clause with the meaning of "in order that" or "because of the fact that", and "show" as a complete intransitive verb (VI1). This analysis can be made semantically acceptable in a marginal way by replacing the original word forms by others syntactically equivalent in the sense that they are either classified alike in the present grammar, or belong to distinct classes that produce the same new predictions when fulfilling a given prediction (e.g., PRZ and NOU both fulfill a prediction P for which there are rules (P, nnn)). The substitute forms were manually inserted in the column "ENGLISH SUBSTITUTE".

Although the interpretation of "that" as CCO is somewhat far-fetched in this particular sentence, the coding of "that" as CCO is needed for such sentences as "It has been kept polished *that* it may glitter forever.", "I am happy *that* you have succeeded." and "I am surprised *that* he did not pass.". Therefore, the possibility of eliminating this interpertation on general grounds is ruled out.

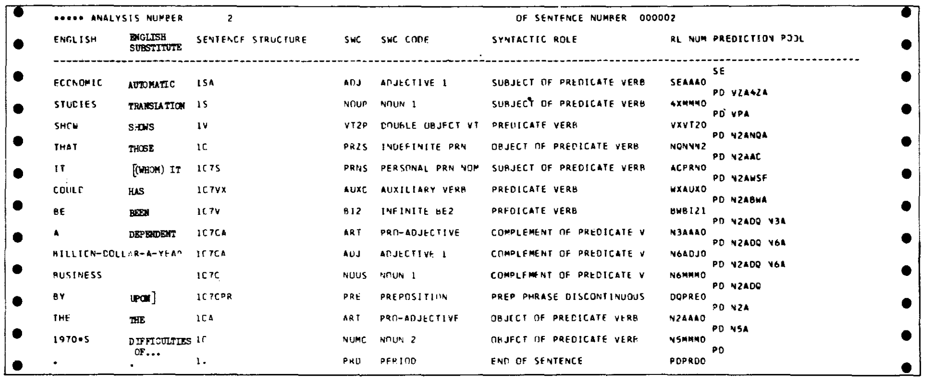
Analysis No. 2 (Fig. 10) treats "show" as a double object transitive verb (VT2), "that" as an indirect object of "show", and "the 1970's" as a direct object of the verb. In this analysis, "that" is modified by the adjective clause ("7") "it could be a billion-dollar-a-year business by". The indicated substitutions make this structure quite plausible so that it too cannot be eliminated on general grounds.

A minor but confusing flaw of Fig. 10 should be pointed out. Although the prediction of an indirect object is represented by "NQ" and that



```
***** ANALYSIS NUMBER                                                OF SENTENCE NUMBER  000002
ENGLISH    ENGLISH        SENTENCE STRUCTURE   SMC   SMC CODE             SYNTACTIC ROLE               RL NUM PREDICTION POOL
           SUBSTITUTE
---------------------------------------------------------------------------------------------------------------------------
                                                                                                              SE
ECCNCMIC   DILIGENT       1SA                  ADJ   ADJECTIVE 1          SUBJECT OF PREDICATE VERB    SEAAAO
                                                                                                              PD VZA4ZA
STUDIES    UNDERGRADUATES 1S                   NOUP  NOUN 1               SUBJECT OF PREDICATE VERB    4XMMMO
                                                                                                              PD VPA
SHOW       WORKED         1V                   VI1P  COMPLETE VI          PREDICATE VERB               VXVI1O
                                                                                                              PD
THAT       (SO) THAT     .1VBR                 CCO   ADVERB CONJ 2        CONJUNCTION                  PDCCO1
                                                                                                              PD SGG
IT         THEY           1VBS                 PRNS  PERSONAL PRN NOM     SUBJECT OF PREDICATE VERB    SGPRNO
                                                                                                              PD VSG
COULD      COULD          1VBVX                AUXC  AUXILIARY VERB       PREDICATE VERB               VXAUXO
                                                                                                              PD BVA
BE         BE             1VBV                 BI2   INFINITE BE2         PREDICATE VERB               BVBI21
                                                                                                              PD N3A
A          HONORS         1VBCA                ART   PRO-ADJECTIVE        COMPLEMENT OF PREDICATE V    N3AAAO
                                                                                                              PD N6A
BILLICN-COLLAR-A-YEAR     1VBCA                ADJ   ADJECTIVE 1          COMPLEMENT OF PREDICATE V    N6ADJO
                                                                                                              PD N6A
BUSINESS   CANDIDATES     1VBC                 NOUS  NOUN 1               COMPLEMENT OF PREDICATE V    N6MMMO
                                                                                                              PD
BY         IN             1VBCPR               PRE   PREPOSITION          PREPOSITION                  PDPREO
                                                                                                              PD NQG
THE        TWO            1VBCPCA              ART   PRO-ADJECTIVE        OBJECT OF PREPOSITION        NQAAAO
                                                                                                              PD NSG
1970*S     YEARS          1VBCPC               NUMC  NOUN 2               OBJECT OF PREPOSITION        N5MMMO
                                                                                                              PD
.          .              1.                   PRD   PERIOD               END OF SENTENCE              PDPRDO
```

Figure 9. Analysis No. 1 of Sentence 2.



```
••••• ANALYSIS NUMPER        2                                                     OF SENTENCE NUMBER  000002
ENGLISH        ENGLISH      SENTENCF STRUCTURE    SWC   SWC CODE             SYNTACTIC ROLE               RL NUM PREDICTION POOL
               SUBSTITUTE
------------------------------------------------------------------------------------------------------------------------------
                                                                                                                 SE
ECCNOMIC       AUTOMATIC    1SA                   ADJ   ADJECTIVE 1          SUBJECT OF PREDICATE VERB    SEAAAO
                                                                                                                 PD VZA4ZA
STUCIES        TRANSLATION  1S                    NOUP  NOUN 1               SUBJECT OF PREDICATE VERB    4XMMMO
                                                                                                                 PD VPA
SHCW           SHOWS        1V                    VT2P  DOUBLE OBJECT VT     PREDICATE VERB               VXVT2O
                                                                                                                 PD N2ANQA
THAT           THOSE        1C                    PRZS  INDEFINITE PRN       OBJECT OF PREDICATE VERB     NONNN2
                                                                                                                 PD N2AAC
IT             [(WHOM) IT   1C7S                  PRNS  PERSONAL PRN NOM     SUBJECT OF PREDICATE VERB    ACPRNO
                                                                                                                 PD N2AWSF
COULC          HAS          1C7VX                 AUXC  AUXILIARY VERB       PREDICATE VERB               WXAUXO
                                                                                                                 PD N2ABWA
BE             BEEN         1C7V                  BI2   INFINITE BE2         PREDICATE VERB               BWBI21
                                                                                                                 PD N2ADQ N3A
A              DEPENDENT    1C7CA                 ART   PRO-ADJECTIVE        COMPLEMENT OF PREDICATE V    N3AAAO
                                                                                                                 PD N2ADQ N6A
BILLICN-DOLLAR-A-YEAR       1C7CA                 ADJ   ADJECTIVE 1          COMPLEMENT OF PREDICATE V    N6ADJO
                                                                                                                 PD N2ADQ N6A
BUSINESS                    1C7C                  NOUS  NOUN 1               COMPLEMENT OF PREDICATE V    N6MMMO
                                                                                                                 PD N2ADQ
BY             UPON]        1C7CPR                PRE   PREPOSITION          PREP PHRASE DISCONTINUOUS    DQPREO
                                                                                                                 PD N2A
THE            THE          1CA                   ART   PRO-ADJECTIVE        OBJECT OF PREDICATE VERB     N2AAAO
                                                                                                                 PD N5A
1970•S         DIFFICULTIES 1C                    NUMC  NOUN 2               OBJECT OF PREDICATE VERB     N5MMMO
                OF...                                                                                            PD
.              .            1.                    PRD   PERIOD               END OF SENTENCE              PDPRDO
```

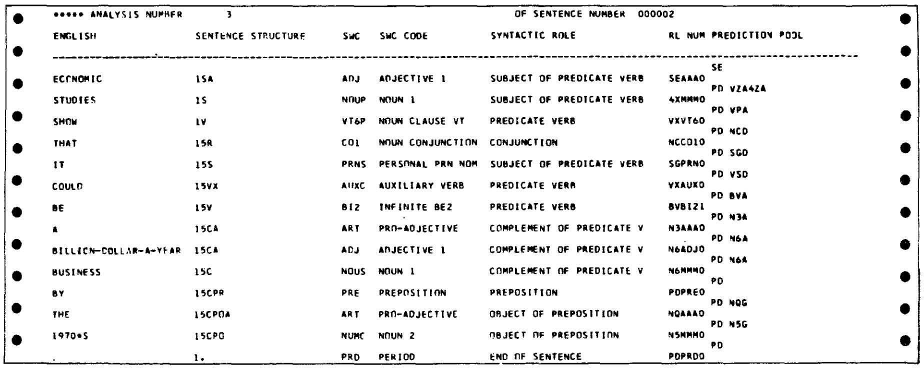
Figure 10. Analysis No. 2 of Sentence 2.

of a direct object by "N2" in the subrule (VX, VT2)-O, the two structures are not distinguished by the current diagramming routine: both are represented by the same structure symbol "O". Therefore, in the structure diagram of Fig. 10, it looks as if the basic pattern of the sentence were "S" (subject) -"V" (verb) -"O" (object) -"." (period), although the presence of two "1O's", one for "that" and the other for "1970's", indicates that the sentence has two distinct object heads. The boundaries of the indirect and direct objects are not explicit in the diagram, but have to be identified with the aid of the pushdown history in the column "PREDICTION POOL". The distinction between structure symbols for an indirect and direct-object has to be embodied in the diagramming routine.

An Analysis No. 3 (Fig. 11), "that" is regarded as a noun conjunction (CO1) which introduces a nominal object clause ("5") of "show" as a noun clause transitive verb (VT6). This is the analysis which is semantically acceptable except for the dependency of the floating prepositional phrase "by the 1970's" which was not handled as well here as similar structures in Sentence 1, for reasons mentioned in Section 2.2.

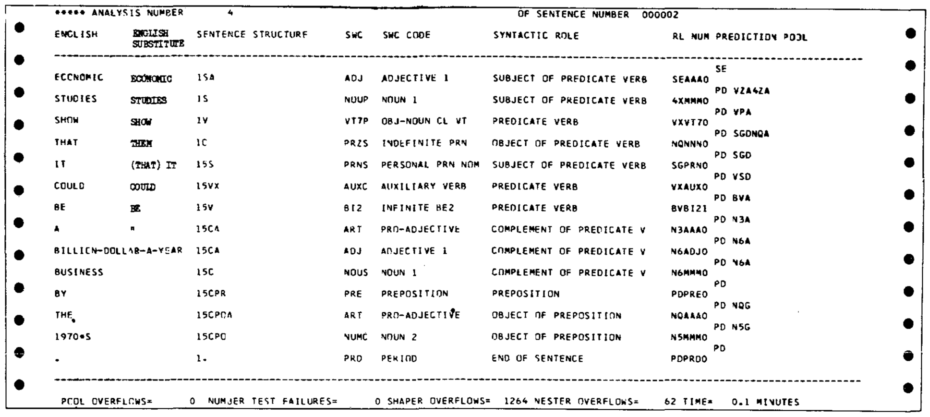
In Analysis No. 4 (Fig. 12), "that" is regarded as an indirect object of the VT7 "show", with "it could be a billion-dollar-a-year business by the 1970's", without an introductory conjunction, interpreted as the object clause of "show". Here again plausible substitutions exist, so that elimination on general grounds is not indicated.

One way of eliminating Analysis No. 1 of Sentence 2 is to preclude the use of "that" by itself as a CCO except when preceded by certain adjectives and past participles of "emotion" as in "I am *happy* (*glad, sorry,* etc.) that you have succeeded" and "I am *surprised* (*disappointed, delighted,*.etc.) that you have passed". Indeed,



```
••••• ANALYSIS NUMBER        3                                         OF SENTENCE NUMBER  000002
ENGLISH                 SENTENCE STRUCTURE    SWC   SWC CODE             SYNTACTIC ROLE               RL NUM PREDICTION POOL
------------------------------------------------------------------------------------------------------------------------------
                                                                                                                 SE
ECCNOMIC                1SA                   ADJ   ADJECTIVE 1          SUBJECT OF PREDICATE VERB    SEAAAO
                                                                                                                 PD VZA4ZA
STUDIES                 1S                    NOUP  NOUN 1               SUBJECT OF PREDICATE VERB    4XMMMO
                                                                                                                 PD VPA
SHOW                    1V                    VT6P  NOUN CLAUSE VT       PREDICATE VERB               VXVT6O
                                                                                                                 PD NCD
THAT                    15R                   CO1   NOUN CONJUNCTION     CONJUNCTION                  NCCD1O
                                                                                                                 PD SGD
IT                      15S                   PRNS  PERSONAL PRN NOM     SUBJECT OF PREDICATE VERB    SGPRNO
                                                                                                                 PD VSD
COULD                   15VX                  AUXC  AUXILIARY VERB       PREDICATE VERB               VXAUXO
                                                                                                                 PD BVA
BE                      15V                   BI2   INFINITE BE2         PREDICATE VERB               BVBI21
                                                                                                                 PD N3A
A                       15CA                  ART   PRO-ADJECTIVE        COMPLEMENT OF PREDICATE V    N3AAAO
                                                                                                                 PD N6A
BILLICN-DOLLAR-A-YEAR   15CA                  ADJ   ADJECTIVE 1          COMPLEMENT OF PREDICATE V    N6ADJO
                                                                                                                 PD N6A
BUSINESS                15C                   NOUS  NOUN 1               COMPLEMENT OF PREDICATE V    N6MMMO
                                                                                                                 PD
BY                      15CPR                 PRE   PREPOSITION          PREPOSITION                  PDPREO
                                                                                                                 PD NQG
THE                     15CPOA                ART   PRO-ADJECTIVE        OBJECT OF PREPOSITION        NQAAAO
                                                                                                                 PD N5G
1970•S                  15CPO                 NUMC  NOUN 2               OBJECT OF PREPOSITION        N5MMMO
                                                                                                                 PD
.                       1.                    PRD   PERIOD               END OF SENTENCE              PDPRDO
```

Figure 11. Analysis No. 3 of Sentence 2.



```
***** ANALYSIS NUMBER       4                                              OF SENTENCE NUMBER  000002
ENGLISH        ENGLISH       SENTENCE STRUCTURE   SWC   SWC CODE             SYNTACTIC ROLE               RL NUM PREDICTION POOL
               SUBSTITUTE
-----------------------------------------------------------------------------------------------------------------------------
                                                                                                                 SE
ECCNOMIC       ECONOMIC      15A                  ADJ   ADJECTIVE 1          SUBJECT OF PREDICATE VERB    SEAAAO
                                                                                                                 PD VZA4ZA
STUDIES        STUDIES       1S                   NOUP  NOUN 1               SUBJECT OF PREDICATE VERB    4XMMMO
                                                                                                                 PD VPA
SHOW           SHOW          1V                   VT7P  OBJ-NOUN CL VT       PREDICATE VERB               VXVT7O
                                                                                                                 PD SGDNQA
THAT           THEM          1C                   PRZS  INDEFINITE PRN       OBJECT OF PREDICATE VERB     NQNNNO
                                                                                                                 PD SGD
IT             (THAT) IT     15S                  PRNS  PERSONAL PRN NOM     SUBJECT OF PREDICATE VERB    SGPRNO
                                                                                                                 PD VSD
COULD          COULD         15VX                 AUXC  AUXILIARY VERB       PREDICATE VERB               VXAUXO
                                                                                                                 PD BVA
BE             BE            15V                  BI2   INFINITE BE2         PREDICATE VERB               BVBI21
                                                                                                                 PD N3A
A              "             15CA                 ART   PRO-ADJECTIVE        COMPLEMENT OF PREDICATE V    N3AAAO
                                                                                                                 PD N6A
BILLICN-DOLLAR-A-YEAR        15CA                 ADJ   ADJECTIVE 1          COMPLEMENT OF PREDICATE V    N6ADJO
                                                                                                                 PD N6A
BUSINESS                     15C                  NOUS  NOUN 1               COMPLEMENT OF PREDICATE V    N6MMMO
                                                                                                                 PD
BY                           15CPR                PRE   PREPOSITION          PREPOSITION                  PDPREO
                                                                                                                 PD NQG
THE                          15CPOA               ART   PRO-ADJECTIVE        OBJECT OF PREPOSITION        NQAAAO
                                                                                                                 PD N5G
1970*S                       15CPO                NUMC  NOUN 2               OBJECT OF PREPOSITION        N5MMMO
                                                                                                                 PD
.                            1.                   PRO   PERIOD               END OF SENTENCE              PDPRDO
-----------------------------------------------------------------------------------------------------------------------------
 PCOL OVERFLOWS=      0  NUMJER TEST FAILURES=      0 SHAPER OVERFLOWS=  1264 NESTER OVERFLOWS=     62 TIME=   0.1 MINUTES
```

Figure 12. Analysis No. 4 of Sentence 2.

the probability of the occurrences of such sentences as "It has been kept polished *that* it may glitter forever." will be fairly low since one would more often say "so that" or "in order that" on such occasions.

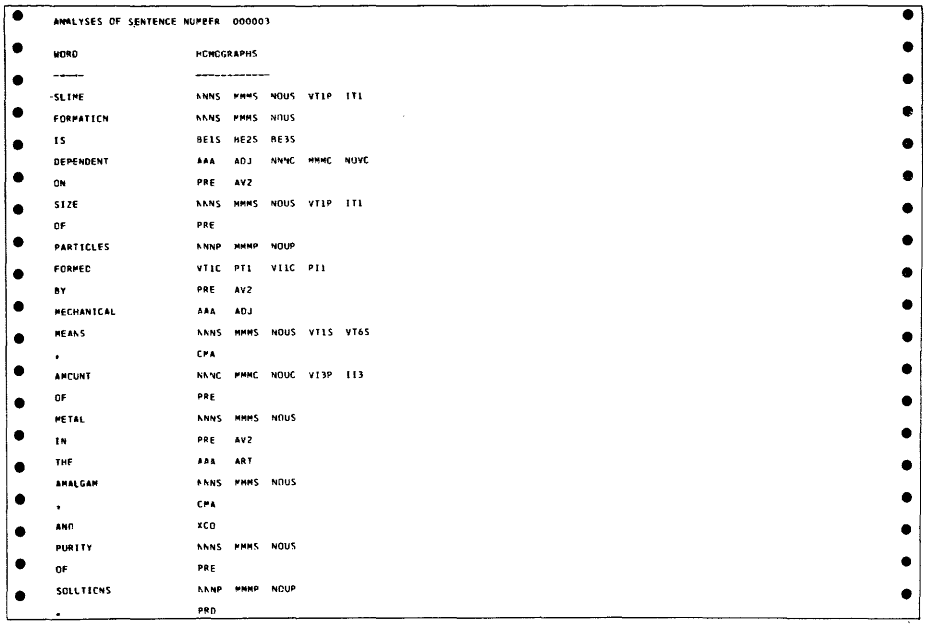
The emegence of Analysis No. 1 may also be attributed to the coding of "show" as an intransitive verb (VI1) for sentences such as "They *show up* every morning at eight." or "The tuberculosis tests often *show up* positive.". Since "show" as an intransitive verb seems always to require a special adverb to follow it, establishing such a subclass of VI1 and making a prediction of such an adverb will make it possible to discard Analysis No. 1 of Sentence 2.

In Analysis No. 2 "that" is interpreted as the indirect object of a double object transitive verb (VT2). It is common to use "that" as an indirect object of a VT2, as in "He gave *that* serious consideration." which means "He gave serious consideration to *that* (matter).". However, the occurrence of "that" as PRZ modified by an adjective clause which is not itself introduced by the relative pronoun "which", either in the nominative case (RL1) or in the accusative case (RL2), has some unusual features. It is awkward but admissible to say "He does that which pleases most of his constituents.". ("He does what pleases . . . " is smoother), but it is poetic, perhaps normally ungrammatical, to omit "which" and say "He has that all people desire."; the spoken version requires intonational gymnastics to be understood, and like the written version, seems more at home in a sermon than in scientific prose. Analysis No. 2 could be deleted by prohibiting "that" as PRZ from being modified by an adjective clause not introduced by "which".

As for the fourth analysis, on the assumption that "that" as PRZ is used as the object of a VT7 only in sentences one might address to children such as "We tell that when and where it should stop.", with "that" meaning "(to) that toy", and that such sentences most likely never appear in scientific papers (Would Piaget accept this?), one could eliminate Analysis No. 4 by prohibiting the acceptance of "that" by the indirect object prediction ("NQ") for verbs of category VT7. At such junctures one must be prepared to make explicit decisions about what will be regarded as grammatical and what will not, and assess the consequences of these decisions!

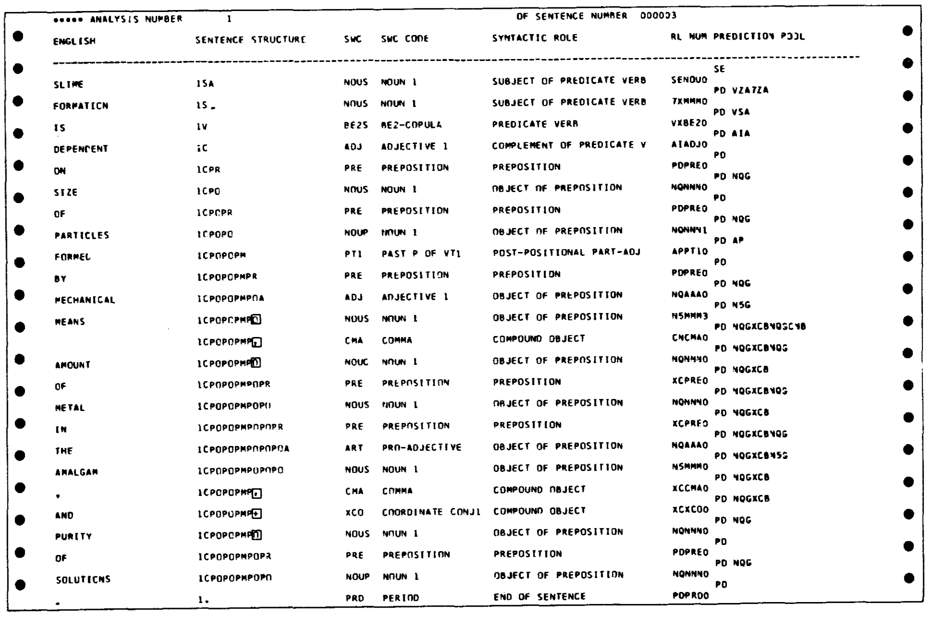
2.4 Sentence 3, "Slime formation is dependent on size of particles formed by mechanical means, amount of metal in the amalgam, and purity of solutions." was coded as shown in Fig. 13. The five analyses obtained for this sentence are shown in Figs. 14 through 18.

The selected syntactic word classes are the same in the first two (Figs. 14 and 15). It therefore is not homographs, but multiple functions of word classes that give rise to these two analyses. The differences between the two can best be appreciated by looking at the structure symbols which the "," and "+" symbols connect together in the sentence structure diagrams. In Analysis No. 1, two ","s and a "+" appear at the same level, showing that "means", "amount" and "purity" are all objects of the preposition "(formed) by" (i.e., "formed by



```
ANALYSES OF SENTENCE NUMBER  000003

WORD                HOMOGRAPHS
-----               ----------
-SLIME              NNNS  MMMS  NOUS  VT1P  IT1
FORMATICN           NNNS  MMMS  NOUS
IS                  BE1S  BE2S  BE3S
DEPENDENT           AAA   ADJ   NNNC  MMMC  NOVC
ON                  PRE   AV2
SIZE                NNNS  MMMS  NOUS  VT1P  IT1
OF                  PRE
PARTICLES           NNNP  MMMP  NOUP
FORMED              VT1C  PT1   VI1C  PI1
BY                  PRE   AV2
MECHANICAL          AAA   ADJ
MEANS               NNNS  MMMS  NOUS  VT1S  VT6S
,                   CMA
AMCUNT              NNNC  MMMC  NOUC  VI3P  II3
OF                  PRE
METAL               NNNS  MMMS  NOUS
IN                  PRE   AV2
THE                 AAA   ART
AMALGAM             NNNS  MMMS  NOUS
,                   CMA
AND                 XCO
PURITY              NNNS  MMMS  NOUS
OF                  PRE
SOLLTICNS           NNNP  MMMP  NOUP
.                   PRD
```

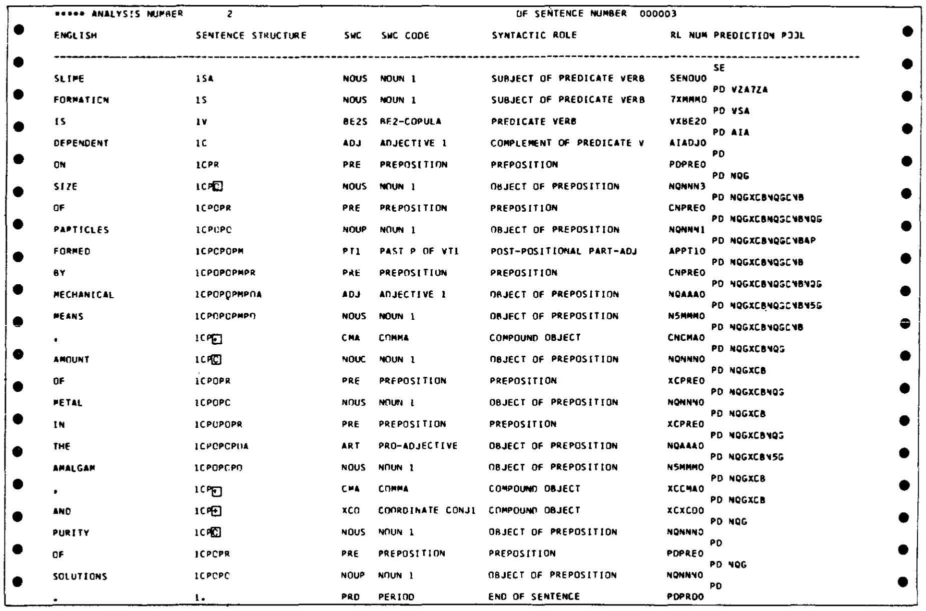
Figure 13. Coding of Sentence 3.



```
••••• ANALYSIS NUMBER       1                               OF SENTENCE NUMBER  000003
ENGLISH          SENTENCE STRUCTURE   SWC   SWC CODE            SYNTACTIC ROLE                RL NUM PREDICTION POOL
---------------------------------------------------------------------------------------------------------------------
                                                                                                     SE
SLIME            1SA                  NOUS  NOUN 1              SUBJECT OF PREDICATE VERB     SENOUO
                                                                                                     PD VZA7ZA
FORMATICN        1S_                  NOUS  NOUN 1              SUBJECT OF PREDICATE VERB     7XMMMO
                                                                                                     PD VSA
IS               1V                   BE2S  BE2-COPULA          PREDICATE VERB                VXBE2O
                                                                                                     PD AIA
DEPENDENT        1C                   ADJ   ADJECTIVE 1         COMPLEMENT OF PREDICATE V     AIADJO
                                                                                                     PD
ON               1CPR                 PRE   PREPOSITION         PREPOSITION                   PDPREO
                                                                                                     PD NQG
SIZE             1CPO                 NOUS  NOUN 1              OBJECT OF PREPOSITION         NQNNNO
                                                                                                     PD
OF               1CPOPR               PRE   PREPOSITION         PREPOSITION                   PDPREO
                                                                                                     PD NQG
PARTICLES        1CPOPO               NOUP  NOUN 1              OBJECT OF PREPOSITION         NONNN1
                                                                                                     PD AP
FORMEL           1CPOPOPM             PT1   PAST P OF VT1       POST-POSITIONAL PART-ADJ      APPT1O
                                                                                                     PD
BY               1CPOPOPMPR           PRE   PREPOSITION         PREPOSITION                   PDPREO
                                                                                                     PD NQG
MECHANICAL       1CPOPOPMPOA          ADJ   ADJECTIVE 1         OBJECT OF PREPOSITION         NQAAAO
                                                                                                     PD N5G
MEANS            1CPOPCPMPO           NOUS  NOUN 1              OBJECT OF PREPOSITION         N5MMM3
                                                                                                     PD NQGXCBNQGCNB
                 1CPOPOPMP,           CMA   COMMA               COMPOUND OBJECT               CNCMAO
                                                                                                     PD NQGXCBNQG
AMOUNT           1CPOPOPMPO           NOUC  NOUN 1              OBJECT OF PREPOSITION         NQNNNO
                                                                                                     PD NQGXCB
OF               1CPOPOPMPOPR         PRE   PREPOSITION         PREPOSITION                   XCPREO
                                                                                                     PD NQGXCBNQG
METAL            1CPOPOPMPOPO         NOUS  NOUN 1              OBJECT OF PREPOSITION         NQNNNO
                                                                                                     PD NQGXCB
IN               1CPOPOPMPOPOPR       PRE   PREPOSITION         PREPOSITION                   XCPREO
                                                                                                     PD NQGXCBNQG
THE              1CPOPOPMPOPOPOA      ART   PRO-ADJECTIVE       OBJECT OF PREPOSITION         NQAAAO
                                                                                                     PD NQGXCBN5G
AMALGAM          1CPOPOPMPOPOPO       NOUS  NOUN 1              OBJECT OF PREPOSITION         N5MMMO
                                                                                                     PD NQGXCB
,                1CPOPOPMP,           CMA   COMMA               COMPOUND OBJECT               XCCMAO
                                                                                                     PD NQGXCB
AND              1CPOPOPMP+           XCO   COORDINATE CONJ1    COMPOUND OBJECT               XCXCOO
                                                                                                     PD NQG
PURITY           1CPOPOPMPO           NOUS  NOUN 1              OBJECT OF PREPOSITION         NQNNNO
                                                                                                     PD
OF               1CPOPOPMPOPR         PRE   PREPOSITION         PREPOSITION                   PDPREO
                                                                                                     PD NQG
SOLUTICNS        1CPOPOPMPOPO         NOUP  NOUN 1              OBJECT OF PREPOSITION         NQNNNO
                                                                                                     PD
.                1.                   PRD   PERIOD              END OF SENTENCE               PDPROO
```

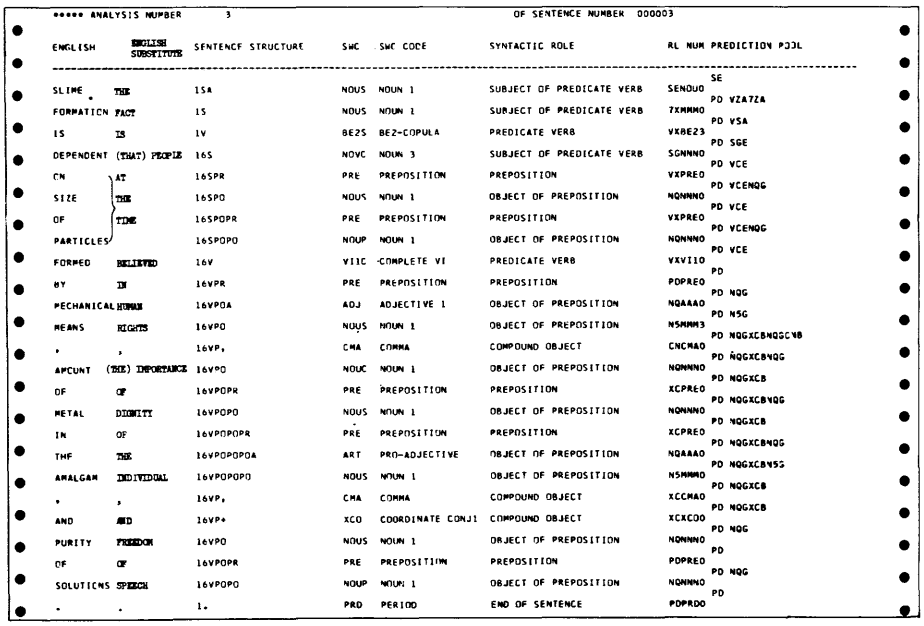
Figure 14. Analysis No. 1 of Sentence 3.

means . . . , amount, . . . and purity . . . "). In Analysis No. 2, on the other hand, "size", "amount" and "purity" appear at the same level, forming a three-member object of "(dependent) on", (i.e., "dependent on size . . . , amount, . . . and purity . . . "). Although it is the second analysis that is semantically acceptable for this particular sentence, the first analysis is syntactically as legitimate as the second one. (See Type 2 ambiguity, Section 4.1 of Greibach[8]). Rejecting it in this case requires much deeper insight into semantics than



```
***** ANALYSIS NUMBER     2                                      OF SENTENCE NUMBER  000003
ENGLISH          SENTENCE STRUCTURE    SWC   SWC CODE            SYNTACTIC ROLE             RL NUM PREDICTION POOL
-------------------------------------------------------------------------------------------------------------
                                                                                                  SE
SLIME            1SA                   NOUS  NOUN 1              SUBJECT OF PREDICATE VERB  SENOUO
                                                                                                  PD VZA7ZA
FORMATION        1S                    NOUS  NOUN 1              SUBJECT OF PREDICATE VERB  7XMMMO
                                                                                                  PD VSA
IS               1V                    BE2S  BE2-COPULA          PREDICATE VERB             VXBE20
                                                                                                  PD AIA
DEPENDENT        1C                    ADJ   ADJECTIVE 1         COMPLEMENT OF PREDICATE V  AIADJO
                                                                                                  PD
ON               1CPR                  PRE   PREPOSITION         PREPOSITION                PDPREO
                                                                                                  PD NQG
SIZE             1CPC                  NOUS  NOUN 1              OBJECT OF PREPOSITION      NQNNN3
                                                                                                  PD NQGXCBNQGCNB
OF               1CPOPR                PRE   PREPOSITION         PREPOSITION                CNPREO
                                                                                                  PD NQGXCBNQGCNBNQG
PARTICLES        1CPOPC                NOUP  NOUN 1              OBJECT OF PREPOSITION      NQNNN1
                                                                                                  PD NQGXCBNQGCNBAP
FORMED           1CPCPOPM              PT1   PAST P OF VT1       POST-POSITIONAL PART-ADJ   APPT10
                                                                                                  PD NQGXCBNQGCNB
BY               1CPOPOPMPR            PRE   PREPOSITION         PREPOSITION                CNPREO
                                                                                                  PD NQGXCBNQGCNBNQG
MECHANICAL       1CPOPOPMPOA           ADJ   ADJECTIVE 1         OBJECT OF PREPOSITION      NQAAAO
                                                                                                  PD NQGXCBNQGCNBN5G
MEANS            1CPOPOPMPO            NOUS  NOUN 1              OBJECT OF PREPOSITION      N5MMMO
                                                                                                  PD NQGXCBNQGCNB
,                1CP,                  CMA   COMMA               COMPOUND OBJECT            CNCMAO
                                                                                                  PD NQGXCBNQG
AMOUNT           1CPC                  NOUC  NOUN 1              OBJECT OF PREPOSITION      NQNNNO
                                                                                                  PD NQGXCB
OF               1CPOPR                PRE   PREPOSITION         PREPOSITION                XCPREO
                                                                                                  PD NQGXCBNQG
METAL            1CPOPC                NOUS  NOUN 1              OBJECT OF PREPOSITION      NQNNNO
                                                                                                  PD NQGXCB
IN               1CPOPOPR              PRE   PREPOSITION         PREPOSITION                XCPREO
                                                                                                  PD NQGXCBNQG
THE              1CPOPCPOA             ART   PRO-ADJECTIVE       OBJECT OF PREPOSITION      NQAAAO
                                                                                                  PD NQGXCBN5G
AMALGAM          1CPOPCPO              NOUS  NOUN 1              OBJECT OF PREPOSITION      N5MMMO
                                                                                                  PD NQGXCB
,                1CP,                  CMA   COMMA               COMPOUND OBJECT            XCCMAO
                                                                                                  PD NQGXCB
AND              1CP+                  XCO   COORDINATE CONJ1    COMPOUND OBJECT            XCXCOO
                                                                                                  PD NQG
PURITY           1CPC                  NOUS  NOUN 1              OBJECT OF PREPOSITION      NQNNNO
                                                                                                  PD
OF               1CPCPR                PRE   PREPOSITION         PREPOSITION                PDPREO
                                                                                                  PD NQG
SOLUTIONS        1CPCPC                NOUP  NOUN 1              OBJECT OF PREPOSITION      NQNNNO
                                                                                                  PD
.                1.                    PRD   PERIOD              END OF SENTENCE            PDPRDO
```

Figure 15. Analysis No. 2 of Sentence 3.

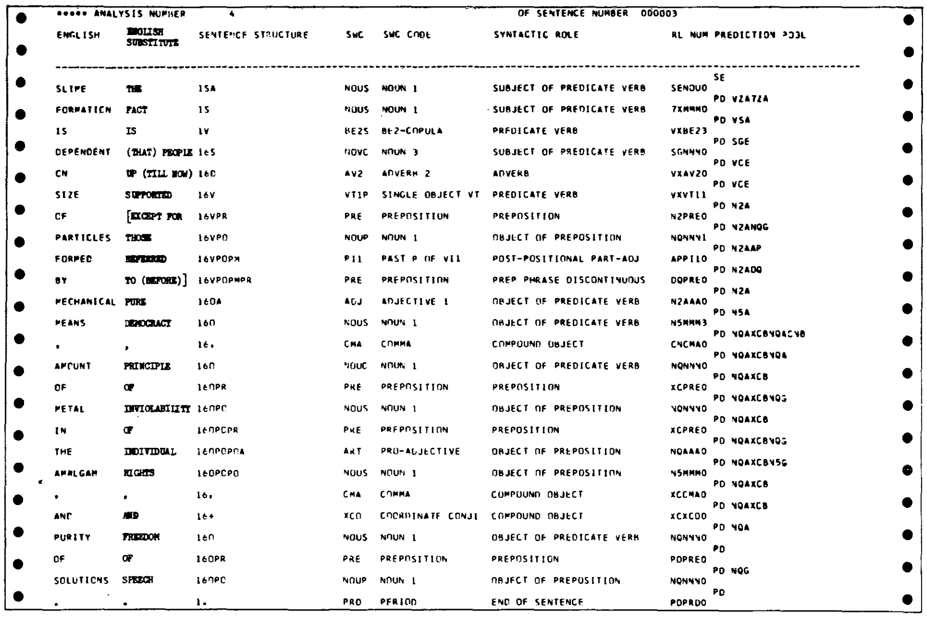


```
***** ANALYSIS NUMBER     3                                                OF SENTENCE NUMBER  000003
ENGLISH    ENGLISH          SENTENCE STRUCTURE  SWC   SWC CODE            SYNTACTIC ROLE             RL NUM PREDICTION POOL
           SUBSTITUTE
-----------------------------------------------------------------------------------------------------------------------
                                                                                                            SE
SLIME      THE              1SA                 NOUS  NOUN 1              SUBJECT OF PREDICATE VERB  SENOUO
                                                                                                            PD VZA7ZA
FORMATION  FACT             1S                  NOUS  NOUN 1              SUBJECT OF PREDICATE VERB  7XMMMO
                                                                                                            PD VSA
IS         IS               1V                  BE2S  BE2-COPULA          PREDICATE VERB             VXBE23
                                                                                                            PD SGE
DEPENDENT  (THAT) PEOPLE    16S                 NOVC  NOUN 3              SUBJECT OF PREDICATE VERB  SGNNNO
                                                                                                            PD VCE
ON        \ AT              16SPR               PRE   PREPOSITION         PREPOSITION                VXPREO
                                                                                                            PD VCENQG
SIZE       | THE            16SPO               NOUS  NOUN 1              OBJECT OF PREPOSITION      NQNNNO
                                                                                                            PD VCE
OF         } TIME           16SPOPR             PRE   PREPOSITION         PREPOSITION                VXPREO
                                                                                                            PD VCENQG
PARTICLES /                 16SPOPO             NOUP  NOUN 1              OBJECT OF PREPOSITION      NQNNNO
                                                                                                            PD VCE
FORMED     BELIEVED         16V                 VI1C  COMPLETE VI         PREDICATE VERB             VXVI10
                                                                                                            PD
BY         IN               16VPR               PRE   PREPOSITION         PREPOSITION                PDPREO
                                                                                                            PD NQG
MECHANICAL HUMAN            16VPOA              ADJ   ADJECTIVE 1         OBJECT OF PREPOSITION      NQAAAO
                                                                                                            PD N5G
MEANS      RIGHTS           16VPO               NOUS  NOUN 1              OBJECT OF PREPOSITION      N5MMM3
                                                                                                            PD NQGXCBNQGCNB
,          ,                16VP,               CMA   COMMA               COMPOUND OBJECT            CNCMAO
                                                                                                            PD NQGXCBNQG
AMOUNT     (THE) IMPORTANCE 16VPO               NOUC  NOUN 1              OBJECT OF PREPOSITION      NQNNNO
                                                                                                            PD NQGXCB
OF         OF               16VPOPR             PRE   PREPOSITION         PREPOSITION                XCPREO
                                                                                                            PD NQGXCBNQG
METAL      DIGNITY          16VPOPO             NOUS  NOUN 1              OBJECT OF PREPOSITION      NQNNNO
                                                                                                            PD NQGXCB
IN         OF               16VPOPOPR           PRE   PREPOSITION         PREPOSITION                XCPREO
                                                                                                            PD NQGXCBNQG
THE        THE              16VPOPOPOA          ART   PRO-ADJECTIVE       OBJECT OF PREPOSITION      NQAAAO
                                                                                                            PD NQGXCBN5G
AMALGAM    INDIVIDUAL       16VPOPOPO           NOUS  NOUN 1              OBJECT OF PREPOSITION      N5MMMO
                                                                                                            PD NQGXCB
,          ,                16VP,               CMA   COMMA               COMPOUND OBJECT            XCCMAO
                                                                                                            PD NQGXCB
AND        AND              16VP+               XCO   COORDINATE CONJ1    COMPOUND OBJECT            XCXCOO
                                                                                                            PD NQG
PURITY     FREEDOM          16VPO               NOUS  NOUN 1              OBJECT OF PREPOSITION      NQNNNO
                                                                                                            PD
OF         OF               16VPOPR             PRE   PREPOSITION         PREPOSITION                PDPREO
                                                                                                            PD NQG
SOLUTIONS  SPEECH           16VPOPO             NOUP  NOUN 1              OBJECT OF PREPOSITION      NQNNNO
                                                                                                            PD
.          .                1.                  PRD   PERIOD              END OF SENTENCE            PDPRDO
```

Figure 16. Analysis No. 3 of Sentence 3.

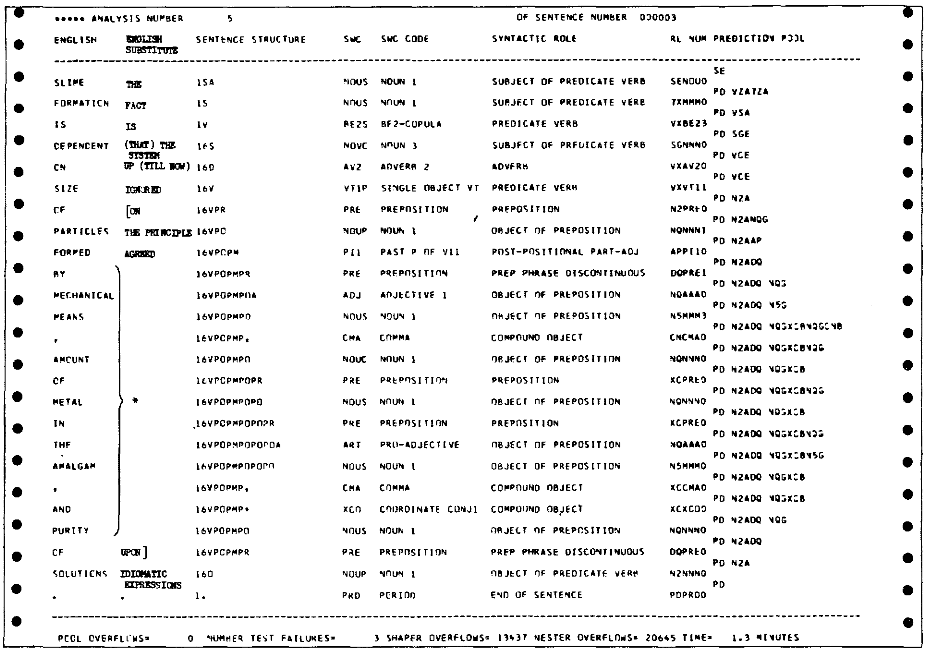
is now available. Cases such as these underline the great importance of retaining human links in any chain for natural language data processing and the danger of relying on any method of syntactic analysis that does not properly account for ambiguities.

The emergence of a compound object "amalgam, and purity" or "metal, and purity" has been precluded since the current grammar regards as ill-formed the use of a comma for a two-member compound noun phrase. This structure has been excluded from the grammar



```
***** ANALYSIS NUMBER      4                                                   OF SENTENCE NUMBER  000003
ENGLISH     ENGLISH        SENTENCE STRUCTURE    SWC    SWC CODE           SYNTACTIC ROLE                RL NUM PREDICTION POOL
            SUBSTITUTE
------------------------------------------------------------------------------------------------------------------------------
                                                                                                                SE
SLIME       THE            1SA                   NOUS   NOUN 1             SUBJECT OF PREDICATE VERB     SENOUO
                                                                                                                PD VZA7ZA
FORMATION   FACT           1S                    NOUS   NOUN 1             SUBJECT OF PREDICATE VERB     7XMMMO
                                                                                                                PD VSA
IS          IS             1V                    BE2S   BE2-COPULA         PREDICATE VERB                VXBE23
                                                                                                                PD SGE
DEPENDENT   (THAT) PEOPLE  16S                   NOVC   NOUN 3             SUBJECT OF PREDICATE VERB     SGNNNO
                                                                                                                PD VCE
ON          UP (TILL NOW)  16O                   AV2    ADVERB 2           ADVERB                        VXAV20
                                                                                                                PD VCE
SIZE        SUPPORTED      16V                   VT1P   SINGLE OBJECT VT   PREDICATE VERB                VXVT11
                                                                                                                PD N2A
OF          [EXCEPT FOR    16VPR                 PRE    PREPOSITION        PREPOSITION                   N2PREO
                                                                                                                PD N2ANQG
PARTICLES   THOSE          16VPO                 NOUP   NOUN 1             OBJECT OF PREPOSITION         NONNN1
                                                                                                                PD N2AAP
FORMED      REFERRED       16VPOPM               PI1    PAST P OF VI1      POST-POSITIONAL PART-ADJ      APPI10
                                                                                                                PD N2ADQ
BY          TO (BEFORE)]   16VPOPMPR             PRE    PREPOSITION        PREP PHRASE DISCONTINUOUS     DQPREO
                                                                                                                PD N2A
MECHANICAL  PURE           16OA                  ADJ    ADJECTIVE 1        OBJECT OF PREDICATE VERB      N2AAAO
                                                                                                                PD N5A
MEANS       DEMOCRACY      16O                   NOUS   NOUN 1             OBJECT OF PREDICATE VERB      N5MMM3
                                                                                                                PD NQAXCBNQACNB
,           ,              16,                   CMA    COMMA              COMPOUND OBJECT               CNCMAO
                                                                                                                PD NQAXCBNQA
AMOUNT      PRINCIPLE      16O                   NOUC   NOUN 1             OBJECT OF PREDICATE VERB      NQNNNO
                                                                                                                PD NQAXCB
OF          OF             16OPR                 PRE    PREPOSITION        PREPOSITION                   XCPREO
                                                                                                                PD NQAXCBNQG
METAL       INVIOLABILITY  16OPO                 NOUS   NOUN 1             OBJECT OF PREPOSITION         NQNNNO
                                                                                                                PD NQAXCB
IN          OF             16OPOPR               PRE    PREPOSITION        PREPOSITION                   XCPREO
                                                                                                                PD NQAXCBNQG
THE         INDIVIDUAL     16OPOPOA              ART    PRO-ADJECTIVE      OBJECT OF PREPOSITION         NQAAAO
                                                                                                                PD NQAXCBN5G
AMALGAM     RIGHTS         16OPOPO               NOUS   NOUN 1             OBJECT OF PREPOSITION         N5MMMO
                                                                                                                PD NQAXCB
,           ,              16,                   CMA    COMMA              COMPOUND OBJECT               XCCMAO
                                                                                                                PD NQAXCB
AND         AND            16+                   XCO    COORDINATE CONJ1   COMPOUND OBJECT               XCXCOO
                                                                                                                PD NQA
PURITY      FREEDOM        16O                   NOUS   NOUN 1             OBJECT OF PREDICATE VERB      NQNNNO
                                                                                                                PD
OF          OF             16OPR                 PRE    PREPOSITION        PREPOSITION                   PDPREO
                                                                                                                PD NQG
SOLUTIONS   SPEECH         16OPO                 NOUP   NOUN 1             OBJECT OF PREPOSITION         NQNNNO
                                                                                                                PD
.           .              1.                    PRD    PERIOD             END OF SENTENCE               PDPRDO
```

Figure 17. Analysis No. 4 of Sentence 3.



```
***** ANALYSIS NUMBER      5                                                   OF SENTENCE NUMBER  000003
ENGLISH     ENGLISH          SENTENCE STRUCTURE    SWC    SWC CODE           SYNTACTIC ROLE                RL NUM PREDICTION POOL
            SUBSTITUTE
--------------------------------------------------------------------------------------------------------------------------------
                                                                                                                  SE
SLIME       THE              1SA                   NOUS   NOUN 1             SUBJECT OF PREDICATE VERB     SENOUO
                                                                                                                  PD VZA7ZA
FORMATION   FACT             1S                    NOUS   NOUN 1             SUBJECT OF PREDICATE VERB     7XMMMO
                                                                                                                  PD VSA
IS          IS               1V                    BE2S   BE2-COPULA         PREDICATE VERB                VXBE23
                                                                                                                  PD SGE
DEPENDENT   (THAT) THE       16S                   NOVC   NOUN 3             SUBJECT OF PREDICATE VERB     SGNNNO
             SYSTEM                                                                                               PD VCE
ON          UP (TILL NOW)    16O                   AV2    ADVERB 2           ADVERB                        VXAV20
                                                                                                                  PD VCE
SIZE        IGNORED          16V                   VT1P   SINGLE OBJECT VT   PREDICATE VERB                VXVT11
                                                                                                                  PD N2A
OF          [ON              16VPR                 PRE    PREPOSITION        PREPOSITION                   N2PREO
                                                                                                                  PD N2ANQG
PARTICLES   THE PRINCIPLE    16VPO                 NOUP   NOUN 1             OBJECT OF PREPOSITION         NQNNN1
                                                                                                                  PD N2AAP
FORMED      AGREED           16VPOPM               PI1    PAST P OF VI1      POST-POSITIONAL PART-ADJ      APPI10
                                                                                                                  PD N2ADQ
BY         ⎫                 16VPOPMPR             PRE    PREPOSITION        PREP PHRASE DISCONTINUOUS     DQPRE1
                                                                                                                  PD N2ADQ NQG
MECHANICAL ⎪                 16VPOPMPOA            ADJ    ADJECTIVE 1        OBJECT OF PREPOSITION         NQAAAO
                                                                                                                  PD N2ADQ N5G
MEANS      ⎪                 16VPOPMPO             NOUS   NOUN 1             OBJECT OF PREPOSITION         N5MMM3
                                                                                                                  PD N2ADQ NQGXCBNQGCNB
,          ⎪                 16VPOPMP,             CMA    COMMA              COMPOUND OBJECT               CNCMAO
                                                                                                                  PD N2ADQ NQGXCBNQG
AMOUNT     ⎪                 16VPOPMPO             NOUC   NOUN 1             OBJECT OF PREPOSITION         NQNNNO
                                                                                                                  PD N2ADQ NQGXCB
OF         ⎪                 16VPOPMPOPR           PRE    PREPOSITION        PREPOSITION                   XCPREO
                                                                                                                  PD N2ADQ NQGXCBNQG
METAL      ⎬ *               16VPOPMPOPO           NOUS   NOUN 1             OBJECT OF PREPOSITION         NQNNNO
                                                                                                                  PD N2ADQ NQGXCB
IN         ⎪                 16VPOPMPOPOPR         PRE    PREPOSITION        PREPOSITION                   XCPREO
                                                                                                                  PD N2ADQ NQGXCBNQG
THE        ⎪                 16VPOPMPOPOPOA        ART    PRO-ADJECTIVE      OBJECT OF PREPOSITION         NQAAAO
                                                                                                                  PD N2ADQ NQGXCBN5G
AMALGAM    ⎪                 16VPOPMPOPOPO         NOUS   NOUN 1             OBJECT OF PREPOSITION         N5MMMO
                                                                                                                  PD N2ADQ NQGXCB
,          ⎪                 16VPOPMP,             CMA    COMMA              COMPOUND OBJECT               XCCMAO
                                                                                                                  PD N2ADQ NQGXCB
AND        ⎪                 16VPOPMP+             XCO    COORDINATE CONJ1   COMPOUND OBJECT               XCXCOO
                                                                                                                  PD N2ADQ NQG
PURITY     ⎭                 16VPOPMPO             NOUS   NOUN 1             OBJECT OF PREPOSITION         NQNNNO
                                                                                                                  PD N2ADQ
OF          UPON]            16VPOPMPR             PRE    PREPOSITION        PREP PHRASE DISCONTINUOUS     DQPREO
                                                                                                                  PD N2A
SOLUTIONS   IDIOMATIC        16O                   NOUP   NOUN 1             OBJECT OF PREDICATE VERB      N2NNNO
             EXPRESSIONS                                                                                          PD
.           .                1.                    PRD    PERIOD             END OF SENTENCE               PDPRDO
--------------------------------------------------------------------------------------------------------------------------------
POOL OVERFLOWS=      0  NUMBER TEST FAILURES=      3 SHAPER OVERFLOWS= 13437 NESTER OVERFLOWS= 20645 TIME=    1.3 MINUTES
```

Figure 18. Analysis No. 5 of Sentence 3.

not because it would be difficult to recognize it as well-formed, but rather because its inclusion at this time would cause an excessive increase in the number of semantically unacceptable analyses for common sentence types which do not have such a structure among their normal semantically acceptable analyses. For example, a sentence such as "Time passes, and the world changes." would give two semantically unacceptable analyses if a compound noun phrase "amalgam, and purity" were allowed as well-formed. In one analysis, "time (NOU) passes

(NOU), and the world" would be regarded as a compound subject of "changes (VI1)", while in the second analysis, "passes (NOU), and the world (NOU) changes (NOU)" would be regarded as a compound object of the imperative verb "time (IT1)".

The emergence of a three-member noun phrase "particles, amount, and purity" has also been precluded since the current grammar does not accept a post-positional adjective, participle or clause which modifies the first member of a compound noun phrase. Rules can readily be added to the grammar to enable the analyzer to accept these structures. Such rules, if embodied in the grammar, would yield a semantically and syntactically acceptable analysis for sentences such as "I like wine *imported from France,* beer from Germany and sake from Japan.", although they would have the unpleasant effect of producing a semantically unacceptable analysis "particles, amount, and purity" in cases such as Sentence 3.

The remaining three analyses of Sentence 3 represent a structure similar to that of "The fact is smoking kills." in which "smoking kills" constitutes a complement clause of "is". The problem here is that too many means of eliminating these three analyses suggest themselves and that the consequences of any alternative are difficult to predict in detail *a priori.*

One obvious technique would be to treat "The fact is smoking kills." as ill-formed insisting instead on "The fact is: smoking kills.". In the absence of enforceable normative techniques, and that is the usual practical situation, this choice is less attractive than might appear at first thought.

A more promising approach might be based on a refinement of word classes. This leaves open the acceptability of the three analyses under appropriate substitution. Since "formation" does not seem to belong to the category of nouns such as "fact", "plan" and "idea" which, as the subject of a copula "be", can introduce a complement clause, all three analyses could be discarded by refining the nominal class definitions. Again in all three analyses, the word form "dependent" is interpreted as NOVC with the meaning of "one who depends on or looks to another for support", as in "I have one *dependent.*" or "The *dependent* and the underprivileged need greater educational opportunities.". The analyses could therefore be deleted also by refining the specification of noun classes in order to group "dependent" with other nouns which cannot form the head of a noun phrase without being preceded by one of such noun phrase introducers as "one", "a", "the", or "my".

In Analysis No. 3 (Fig. 16), the complement clause is composed of "dependent" as subject and "formed" as predicate verb. "Formed" is a complete intransitive verb (VI1C) as in "Ice *formed* under the wings.". The analysis can be made semantically acceptable by replacing the original word forms by those manually inserted in the column "ENGLISH SUBSTITUTE".

In Analysis No. 4 (Fig. 17) the complement clause is composed of "dependent" as plural subject, "size" as predicate verb, and "mechanical means, amount . . . , and purity . . . " as object of the predicate verb. Much remains to be studied about the behavior of adverbs of the class AV2 ("on") which are accepted as floating structures in the current grammar.

The occurrence of a prepositional phrase ("of particles . . . ") between a predicate verb and an object is not uncommon, as in "The author sketches *in the first chapter* an outline of historical and descriptive linguistics.". The interpretation of "formed by" as a post-positional modifier of "particles"—with "formed" as PI1—raises important problems. The current grammar accepts any PI1 as a post-positional modifier if it is followed by PRE. This provision is for structures such as "This is the boy *run over* by a car.", "This is a topic *come across* in various places.". It seems, however, that certain members of PI1 cannot be used as post-positional modifier and that each member of PI1 that can be used as post-positional modifier can be followed only by a limited class of prepositions peculiar to itself. This suggests the necessity for some refinement of verb classification.

Analysis No. 5 (Fig. 18) has a complement clause whose predicate verb "size" governs the object "solutions" which is widely separated from the verb by a long prepositional phrase "of particles . . . purity of". The prepositional phrase has a structure similar to that of "on the principle agreed [intervening prepositional phrase] upon" (see English substitutes in Fig. 18). The asterisk in the column "ENGLISH SUBSTITUTE" indicates that the interpretation given for the corresponding word forms cannot be made semantically acceptable by any English substitutes.

Although Analysis No. 5 can be discarded by any of the techniques proposed in the preceding three paragraphs, its emergence also suggests the necessity for more careful study of floating structures. First, it is possible to establish a prediction of a preposition which cannot accept a floating structure. Such a prediction could be generated after the processing of verbs such as "agreed", "run", "come", and "talked" (all PI1) of "This is a principle *agreed upon* by the people.", "This is the boy *run over* by the car.", "This is a topic *come across* in various places." and "This is a book *talked about* in various circles." respectively, since a floating structure seldom appears between "agreed" and "upon", "run" and "over", and so forth.

The provision would, however, also rule out less frequent structures on the borderline of grammaticality such as "This is the principle *agreed* finally *upon* by the people.", "This is the boy *run* completely *over* by the car.", "This is a topic *come* constantly *across* in various places.", "This is a book *talked* constantly *about* in various circles.". This may or may not be desirable. In any case, although some would agree that the above four sentences with inserted adverbs "finally", "completely", "constantly", and "constantly" are well-formed, even if colloquial and awkward, most would agree that the replacement of each of these adverbs by a longer adverbial phrase would turn the sentences into ill-formed sentences. It would be most unlikely to have sentences such as "This is the principle agreed *finally and unanimously* upon by the people.", or "This is the principle agreed *with no opposition* upon by the people.". The problem here is that the intervening phrases are *too long*.

The criterion of whether an inserted structure is *too long* or *not too long* is quite subjective at this moment. It does not always depend upon the number of words in such a structure, but upon the relative length of the structure in connection with those structures which precede and/or succeed it. Contrast " . . . thereby insuring *against all enemies* the peace and security of . . . " with " . . . thereby insuring *against interference from noise due to excessive crowding of channels* radio astronomy." and with " . . . thereby insuring *against interference from noise due to excessive crowding of channels* not only radio astronomy but also other scientific and communication enterprises that require freedom from interference.".

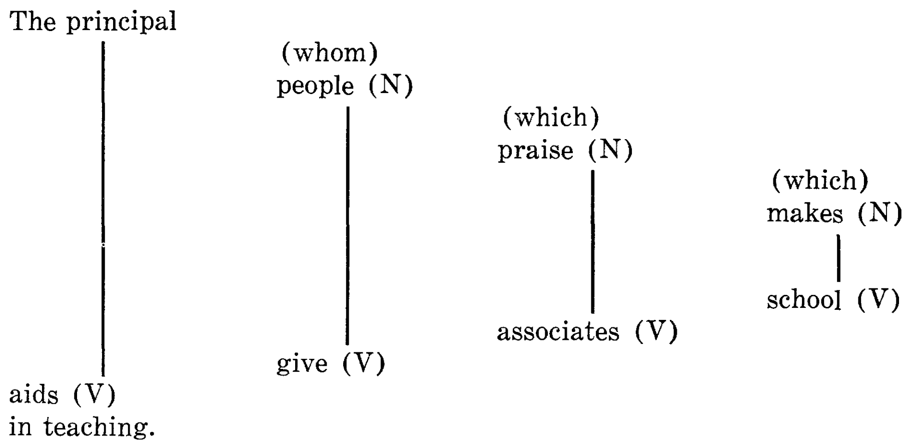
If such a criterion (more or less pertaining to style) could be successfully formalized, the automatic syntactic analysis of languages could be greatly improved.

2.5 The version of the English analyzer (referred to as 1963-FJCC version) used for Sentences 1, 2 and 3 of this section differs from the version (refered to as 1962-IFIP version), described in our previous paper[12, 13], in the following two points: (1) the system has been entirely reprogrammed to attain higher efficiency in program performance, resulting in a speed-up of processing time by a factor of 5 over the 1962-IFIP version; (2) the feature of "droppable" predictions mentioned in Section 2.2 of this paper has been added with an increase in speed by a factor of 2.5. Hence the new version is an order of magnitude faster than the old.

Several other techniques, now being planned for incorporation in the 1963-FJCC version, will eliminate irrelevant paths in syntactic analysis without destroying any paths which may yield acceptable analyses.

(a) Generalized Shaper: At each stage of analysis, the program compares the number of (non-droppable) "comma" predictions and "and" predictions respectively with the number of commas and ands remaining to be processed in the sentence. If the former is greater than the latter, the path is discarded. A similar comparison is to be made between participial predictions and participial word classes. This technique, originated by Plath for Russian,[16] has been experimentally programmed for the 1962-IFIP version where it reduced processing time by a factor of 5. It is expected that this technique, when incorporated in the 1963-FJCC version, will increase the speed by a factor of at least 3. It is yet to be determined where the break-even lies between the time required for making such tests and the time saved by the elimination of irrelevant paths due to such tests.

(b) Self-embedding Test: Independent of the "Nester" test described in Section 2.2, the program checks how many self-embedded structures a given prediction pool contains at each stage of the analysis of a sentence. For initial experiments, any pool which contains more than 3 predicate and clause predictions will be discarded on the assumption that it predicts a structure too deeply self-embedded ever to occur in natural well-formed sentences. This test is expected to be effective especially when a given sentence has a series of contiguous nouns because it would reject the possibility of the first noun being modified by an adjective clause initiated by the second noun (as in "The boy *people* (N) *praise* (V) is . . . ") and the second noun in turn being modified by another adjective clause initiated by the third noun, and so on. For example, this test would accept the syntactically and semantically acceptable interpretation of "The principal people praise makes school associates give aids in teaching." as "The principal (whom) people (N) praise (V) makes (V) school (N) associates (N) give (V) aids (N) in teaching.", while rejecting the interpretation of the same sentence as containing three self-embedded adjective clauses:



The expected processing time of sample sentences by the projected program incorporating these additional features is shown in Fig. 19, together with the actual processing time of the same sentences by the 1962-IFIP version and the 1963-FJCC version.

In addition to these two techniques which are now being programmed, another technique of Plath's for avoiding repetitive local parsings is now being studied for the English analyzer. This technique, already programmed for the Russian analyzer, has proved to be effective for longer sentences by sharply bounding the exponential dependence of processing time on sentence length toward the limiting case of log exp or linear dependence.

3. DIRECTED PRODUCTION ANALYZER AND PHRASE STRUCTURE GENERATORS

3.1 The primary output of a dpa is the sequence of couples (P, c) and of prediction pools. This output specifies the structure of a sentence in a definite and useful way but, standing alone, lacks the intuitive immediacy of the more familiar immediate constituent or dependency tree structural representations. The mapping from this form of output to a more natural tree form, effected by an editing program using the syntactic role indicators, prediction indices, and shifting codes as described and displayed in Section 2, is only one of many possible ones, of which several might well be both more appealing and more useful.

Sentence no.	Sentence Length	No. of Analyses	Sentence	1962-IFIP	1963-FJCC	Expected
				mins	mins	mins
1	17	1	The increase in flow stress was attributed to vacancies, which have appreciable mobility at -72.	0.4	0.0	0.0
2	14	4	Economic studies show that it could be a billion-dollar-a-year business by the 1970's.	0.9	0.0	0.0
3	25	5	Slime formation is dependent on size of particles formed by mechanical means, amount of metal in the amalgam, and purity of solutions.	11.2	1.3	0.3
4	18	1	Single strain reversals at -72 not only produced the N effect but also increased the flow stress.	0.7	0.1	0.0
5	35	12	A shear stress applied during the recovery had no effect on the amount of recovery, if the stress was less than the instantaneous yield point, irrespective of the direction of the stress.	120.0?	10.3	2.0
6	16	40	People who apply for marriage licenses wearing shorts or pedal pushers will be denied licenses.	2.0	0.1	0.0
7	17	4	Nearly all authorities agree that this will be the first practical, large-scale use of space.	0.9	0.0	0.0
8	16	3	A clutch of major companies has been pressing to get such a system into being.	13.5?	0.1	0.0
9	23	7	The U.S. has reached a momentous point of decision in a project that only a few years ago would have seemed improbable.	2.5?	0.2	0.0
10	23	25	Technologically speaking, there are three basic contending schemes, with a number of variations, for orbiting a communication satellite.	22.5	1.5	0.3

Figure 19. Processing Time.

Greibach[8] (Section 3.1) has made it clear that a strictly bi-unique correspondence between an arbitrary psg and an inverse dpa is too much to hope for: every dpa is the inverse of infinitely many psg's and, furthermore, given a psg-dpa pair, it is undecidable in general whether or not the latter is the inverse of the former. She has shown, however (Figure 6 of Greibach[8]; Section 6.2 of Greibach[6]), that the passage from psg to dpa can be restricted so as to proceed in a unique "natural" way.

It follows that, given a psg, a dpa can be constructed which will, together with a mapping of remarkable conceptual simplicity that can be effected with less cumbersome apparatus than that of Section 2, display the structure of a sentence in conventional phrase structure form. There is also some empirical evidence to the effect that, given a dpa with shifting codes, etc., a psg can be constructed from it which, when converted to standard form in the "natural" way, yields something close to the original dpa. Hence there is some hope that, although the mapping from dpa to psg is not abstractly single valued, the loop of Fig. 6 of Greibach[8] might at least be closed in a unique self-consistent way.

Consider the directed productions

$$(\text{S, art}) \rightarrow \text{art SP}' \text{ VP PD } (SV) \quad (1)$$
$$(\text{SP}', \text{nnn}) \rightarrow \text{nnn } (SV) \quad (2)$$
$$(\text{VP, vi}) \rightarrow \text{vi } (PV) \quad (3)$$
$$(\text{PD, prd}) \rightarrow \text{prd } (ES) \quad (4)$$

where "art" stands for an article, "SP′" for a subject phrase modified by an adjective, and "VP" and "PD" for a predicate and a period, respectively. "*SV*" is a role indicator for the subject of a verb, "PV" for a predicate verb, and "*ES*" for an end-of-sentence mark. These productions are sufficient for the obvious analysis of the sentence "The summer came.".

The psg productions

$$\text{S} \rightarrow \text{SP VP PD} \quad (5)$$
$$\text{SP} \rightarrow \text{T SP}' \quad (6)$$
$$\text{T} \rightarrow \text{art} \quad (7)$$
$$\text{SP}' \rightarrow \text{nnn} \quad (8)$$
$$\text{VP} \rightarrow \text{vi} \quad (9)$$
$$\text{PD} \rightarrow \text{prd} \quad (10)$$

are adequate to generate the same sentence with tree structure as in Fig. 20.

Productions (5)-(7) of the psg correspond to production (1) of the dpa, in the manner de-

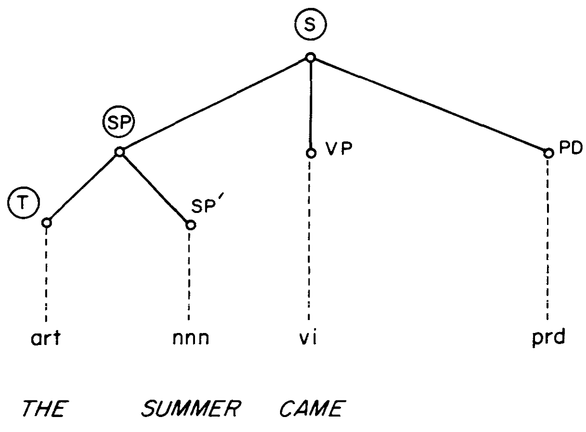


Figure 20. Tree Structure for "The summer came."

scribed by Greibach[8] (Figs. 1, 2 and 3). It is the absence from (1) of the circled symbols of Fig. 20, which we shall call *virtual* predictions, which in a sense differentiate the dpa from the psg. When "the" as "art" is accepted by the rule (1), the predictions S, SP and T are virtually fulfilled in whole ("T") or in part ("S", "SP"). The fact that T is a constituent of SP is essentially what is denoted by the role indicator (*SV*) in (1). Since SP′ is also a constituent of SP it is at a lower level than "VP" and "PD", although it appears undistinguished from the latter in (1). It is this disparity which is corrected by the shifting codes associated with dpa productions of the actual English grammar. With the fresh insight yielded by Greibach's theoretical results it appears possible, if desirable, to dispense with the *ad hoc* tree mapping apparatus built into the editing program in favor of more natural and elegant techniques.

These techniques are based on a new extension of parenthesis-free or Polish prefix notation in which predictions are treated as functors which, unlike conventional functors, do not have a fixed degree, but instead are explicitly labelled with a degree determined by the actual or virtual production by which they are expanded.

3.2 Consider the following augmented directed production as a replacement for production (1).

$$(\mathrm{S}, \mathrm{art}) \rightarrow \mathrm{S}_3\ \mathrm{SP}_2\ \mathrm{T}_1\ \mathrm{art}\ \mathrm{SP}'\ \mathrm{VP}\ \mathrm{PD}. \qquad (11)$$

In (11), subscripted expressions are interpreted as functors of degree specified by their subscripts. All other expressions are interpreted as variables (functors of degree 0). It is an immediate consequence of this interpretation that the right-hand side of any production written in this form is itself a well-formed string in parenthesis-free notation. Hence a grammar of this type would lend itself to mechanical checks for the well-formation of its rules, a property of considerable practical importance, say, in verifying the key-punching of a large grammar table.

If the subscripted expressions in (11) are ignored, (11) corresponds directly to (1). The subscripted expressions may, however, also be identified with the virtual predictions of Fig. 20. "T", as a functor of degree 1, has argument "art"; "SP", of degree 2, has as arguments the well-formed formulas "T_1 art" and "SP′", and the three arguments of S_3 are the well-formed formulas "SP_2 T_1 art SP′", "VP", and "PD". Any functor whose scope includes only subscripted expressions and terminal symbols corresponds to a wholly fulfilled virtual prediction (e.g., "T_1"), otherwise to a partially fulfilled one (e.g., "SP_2").

The production (11) ascribes degree 3 to "S". Other productions need not ascribe the same degree. Thus, in

$$(\mathrm{S}, \mathrm{ii}) \rightarrow \mathrm{S}_2\ \mathrm{VP}_1\ \mathrm{ii}\ \mathrm{PD} \qquad (12)$$

"S" is ascribed degree 2. The terminal symbol "ii" stands for the infinite form of an intransitive verb, and (12) accounts for structures such as "Go.".

3.3 It is obvious how to get augmented directed productions of the form (11) or (12) from the phrase structure tree of any sentence, since that tree is always finite. However, the productions of an arbitrary psg may provide for infinite left-branching structures (e.g., X → XY), hence more subtle difficulties arise when mapping the psg into a psg in standard form because the application of *every* production of such a psg must yield a terminal symbol.

Greibach's normal form theorem not only shows that such provisions can be made effectively for an arbitrary psg but it also implicitly converts left-branching structures into right-branching ones by eliminating such productions as $X \rightarrow XY$ while creating or retaining others of forms such as $X \rightarrow aXZ$ (Fig. 5 of Greibach[8]). As a consequence, and so far as phrase structure grammars are concerned, the direction of branching is shown to be not so much an intrinsic property of a language as a property of a grammar describing the language although the freedom of self-embedding is preserved. In fact, even a language so inherently "left-to-right" in appearance as parenthesis-free notation itself can be generated, hence analyzed, entirely in a right-to-left mode! In view, however, of the fact that English is written and read from left-to-right, of the desirability of generating (analyzing) a terminal symbol each time a production is applied, and of Yngve's[20] arguments about the desirability of limited left-branching, the psg in standard form and the corresponding dpa suggest themselves as potential mechanisms for speakers and hearers respectively, and hence as worthy objects of further study by psychologists and linguists. The authors are deeply impressed with the simplicity and elegance of the corresponding machine realization of such grammars but this, of course, is in itself no argument at all in favor of their adoption as explanatory models for human synthesis and analysis of sentences without some careful experimentation. It should go without saying that if transformations in the sense of Chomsky[1] are to be applied to any given sentence, the phrase markers for the sentence must be at hand. The realization of a dpa is therefore an essential prerequisite for the effective application of transformational grammars to sentence analysis.

As pointed out in Section 3.1, going from a dpa to a psg (other than the obvious but non-intuitive standard form inverse) is not a simple matter, and considerable theoretical and experimental work remains to be done. The available structure symbols and shift codes do appear to lead readily to the conversion of the current dpa grammar to one whose rules are augmented directed productions. Whether the resulting psg can, using Greibach's normal form theorem, be reconverted to the current dpa, thereby closing the loop of Fig. 6 (Greibach[8]), remains to be seen, but there is some ground for optimism at present.

3.4 Quite fortunately, the analysis program for a system based on augmented directed productions can be precisely that for the present one except that the former requires two prediction pools instead of one. The first pool is used for storing fulfilled (subscripted) predictions and terminal symbols[§], the second for storing unfulfilled (and therefore active) predictions. Each time the topmost prediction in the active pool is processed against a word class of the next word, the subscripted predictions and the terminal symbol of the subrule are stored in the fulfilled pool in the same order as they appear in the formula. Remaining active (non-subscripted) predictions are stored in the active pool. The performance of these two pools is illustrated below using "The man saw the boy." as an example.

For the sake of simplicity of explanation, only the path which leads to the acceptable analysis of this sentence is followed here. Augmented directed productions which are needed for this path are:

Note

vt:	transitive verb
V:	verb prediction
OP:	object phrase prediction
OP′:	modified object phrase prediction
prd:	period

$$(\text{S, art}) \rightarrow \text{S}_3\ \text{SP}_2\ \text{T}_1\ \text{art SP}'\ \text{VP PD} \qquad (13)$$
$$(\text{SP}', \text{nnn}) \rightarrow \text{SP}'_1\ \text{nnn} \qquad (14)$$
$$(\text{VP, vt}) \rightarrow \text{VP}_2\ \text{V}_1\ \text{vt OP} \qquad (15)$$
$$(\text{OP, art}) \rightarrow \text{OP}_2\ \text{T}_1\ \text{art OP}' \qquad (16)$$
$$(\text{OP}', \text{nnn}) \rightarrow \text{OP}'_1\ \text{nnn} \qquad (17)$$
$$(\text{PD, prd}) \rightarrow \text{PD}_1\ \text{prd} \qquad (18)$$

Figure 21 shows the status of the two pools after the processing of each word of the sentence. Initially, the fulfilled pool is empty, and the active pool contains the initial symbol S. The second line shows that after the processing of "the" as art, "$\text{S}_3\ \text{SP}_2\ \text{T}_1$ art" of (13) have been stored in the fulfilled pool, and that "SP′

§ Formally, the first pool is simply an output tape with writing-head only and not a pushdown store, since nothing is read from it in the course of further analysis.

Fulfilled Pool bottom ⟷ top	Active Pool top ⟷ bottom
	S
S_3 SP_2 T_1 art	SP' VP PD
S_3 SP_2 T_1 art SP'_1 nnn	VP PD
S_3 SP_2 T_1 art SP'_1 nnn VP_2 V_1 vt	OP PD
S_3 SP_2 T_1 art SP'_1 nnn VP_2 V_1 vt OP_2 T_1 art	OP' PD
S_3 SP_2 T_1 art SP'_1 nnn VP_2 V_1 vt OP_2 T_1 art OP'_1 nnn	PD
S_3 SP_2 T_1 art SP'_1 nnn VP_2 V_1 vt OP_2 T_1 art OP'_1 nnn PD_1 prd	

Figure 21. Analysis of "The man saw the girl."

VP PD" have been stored in the active pool, replacing the previous topmost prediction S. The series of symbols contained in the fulfilled pool in the last line of Fig. 21 is the output of the analysis of this sentence.

In this proposed system, discontinuous structures such as "*It* is true *that he is right.*" can probably be treated in the same way as in the current analyzer; a production for (S, it) will be

$$(\text{S, it}) \rightarrow \text{S}_4\,\text{SP}_1\ \text{it VP NC PD} \qquad (19)$$

where "it" stands for a temporary subject, and NC for a noun clause which is the true subject of the sentence. If it is desired that the connection between SP and NC be explicitly identified, it is possible to assign special marks to these predictions showing that they constitute a single (discontinuous) structure.

Adverbs, prepositional phrases, etc. will be accepted as floating structures and their symbols ignored when checking for well-formation of a production as a whole. For example,

$$(\text{VP, adv}) \rightarrow \text{ADV}_1^{*}\ \text{adv VP} \qquad (20)$$

indicates that, although an adv is joined to the structure ADV of degree 1, ADV itself is outside the structure of VP, with its dependency undetermined.

3.5 Since the analyzer output is in parenthesis-free form, it can be interpreted as a phrase-structure tree without further formal ado, although, if a more graphic tree form is desired, additional editing is obviously possible.

The output of such an analyzer has an abvious kinship to that of Yngve's[2] model of random sentence production. There are, however, significant differences beyond the obvious one that it is easier to describe the structure of a sentence being synthesized than that of one being analyzed.

In Yngve's model, no degrees are assigned to non-terminal symbols such as "S", "NP", "VP", etc. The difference is non-trivial. There is the bonus of a mechanical check for well-formation of rules, not a negligible factor in major clerical enterprises. More important, however, there is the fact that Yngve's output formulas become ambiguous whenever rules are included in the grammar of which the left-hand symbol is the same but the number of right-hand symbols is different. In order to make Yngve's output unambiguous, SP of SP → T SP′ has to be distinguished from SP of SP → nnn. This would entail duplication of rules such as S → SP VP PD: one rule for SP with degree 1, the second rule for SP with degree 2. The proposed technique for specifying degrees for fulfilled predictions resolves this dilemma.

REFERENCES

1. CHOMSKY, N., "Three Models for the Description of Language," *IRE Trans. on Information Theory,* Vol. IT-2, 1956.
2. CHOMSKY, N., "Formal Properties of Grammars," in R. R. Bush, E. H. Galanter, and R. D. Luce (Eds.), *Handbook of Mathematical Psychology,* Vol. 2, New York, Wiley, 1962.
3. CHOMSKY, N., and SCHÜTZENBERGER, M. P., "The Algebraic Theory of Context Free Languages," in P. Braffort and D. Hirschberg (Eds.), *Computer Programming and Formal System,* North Holland Publishing Co., Amsterdam, 1963.
4. EVEY, J., "The Theory and Application of Pushdown Store Machines" (Doctoral Thesis), *Math. Ling. and Automatic Translation,* Report No. NSF-10, Harvard Computation Laboratory, 1963.
5. EVEY, J., "Application of Pushdown Store Machines," *AFIPS Proceedings of 1963 Fall Joint Computer Conference,* Las Vegas, Nevada, November 12-14, 1963.
6. GREIBACH, S., "Inverses of Phrase Structure Generators" (Doctoral Thesis), *Math. Ling. and Automatic Translation,* Report

No. NSF-11, Harvard Computation Laboratory, 1963.

7. GREIBACH, S., "The Undecidability of the Ambiguity Problem for Minimal Linear Grammars," *Information and Control,* Vol. 6, No. 2, 1963.
8. GREIBACH, S., "Formal Parsing Systems" (Draft report), submitted for publication, 1963.
9. HAYS, D. G., "On the Value of Dependency Connection," *Proc. of the 1961 International Conference on Machine Translation of Languages and Applied Language Analysis,* her majesty's Stationery Office, London, 1962.
10. KUNO, S., "The Current Grammar for the Multiple-path English Analyzer," *Math. Ling. and Automatic Translation,* Report No. NSF-8, Harvard Computation Laboratory, 1963.
11. KUNO, S., "The Multiple-path Syntactic Analyzer for English," and "Study of Analysis Output of the Multiple-path English Analyzer," *Math. Ling. and Automatic Translation,* Report No. NSF-9, Harvard Computation Laboratory, 1963.
12. KUNO, S., and OETTINGER, A. G., "Multiple-path Syntactic Analyzer," *Information Processing 62,* North-Holland Publishing Co., Amsterdam, 1963.
13. KUNO, S., and OETTINGER, A. G., "Multiple-path Syntactic Analyzer" (updated version), *Math. Ling. and Automatic Translation,* Report No. NSF-8, Harvard Computation Laboratory, 1963.
14. MATTHEWS, G. H., "Analysis by Synthesis of Natural Languages," *Proc. of 1961 International Conference on Machine Translation of Languages and Applied Language Analysis,* her majesty's Stationery Office, London, 1962.
15. OETTINGER, A. G., "Automatic Syntactic Analysis and the Pushdown Store," *Proc. of Symposia in Applied Math.,* Vol. XII, American Mathematical Society, Providence, Rhode Island, 1961.
16. PLATH, W., "Multiple-path Syntactic Analyzer of Russian," *Math. Ling. and Automatic Translation,* Report No. NSF-12, Harvard Computation Laboratory, 1963.
17. RHODES, I., "A New Approach to the Mechanical Translation of Russian," *National Bureau of Standards Report,* 1959, No. 6295, also in *Mechanical Translation,* Vol. 6, 1961.
18. ROBINSON, J. J., "Preliminary Codes and Rules for the Automatic Parsing of English," *Memorandum RM-339-PR,* The RAND Corporation.
19. SAKAI, I., "Syntax in Universal Translation," *Proc. of the 1961 International Conference on Machine Translation of Languages and Applied Language Analysis,* her majesty's Stationary Office, London, 1962.
20. YNGVE, V., "A Model and an Hypothesis for Language Structure," *Proc. of the American Philosophical Society,* Philadelphia, 1960.

A TAPE DICTIONARY FOR LINGUISTIC EXPERIMENTS*

*J. L. Dolby, H. L. Resnikoff, and E. MacMurray***

INTRODUCTION

Almost since the arrival of the first piece of modern computing equipment there has been considerable interest in the potential of this equipment for various aspects of linguistic data handling. Experiments in machine translation date to at least the late forties and more recently experiments in indexing and abstracting are becoming more widespread. Almost without exception these experiments have shown that the basic problems are not trivial and that serious experimentation on an extensive level will be necessary before any of these problems can be conquered—even assuming that solutions of some sort can be found. As a result there has been an increasing amount of speculation about the fundamental nature of linguistic structure and a growing need for fundamental data on which various conjectures about this structure can be tested.

Recently several word lists and dictionaries have come into being to help meet this need. The University of Pennsylvania[1] has compiled a large word list consisting of the union of the non-obsolete words in the Second Edition of Webster's New International Dictionary and four technical dictionaries. As with most of the lists mentioned here, the Pennsylvania list is available in alphabetic order and reverse alphabetic order (that is, the words are arranged alphabetically beginning with the last rather than the first letter). Cornell University has produced a tape dictionary on written and phonetic forms consisting of an updated version of the 20,000 most used words from the *Thorndike Senior Century Dictionary.* Other efforts are known to be underway at Indiana University[2] and Standford University. Further, a number of study groups have prepared small dictionaries containing syntactic information (e.g., the work at Harvard[3]) but these have generally been set up to handle specific texts and expanded only as the need arises.

Early experiments at Lockheed on automatic parsing (Earl, Ricklefs, and Robison[4]) demonstrated that much progress could be made in this area and that a complete dictionary or some combination of a dictionary and a part-of-speech algorithm would be necessary as input to the parsing program. As a result, it was decided to key punch the words of the *Pocket Oxford Dictionary* together with the parts of speech given there in order to determine the extent to which a part-of-speech dictionary could be replaced by a small dictionary and a part-of-speech algorithm. It was found (Resnikoff and Dolby[5]) that it was possible to construct an algorithm that would rival the accuracy of the *Pocket Oxford* when used in conjunction with a small (200 word) dictionary. However, it was also found that neither the algorithm nor the *Pocket Oxford* provided sufficiently accurate and detailed information for the parsing studies.

Concurrently, research on the nature of word breaking, or graphemic syllabification (Dolby

*This work is supported by the Lockheed Independent Research Program.

**Mr. Dolby is with the Lockheed Missiles and Space Company, 3251 Hanover St., Palo Alto, California. Mr. Resnikoff is presently at the University of Munich, Germany, on leave from Lockheed and Mr. MacMurray is with the Statistical Tabulating Corporation. San Francisco.

and Resnikoff[6]), showed that it would be highly desirable to construct an extensive list of words showing the legitimate break positions (by dictionary standards) together with major and minor stress positions. Further, when it became evident that graphemic syllabification was intimately connected with the algorithmic determination of parts of speech (Dolby and Resnikoff[7]) it was obvious that these two types of information should be combined into a single dictionary available for further computer experiments.

In this paper we discuss the sources of information used to construct such a dictionary, the problems in attaining a reasonable degree of accuracy and consistency in such a task and the various output forms that have proven to be useful in this research.

Dictionary Sources

As a first step in the construction of this dictionary, all of the one-syllable words of the *Oxford Universal Dictionary* were studied in considerable detail.[7] It was found that the dictionary part-of-speech designations for these words varied from one source to another but that when *all* sources were considered the designations were quite consistent. By "all" in this case, we mean the parts of speech given in both the 2nd and 3rd editions of *Webster's New International,* the *Oxford Universal Dictionary* and the *Oxford English Dictionary.* It was also found that the consistency increased markedly as the words considered were narrowed down to what we might term the standard words of the language. Thus it was evident that it would be necessary to merge the information from at least two sources to obtain a measure of dictionary consistency and that some information as to the relative utility of the word would also be needed.

As a first approximation, it was decided to obtain all of the left-justified, bold-faced words of the *Shorter Oxford* together with the parts-of-speech given by that dictionary and to add to this the parts of speech for these words found in the 3rd edition of *Webster's New International Dictionary.* In addition, a code was supplied for each part of speech given in each source to designate the status given. If any standard meaning was to be found in that source for that part of speech the word was called standard for that part of speech. Otherwise, the first non-standard designation (such as obsolete, dialectical, archaic, etc.) was used. Figure 1 illustrates a typical page of material as entered by the keypunch operators. Table 1 provides a list of codes used.

TABLE 1
Status Codes

Dictionary Form	Keypunch Code	Edited Code
Dialectical	DI	D
Alien	AL	F
Archaic	AR	A
Colloquial	CO	Q
Capital	CA	C
Erroneous	ER	E
Nonsense	NS	N
Nonce Word	NW	W
Obsolete	OB	O
Poetical	PO	P
Rare	RA	R
Rhetoric	RH	H
Specialized	SP	$
Standard	ST	S
Substandard	SS	Z

Part-of-Speech Codes

1	Noun
2	Adjective
3	Verb
4	Adverb
5	Preposition
6	Conjunction
7	Pronoun
8	Interjection
9	Past
0	Other

The second stage of the operation consisted of a relatively simple editing operation designed to put the information in more usable form. This consisted of several steps. First, the coding scheme originally adopted was designed primarily to simplify the keypunching operation in the interest of minimizing keypunch errors. As a result, a three letter code was used for each part-of-speech, status entry. The overall format was such as to restrict the total

	Part-of-speech and Status Codes		Sequence
Main entry	Shorter Oxford	Webster's Third	Number
CAUSE	1ST3ST6DI	1ST3ST6ST	1107020
CAUSE CELEBRE	1AL	1ST	1107030
CAUSELESS	2ST	2ST	1107040
CAUSERIE	1AL	1ST	1107050
CAUSEUSE	1AL	1ST	1107060
CAUSEWAY	1ST	1ST3ST	1107070
CAUSEY	1ST3DI	1ST3DI	1107080
CAUSIDICAL	2ST		1107090
CAUSON	1OB		1107100
CAUSTIC	1SP2ST	1ST2ST	1107110
CAUSTICITY	1ST	1ST	1107120
CAUTEL	1OB	1OB	1107130
CAUTER	1ST	1ST	1107140
CAUTERANT	1ST	1ST2ST	1107150
CAUTERISM	1OB		1107160
CAUTERIZE	3ST	3ST	1107170
CAUTERY	1ST	1ST	1107180
CAUTION	1ST3ST	1ST3ST	1107190
CAUTIONARY	1ST2ST	1ST2ST	1107200
CAUTIOUS	2ST	2ST	1107210
CAVA	1AL	1ST0ST	1107220
CAVALCADE	1ST3ST	1ST3ST	1107230
CAVALIER	1ST2ST	1ST2ST3ST	1107240
CAVALLY	1ST		1107250
CAVALRY	1ST2ST	1ST2ST	1107260
CAVATINA	1AL	1ST	1107270
CAVE	1ST2OB3ST8AL	1ST2ST3ST	1107280
CAVEAT	1ST3OB	1ST3ST	1107290
CAVEL	1DI3DI	1DI3DI	1107300
CAVENDISH	1ST	1ST	1107310
CAVERN	1ST3ST	1ST2ST3ST	1107320
CAVERNOUS	2ST	2ST	1107330

Figure 1

number of such entries for one word in one dictionary to seven. A few words (such as *a*) had more than seven parts of speech. These added entries were provided on a second card. In the editing run these card-pairs were reduced to a single record on the tape. At the same time, the part-of-speech, status field was reduced to ten columns for each dictionary by representing the part-of-speech by column position and by using a single literal code for the status. Further, the word field was reduced to twenty columns since inspection of the original printout showed that only five words of the entire 73,000 exceeded that number of letters.

In addition, it was possible to add certain derived information to the record. A second word field of twenty columns was added to store the reversed form of the word in left justified position to enable us to obtain "backwards" orderings with standard sorting routines. The backwards ordering is useful in a number of ways, the most important being that it permits us to obtain all the words with the same ending (e.g., the same suffix). Another column was devoted to the coding of the status of the word (as opposed to the status of its meanings given in the part-of-speech fields). The code adopted was *b* if there was at least one standard part-of-speech code in both of the dictionary part-of-speech fields, *x* if only the *Shorter Oxford* provided a standard meaning for the word, *w* if only *Webster's* provided a standard meaning for the word and a blank if neither source provided such a meaning.

Yet another column was devoted to recording a numerical code to approximate the number of syllables in the word since, as already noted, there is now known to be an intimate connection between the number of syllables in the word and its part of speech assignments. The approximation to this was obtained by counting the number of vowel strings in the word entry (but not counting a final *e* as a vowel). The same column was also used to provide an overriding code for prefixes, suffixes, hyhenated words and broken words (the codes being *p, s, h* and *b* respectively). This latter determination was made by quite obvious use of the presence of hyphens at the beginning, end or intermediary portions of the words and by the presence of an intermediary blank in the word.

Finally, a new field of ten columns was devoted to obtaining a merged part-of-speech,

```
                                        Syllable Count
                                        Status    Part-of-speech & Status
Main Entry          Reversed Entry          Shorter                             Sequence #
                                            Oxford   Websters  Merged

CAUSE               ESUAC               1BS S  D  S S  S    S S  S    1107020
CAUSE CELEBRE       ERBELEC ESUAC       BWF       S         S         1107030
CAUSELESS           SSELESUAC           3B S       S         S        1107040
CAUSERIE            EIRESUAC            4WF       S         S         1107050
CAUSEUSE            ESUESUAC            3WF       S         S         1107060
CAUSEWAY            YAWESUAC            4BS       S S       S S       1107070
CAUSEY              YESUAC              2BS D     S D       S D       1107080
CAUSIDICAL          LACIDISUAC          4X S                 S        1107090
CAUSON              NOSUAC              2 O                 O         1107100
CAUSTIC             CITSUAC             2B$S      SS        SS        1107110
CAUSTICITY          YTICITSUAC          4BS       S         S         1107120
CAUTEL              LETUAC              2 O       O         O         1107130
CAUTER              RETUAC              2BS       S         S         1107140
CAUTERANT           TNARETUAC           3BS       SS        SS        1107150
CAUTERISM           MSIRETUAC           3 O                 O         1107160
CAUTERIZE           EZIRETUAC           3B  S       S         S       1107170
CAUTERY             YRETUAC             3BS       S         S         1107180
CAUTION             NOITUAC             2BS S     S S       S S       1107190
CAUTIONARY          YRANOITUAC          4BSS      SS        SS        1107200
CAUTIOUS            SUOITUAC            4B S       S         S        1107210
CAVA                AVAC                2WF       S        SS        S1107220
CAVALCADE           EDACLAVAC           3BS S     S S       S S       1107230
CAVALIER            REILAVAC            3BSS      SSS       SSS       1107240
CAVALLY             YLLAVAC             3XS                 S         1107250
CAVALRY             YRLAVAC             3BSS      SS        SS        1107260
CAVATINA            ANITAVAC            4WF       S         S         1107270
CAVE                EVAC                1BSOS  F  SSS       SSS    F  1107280
CAVEAT              TAEVAC              2BS O     S S       S S       1107290
CAVEL               LEVAC               2 D D     D D       D D       1107300
CAVENDISH           HSIDNEVAC           3BS       S         S         1107310
CAVERN              NREVAC              2BS S     SSS       SSS       1107320
CAVERNOUS           SUONREVAC           3B S       S         S        1107330
```

Figure 2

status field derived from the information given by each dictionary. This again was accomplished in a rather obvious manner giving précedence to standard meanings or the first meaning given (in this case, the one given by the *Shorter Oxford*) whenever the two sources did not agree.

The resulting tape record was thus again expanded to a full eighty columns as is shown in the sample page given in Figure 2. However, in this form it is now possible to sort the information in a number of ways by standard sort routines. Generally, the most useful sort pattern is to provide a major sort on the "number of vowel string" code followed by a minor sort on either forward alphabetic or reverse alphabetic order.

Additions to the Original Source Material

Our original choice of the *Shorter Oxford* as a word list was made because it provided a list of manageable size with excellent part-of-speech information. However, there remained some question in our minds as to the extent of coverage of this source for common American usage. Fortunately, R. L. Venezky was kind enough to make available to us the Cornell University tape of 20,000 commonly used words. All of the words in this list that were not contained in the *Shorter Oxford* were added to original source together with the part-of-speech information for these words provided by *Webster's 3rd*. There were 2490 of these words, incidentally.

The next main step in this area is still in progress. As we have already noted, graphemic syllabification is of considerable interest to us. We have therefore undertaken to punch all of the words of Funk and Wagnall's *New Practical Standard Dictionary* in the syllabified form given by that source together with the accent information there given. The choice of source for this information stemmed from the fact that this was one of the few dictionaries to provide this information on the derived forms in a reasonable manner for direct keypunching. When this list is complete, the syllabification information will be added to the existing information for the words in the original source. (It is not contemplated at this time that all of the words in the Funk and Wagnall's list will be added to the list since it now appears that some 50,000 added entries would be necessary.) When this is done, the present "vowel-string approximation" will be replaced by the actual Funk and Wagnall's syllable count. (It will, however, be of interest to study precisely where the two differ.) Another check of some interest will be the comparison of the Thorndike usage factors given in the Cornell list with the rather crude status designation we have derived on the basis of standard word designations given by our two sources of part-of-speech information.

Accuracy and Consistency

A brief study of the structure of almost any dictionary is sufficient to show that dictionaries are constructed primarily for human, as opposed to machine usage. As a result, a number of conventions had to be established in order to provide a reasonable degree of consistency in the coding. For example, some five different forms were used by the *Shorter Oxford* to indicate obsolete words (with a number of variations within one of these forms). Further, we found it very difficult to determine in advance a precise set of rules that would resolve all the questions that were to arise. As a result, a rough structure of the coding was laid out in advance and this was filled in, in detail, as problems came up. A careful log of all decisions was maintained throughout so that once a form was encountered it could be checked against the log to see if a decision had been made as to how it should be coded.

The generation of the main list was verified completely with additional spot checks throughout the project. The key punching was also verified one hundred percent. A random sample of 280 words was then chosen from the entire list and these were checked and no errors were found. In addition, the symmetric difference between the main list and the Cornell list was checked closely to determine whether any of the words in either difference set could be there due to an error in either list.

Users of the list have been kind enough to report errors as they are found. The spelling research group at Stanford, for instance, has made an intensive study of a random sample of 1500 words from the main list and they report that no spelling errors were found in that sample.

SUMMARY

A tape dictionary of some 75,000 entries has been prepared with part-of-speech, status, usage, graphemic syllabification and stress information. The entries have been sorted alphabetically forward and backward as well as by syllable and by part-of-speech. Comparisons are being drawn between various measures of usage as well as between two measures of the number of syllables in the written form. Considerable care has been taken to minimize the number of errors in the list and to insure a high degree of consistency in the coding. The authors believe that the resulting listing will be of great utility in basic studies of the nature of linguistic data handling.

REFERENCES

1. *Normal and Reverse English Word List* (8 Volumes). Compiled under the direction of A. F. BROWN, University of Pennsylvania (1963).
2. ASTIA Report AD 273 500. Seventh Quarterly Report on Automatic Language Analysis.
3. Mathematical Linguistics and Automatic Translation. Report No. NSF–8. Harvard University, Cambridge, Massachusetts. January 1963.
4. "Automatic Syntactic Analysis of Simple English Sentences." L. L. EARL, B. RICKLEFS, H. R. ROBISON. Lockheed Document 6–90–61–40. July 31, 1961.
5. RESNIKOFF, H. L., and DOLBY, J. L., Correspondence of the July 1963 *Proceedings of the I.E.E.E.*
6. "On the Structure of Graphemic Syllabification," J. L. DOLBY and H. L. RESNIKOFF. Presented at the August 1963 Meeting of the Association for Machine Translation and Computational Linguistics, Denver. (Abstract appears in *Mechanical Translation,* August 1963.)
7. "Prolegomena to a Study of Written English," J. L. DOLBY and H. L. RESINKOFF. Lockheed Document 6–90–63–5. February 1963.

HYBRID SIMULATION OF AN AIRCRAFT ADAPTIVE CONTROL SYSTEM

Peter W. Halbert
Applications Engineer, Electronic Associates, Inc.
Princeton, New Jersey

INTRODUCTION

The Adaptive Control Problem

The general concept of adaptive control has become well known, and the objectives and various types of adaptive systems have been classified.[1] In particular, for adaptive control of a high performance aircraft, it is desired to alter the feedback control parameters in the basic linear control system to correct for unknown, unanticipated, or unaccounted-for changes in the aircraft's operating characteristics, inputs, or criterion of performance.

A typical adaptive scheme to accomplish this purpose measures and evaluates the aircraft's performance over some recent time interval, makes decisions regarding any necessary control loop parameter modifications, and implements these changes.

Of interest is the speed of adaptation. The time required in the adaptive process above (for a reasonable quality of adaptation) is usually much longer than the characteristic time constant of the system.[2] It is suggested here that faster adaptations would obtain if the evaluation of system performance were made on the basis of *predicted* future aircraft performance as well as *measured* past performance.

Accordingly, an adaptive scheme is proposed wherein environmental changes are reflected in an on-board model of the aircraft control system. The model, in turn, is used to predict future aircraft performance and, in conjunction with an optimization program, to determine optimum controller parameters. Prediction is done via high-speed integration of the model equations; therefore, optimization is rapid, and a new set of controller parameters can be instituted almost as soon as a measurement of performance indicates a need for change.

The feasibility of using a predictive technique of this sort is to be determined by means of a hybrid simulation. Of particular interest is the quality of adaptation, i.e., the magnitude of departures from optimum performance in the face of environmental upsets. Since a hybrid approach is applicable to problems requiring both logical calculations and high-speed iterative solution of differential equations, its use is justified in this problem, where these requirements are imposed by the optimization technique.

Hybrid System

One of the major forms of hybrid computation to emerge in recent years is the association of a general purpose analog computer with a complement of general purpose digital logic, memory, and conversion components. The genesis of this type of computational system began with a desire to exploit the high-speed integration capability of the analog computer. The consequent need to control and change the analog's program on the basis of previous results and external inputs and at correspondingly high speed led to the adoption of the parallel digital devices, programmed to exercise the necessary logical control, timing, and data storage functions.

Applications of this type of hybrid computation include iterative calculations, automatic production of a sequence of analog runs, and solutions of partial differential equations via techniques requiring function storage and iteration.[6] The additional capabilities of this system to perform automatic optimization of system parameters and to simulate systems containing digital devices are of interest in this problem.

The adaptive control problem imposes computational requirements of two-speed integration of the system equations with iterations of the high-speed solution under direction of the digital logic elements. These latter are also programmed to automatically test the results of analog iterations, make logical decisions from evaluations of these results, and execute commands aimed at determining optimum system parameters. The analog computer provides the simulation of the aircraft and control system, their models, and also of a dynamic reference model which provides a criterion for evaluating system performance. The parallel digital elements, in addition to their role in directing the optimization, are programmed to act as master control of the entire simulation. With the exception of the representation of the actual aircraft and control system, the simulation is regarded as one of a potential on-board adaptive control system. The simulation itself is considered to be representative of those in which the full speed capability of the analog computer is utilized.

Adaptive Control System

The Overall System

The elements essential to the adaptive technique are shown in Figure 1 and comprise:

1. an Aircraft Control System whose feedback control parameters $\overline{\lambda}$ (vector nomenclature) are the objects of adaptation.
2. a Performance Criterion in the form of a dynamic reference model (with parameters $\overline{q}$), arbitrarily specified but known to produce a desirable response $\overline{x}_{RM}$ (t) to pilot inputs i_p (t).
3. Auxiliary Models (heavy black boxes in Figure 1), having parameters $\overline{\lambda}_M$, $\overline{p}_M$, and q, corresponding to actual system parameters, to be used to calculate (via high-speed integration) the predicted responses $\overline{x}_M$ (t) and $\overline{x}_{RM}$ (t) for use in the optimization program.
4. a Predicted Index of Performance, E, defined by
$$E = \int_{t}^{t+\tau} \left| \overline{x}_{RM} - \overline{x}_M \right| dt \qquad (1)$$
where τ is the prediction interval.
5. Adaptive Process No. 1, an automatic optimization program, which computes $\delta E/\delta\overline{\lambda}_M$ and the optimum $\overline{\lambda}_M$.
6. Adaptive Process No. 2, a parameter tracking program, which calculates optimum $\overline{p}_M$ from measurements of actual system performance.

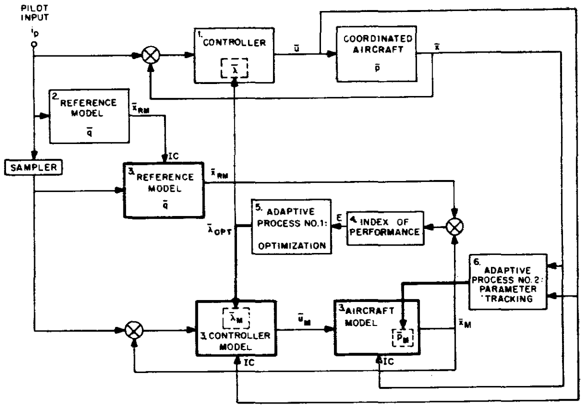


Figure 1. Adaptive Control Scheme.

The form of the Performance Criterion and the use of Auxiliary Models are suggested by the "Model -Reference-Adaptation" technique of Whitaker[3] and an "auxiliary system" concept of adaptation as discussed by Widrow[2], respectively. The approach taken here, however, is different from both of these in its utilization of high-speed prediction.

The predictive technique operates as follows. Adaptive Process No. 1:

1. makes small perturbations in λ_{Mi} (each component of $\overline{\lambda}_M$).

2. measures E over a prediction interval with the perturbed λ_{Mi}.
3. calculates $\delta E/\delta\overline{\lambda}_M$ (approximately).
4. alters $\overline{\lambda}_M$ so that E will be reduced.

Successful modifications of $\overline{\lambda}_M$ initiates identical modification in $\overline{\lambda}$ (via lighter line in Figure 1). High-speed repetition of this process produces the optimum $\overline{\lambda}$ in a very short time interval (typically much less than one second for a system whose major time constant is about three seconds).

By itself, adaptive loop no. 1 does not provide complete adaptive control. A second adaptive process is necessary to track the parameters $\overline{p}_M$, updating the predictive model in an attempt to match aircraft performance. Thus, variations in the operating characteristics or inputs to the actual aircraft occur as parameter changes in the aircraft model, the objective being to alter or force the predictive loop to imitate the actual aircraft as closely as possible. The effects of environmental perturbations also appear in the auxiliary model via the transfer of information on the current state, $\overline{x}$ and $\overline{u}$, of the aircraft and its feedback control system. This information is used as initial conditions in the iterative prediction calculations. In summary, adjustment of $\overline{p}_M$ allows prediction and hence optimization of $\overline{\lambda}_M$ on the basis of the current estimate of the aircraft transfer function; and state variable transfer permits prediction from the correct current state.

It should be evident from the above that the determination of the optimum $\overline{\lambda}$ by calculation of $\delta E/\delta\overline{\lambda}_M$ presumes the constancy of parameters $\overline{p}_M$ and $\overline{q}$, pilot input i_p, and state variables $\overline{x}$ and $\overline{u}$ over the prediction interval τ. Since this condition does not necessarily prevail due to changing environmental conditions, unpredictable pilot behavior, or changing performance criteria, it is necessary to constantly re-evaluate $\delta E/\delta\overline{\lambda}_M$ and the optimum $\overline{\lambda}$. This is a reasonably simple task in the system at hand, since $\overline{p}_M$ (t), $\overline{q}$ (t), $\overline{i}_p$ (t), $\overline{x}$ (t), and $\overline{u}$ (t) will presumably be low-frequency functions and essentially constant during a few high-speed evaluations of the gradient. (Special consideration of i_p (t) is made later.) The need to re-evaluate the optimum emphasizes, however, this fundamental characteristic of predictive adaptation schemes: current optima are calculated on the basis of extrapolated present information and represent the best estimate that can be made; they are not necessarily the true optima which could have been implemented given a prior information on the nature of the system changes and perturbations. The adaptive scheme can thus be visualized as a means for tracking the true optima on the basis of current measurements of performance. It attains the true value only when environmental perturbations no longer appear, i.e., when predicted and actual performance coincide.

Other non-predictive adaptive schemes are, of course, faced with similar problems of current "optima" lagging the true optima. This is due to the time lag between the measurement of performance and implementation of correction[3]. Generally these schemes have much longer lag times than the predictive techniques. Note for comparison purposes that both adaptive processes of Figure 1 operate very rapidly so that corrective action can begin essentially as soon as environmental perturbations become evident in the aircraft outputs. This delay is determined by various aircraft system time constants and the nature of the perturbations.

Although corrective action is fast, the quality of adaptation may be suspect if predicted response differs significantly with actual future performance, or if repeated evaluations of the optima are ineffectual in producing necessary corrections. Such situations can conceivably exist when:

1. the aircraft model and actual aircraft are described by significantly different transfer functions, and the mismatch cannot be compensated by parameter tracking.
2. parameter tracking and state variable transfer is ineffectual in transferring to the model sufficient information on the nature of the actual disturbance.
3. $\overline{i}_p$ and $\overline{p}_M$ vary drastically over the prediction interval.

The consequence of 3. above is evident upon consideration of i_p (t). Values of this function are sampled at the start of each prediction calculation and assumed to prevail over the entire prediction interval. It is true, however, that the optimum $\bar{\lambda}$ depends not only on the present value of i_p (t) but on its functional nature as well. To assume a form for the function, namely i_p (t) = constant, even though the constant is updated from prediction to prediction, may lead to an inaccurate calculation of the optimum. This situation may be overcome by extrapolating the recent past history of i_p (t) over the prediction interval. Such a modification would allow for greater utilization of present information in predicting future performance.

The i_p sampler at the input to the auxiliary system now has a dual purpose. It prevents discontinuous pilot inputs from affecting the prediction during the course of a run (pilot inputs are regarded here as being piecewise continuous with low frequency content over continuous segments), and it becomes part of the extrapolation system.

Aircraft Control System and Models

A yaw control system for a supersonic transport (Figure 2) as presented by Whitaker[4], was chosen to demonstrate the adaptive technique.

The aircraft control system is expected to perform satisfactorily over extremely wide ranges of speed and altitude. The attendant changes in performance characteristics and a variety of environmental conditions make modification of the controller gains necessary. In the system the pilot's input yaw rate is monitored

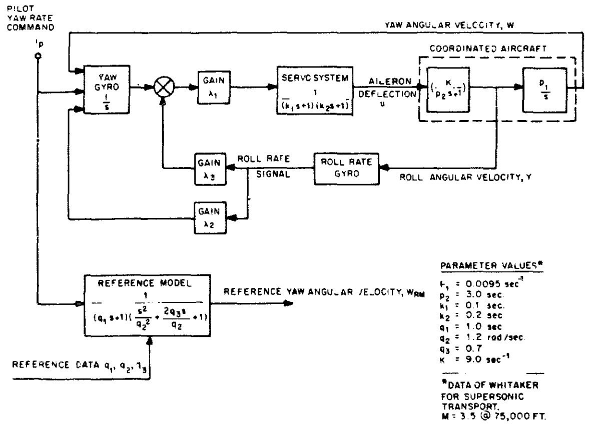


Figure 2. Aircraft Control System Block Diagram. (Form of System and Parameter Values According to Whitaker.[4])

by a yaw integrating gyro which produces a control action causing the vehicle to roll until a component of its angular velocity about its yaw axis is equal to the yaw rate command. (More detailed descriptions of this system are given in Reference 3.) The same basic fifth-order representation was used for both the actual aircraft control system and its model, although nonlinearities and additional inputs were frequently included in the actual system in order to test the adaptive scheme.

Three control loop gains are to be adjusted: λ_1, λ_2, λ_3, corresponding to gains in the forward loop, the roll angle stabilization loop, and the roll rate damping loop, respectively.

The reference model is seen to be a third-order system with three parameters: q_1, q_2, q_3, which are time constant, frequency, and damping factor, respectively. These parameters define the desired response and may be altered at will.

A single aircraft output, the yaw angular velocity W is taken as a measure of aircraft performance. It is assumed that delays in measuring all output variables are negligible.

Index of Performance

A single index of performance given by $E = \int_{t}^{t+\tau} \left| W_{RM} - W \right| dt$ was used to adjust the three parameters. This was later found not to be optimum due to the existence of local minima, insensitivity to certain parameter changes, and parameter cross-coupling, but is retained here for the sake of exposition of the technique. A better approach employs several E's defined by:

$$E_i = \int_{t+\tau_{1i}}^{t+\tau_{2i}} \left| W_{RM} - W \right| dt \qquad i = 1, 2, 3 \qquad (2)$$

where $\tau_{2i} - \tau_{1i}$ is a portion of the prediction interval. The τ values can be chosen to minimize the interactions of the three parameters. This form has been suggested by Whitaker.

Adaptive Process No. 1

The optimization technique used is a modification of the method of steepest descents according to Witsenhausen[5]. A search to find the minimum of E ($\lambda_{M1}, \lambda_{M2}, \lambda_{M3}$) from an arbitrary initial starting point $\lambda_{M1}, \lambda_{M2}, \lambda_{M3}$ and E° proceeds as follows:

1. By perturbing the λ_{Mi} values by a small amount h calculate approximate partial derivatives
$$\frac{\delta E}{\delta\lambda_{Mi}} = \frac{\delta E^{+} - \delta E^{-}}{2h} \qquad (3)$$
where
$$\delta E^{+} = E\,(\lambda^{\circ}_{M1}, \ldots, \lambda^{\circ}_{Mi}+h, \ldots, \lambda^{\circ}_{M3}) - E^{\circ}$$
$$\delta E^{-} = E\,(\lambda_{M1}, \ldots, \lambda_{Mi} - h, \ldots, \lambda_{M3}) - E^{\circ}$$
This requires six high-speed prediction runs.
2. Store the results of all partial derivative determinations. This is done in quantized form to place the storage load on the digital computer. The quantization is done by comparing the partial derivatives with a quantity L, arbitrarily but carefully specified, and determining a new quantity Z_i:
$$\begin{aligned} Z_i &= +1, (\delta E^{+} - \delta E^{-}) < -2hL \\ &= 0, -2hL < (\delta E^{+} - \delta E^{-}) < 2hL \\ &= -1, (\delta E^{+} - \delta E^{-}) > 2hL \qquad (4) \end{aligned}$$
The unit vector Z defines a direction in λ— space in which a decrease in E is probable.
3. Adjust the parameters to $\lambda^{\circ}_{Mi} + Z_i\triangle$, where $\triangle$ is some fixed quantity.
4. Test for an improvement in E. If E is lower, continue updating all parameters in the direction found, until no improvement occurs. When this happens, repeat 3 with a smaller $\triangle$. When this also fails, repeat the whole process starting from the new position in λ_M space.
5. Stop the optimization procedure when either all $Z_i = 0$ or when at least one improvement cannot be obtained with the smaller $\triangle$ proceed steps.

The information flow for this procedure is given in Figure 3. Note the automatic recycle feature provided to continuously re-evaluate the optimum following its initial determination. The recycle is started automatically following a stop condition, provided that a manual stop

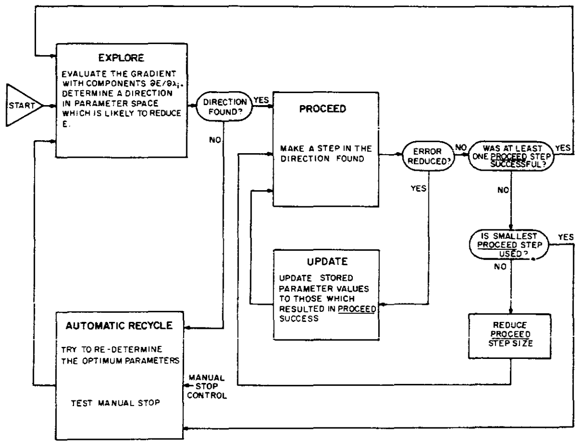

START
EXPLORE
EVALUATE THE GRADIENT WITH COMPONENTS ∂E/∂λi. DETERMINE A DIRECTION IN PARAMETER SPACE WHICH IS LIKELY TO REDUCE E.
DIRECTION FOUND?
YES
NO
PROCEED
MAKE A STEP IN THE DIRECTION FOUND
ERROR REDUCED?
NO
YES
WAS AT LEAST ONE PROCEED STEP SUCCESSFUL?
YES
NO
IS SMALLEST PROCEED STEP USED?
YES
NO
REDUCE PROCEED STEP SIZE
UPDATE
UPDATE STORED PARAMETER VALUES TO THOSE WHICH RESULTED IN PROCEED SUCCESS
AUTOMATIC RECYCLE
TRY TO RE-DETERMINE THE OPTIMUM PARAMETERS
TEST MANUAL STOP
MANUAL STOP CONTROL

Figure 3. Optimization Procedure (Simplified).

command has not been issued by the operator. It is very important upon recycling to re-evaluate the reference error E° by making a run with the λ_i currently in effect. Failure to do so may result in a "locked-in" condition, wherein a new optimum cannot be achieved because the new minimum E is greater than the E° just determined. Such a condition can occur because environmental upsets in effect "shift" the E hypersurface.

Refinements in the basic program, as suggested in Reference 5, were made in order to decrease convergence time.

Adaptive Process No. 2

The actual aircraft can be described by some vector function of the form
$$\dot{\bar{x}} = \bar{f}\,(\bar{x}, \bar{p}, \bar{u}, \bar{I}) \qquad (5)$$
where I refers to environmental inputs. The aircraft model is defined by
$$\dot{\bar{x}}_M = \bar{g}\,(\bar{x}_M, \bar{p}_M, \bar{u}_M) \qquad (6)$$
We seek the values for $\bar{p}_M$ for which the actual system outputs and model outputs coincide as well as possible. This optimum is obtained directly by substituting $\bar{x} = \bar{x}_M$, $\bar{u} = \bar{u}_M$, and $\dot{\bar{x}} = \dot{\bar{x}}_M$ into equation (6) and solving the resulting algebraic expression for $\bar{p}_M$. This technique is valid if the number of components of $\bar{p}_M$ are equal to or less than the number of components of $\dot{\bar{x}}_M$. It also is based on the assumption that the necessary state variables and control variables can be measured. It may be

necessary to differentiate (and filter) aircraft outputs to obtain certain components of $\bar{x}$.

It is of interest to note that differences between the actual aircraft and its model ($\bar{f} \neq \bar{g}$) or the existence of environmental inputs ($\bar{I} \neq$ zero) are reflected in the model via changes in $\bar{p}_M$.

For the system under consideration equation (6) is

$$\begin{aligned} \dot{W}_M &= p_{M1} Y_M \\ \dot{Y}_M &= \frac{1}{p_{M2}} (K_M u_M - Y_M) \end{aligned} \qquad (7)$$

where K_M is taken to be constant. The optimum p_{Mi} are given by

$$p_{M1} = \dot{W}/Y$$

$$p_{M2} = (K_M u - Y)/\dot{Y}$$

A difficulty in determining these quotients occurs under steady state conditions, when numerators and denominators become zero but p_{M1} and p_{M2} are determinate. A division circuit using the method of steepest descents[7] affords a solution to the problem and was used in the simulation. Initial values for p_{M1} and p_{M2} were obtained from a knowledge of p_1 and p_2 under assumptions that $\bar{f} = \bar{g}$ and $\bar{I} = 0$ initially.

The roll rate Y is measured by a roll rate gyro as part of the basic control system. $\dot{W}$ and $\dot{Y}$ would have to be determined by differentiation of aircraft outputs W and Y. The main servo output u could be measured easily.

Other, more sophisticated parameter tracking techniques could be used.[8] The possibility of time-sharing the optimization techniques of Adaptive Loop No. 1 was also considered. These alternatives are feasible, but require considerable additional computation. The technique used provided satisfactory results with a modest expenditure of computational equipment.

Hybrid Program

Allocation of Tasks

The hybrid system was an Electronic Associates HYDAC 2000 computer consisting of a PACE 231-R general purpose analog computer

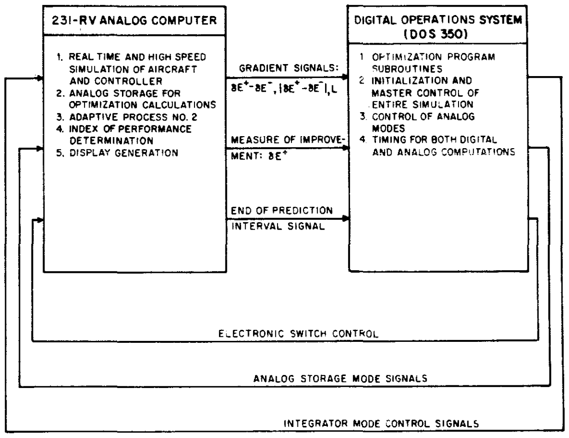

231-RV ANALOG COMPUTER
1. REAL TIME AND HIGH SPEED SIMULATION OF AIRCRAFT AND CONTROLLER
2. ANALOG STORAGE FOR OPTIMIZATION CALCULATIONS
3. ADAPTIVE PROCESS NO. 2
4. INDEX OF PERFORMANCE DETERMINATION
5. DISPLAY GENERATION
GRADIENT SIGNALS: δE⁺-δE⁻, |δE⁺-δE⁻|, L
MEASURE OF IMPROVEMENT: δE⁺
END OF PREDICTION INTERVAL SIGNAL
DIGITAL OPERATIONS SYSTEM (DOS 350)
1 OPTIMIZATION PROGRAM SUBROUTINES
2 INITIALIZATION AND MASTER CONTROL OF ENTIRE SIMULATION
3 CONTROL OF ANALOG MODES
4 TIMING FOR BOTH DIGITAL AND ANALOG COMPUTATIONS
ELECTRONIC SWITCH CONTROL
ANALOG STORAGE MODE SIGNALS
INTEGRATOR MODE CONTROL SIGNALS

Figure 4. Allocation of Computer Tasks.

and a digital operations system (DOS 350). The 231-R was equipped with a high-speed electronic mode control unit (MLG). The allocation of computational tasks are shown in Figure 4.

The Analog Program

An unscaled schematic for the basic aircraft system is given in Figure 5. Two of these circuits are required, one operating in real time to represent the actual system, the other at high speed (1000 times faster) for prediction. Noise inputs, failure conditions, and changes in the structure of the circuit were applied only to the real time circuit. The parameters p_1 and p_2 were fixed in the actual system circuit, but variable (as determined by the tracking process) in the predictive circuit. Initial conditions for the actual system were taken to be zero; the

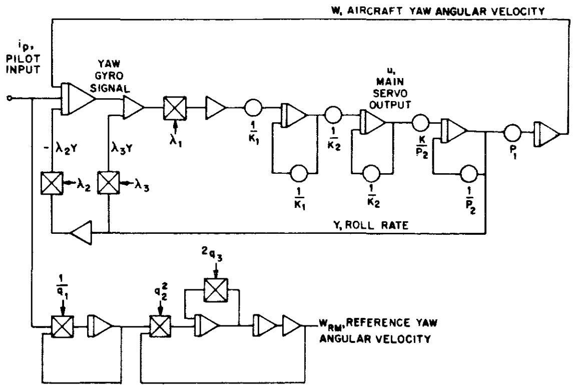


Figure 5. Analog Circuit for Aircraft, Controller, and Reference Model (Unscaled).

initial conditions for integrators in the predictive circuit were taken from outputs of their corresponding real time integrators. The parameters λ_M and λ were determined for the two circuits by different logic programs: the λ_M were modified to evaluate $\delta E/\delta\lambda M$ but λ°_i were changed only if prediction with $\lambda^\circ_{Mi} + Z_i\Delta$ yielded a smaller E. (Note that λ°_{Mi} is equivalent to λ_i.)

Proper scaling of the gains λ_{Mi} and λ_i was essential for the optimization process. Use of a single, fixed exploration step size, h, depends on this. Improper scaling results in a gain between $\delta E/\delta\lambda_{Mi}$ and λ_{Mi} that is either too high or too low, causing the optimization to stop at a non-optimum point or produce unstable solutions. A few simple trial-and-error experiments on the computer permitted determination of appropriate scale factors.

Two steepest-descent division circuits, of the type shown in Reference 7, were implemented to calculate p_{M1}, and p_{M2} according to equation (8). In the simulation the derivatives $\dot{W}$ and $\dot{Y}$ appear as explicit voltages and need not be determined by differentiation.

The DOS Program

The DOS consists of a collection of programmable, parallel digital logic, memory, and conversion components. Only the logic and some conversion components were used in this problem. These include logic gates, clocked flip-flops, shift registers, monostables, counters, comparators, and electronic switches, which can be interconnected on a patch panel in a manner similar to analog computer programming. Propositional and sequential logic calculations are handled by the gates and flip-flops, respectively, or by packaged combinations of these (shift registers, counters, etc.). The comparators and electronic switches provide the analog-to-logic and logic-to-analog conversions. In addition, logic levels on the DOS can control

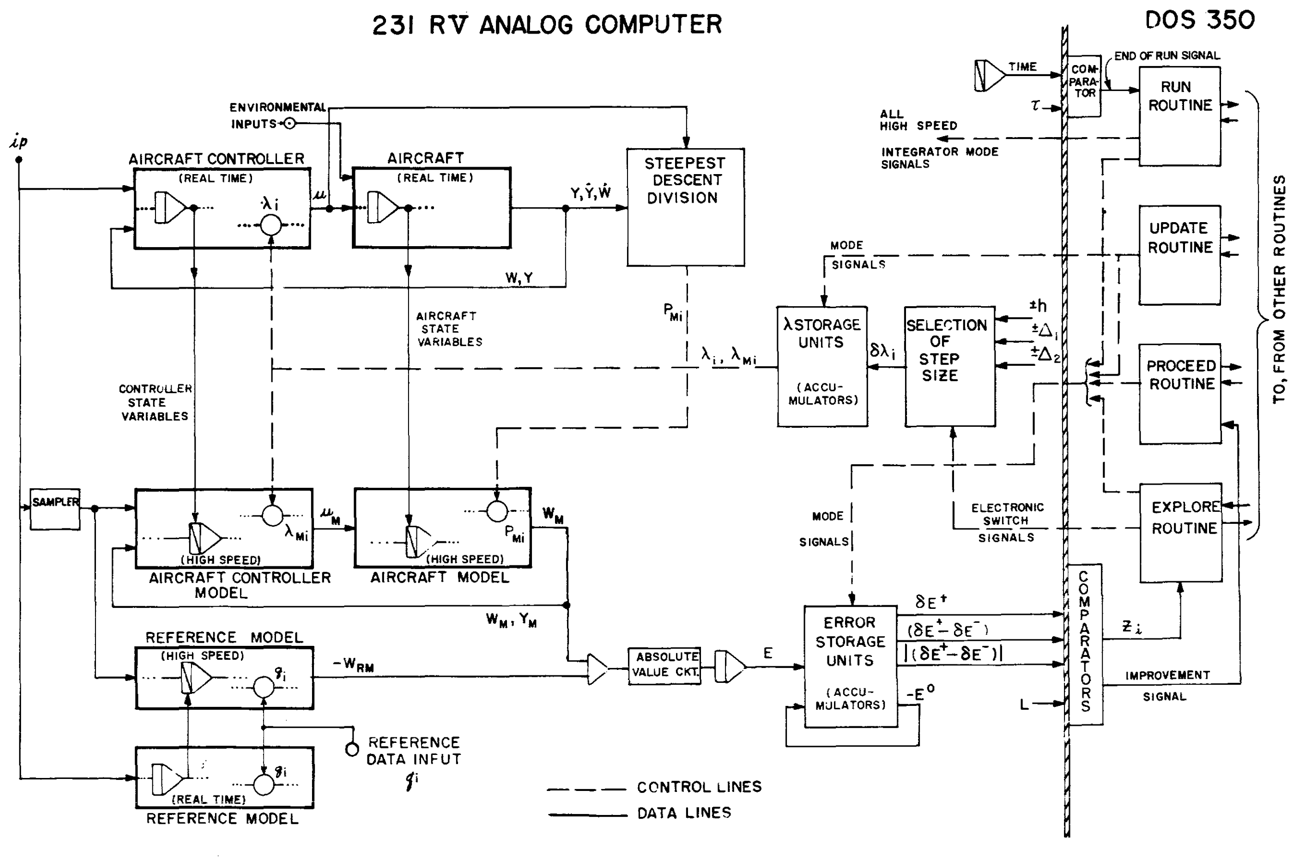


Figure 6. Simplified Schematic of the Hybrid Program.

electronic switches in the analog circuits, the modes of analog integrators, and the modes of track-hold analog storage units. The operating details of these devices as well as their program symbols are considered elsewhere[5, 6, 9].

In this simulation the DOS directs the entire optimization program. The mechanization of the program, including circuit diagrams, has been given by Witsenhausen[5]. It suffices here to note the program structure—a set of subroutines, appropriately interlocked, whose functions are to:

1. select the step sizes h, $\triangle_1$, and $\triangle_2$ and set the appropriate switches accordingly. (See Figure 6.)
2. test for stop—recycle conditions.
3. initiate a high-speed analog run to determine E, with a single parameter augmented by +h, then —h, repeating this for all three parameters.
4. evaluate Z_i according equation (5) and store the results in digital form during the determination in step 3.
5. initiate a high-speed analog run with the appropriate $Z_i\triangle$ modifications in all three parameters and test for improvement.
6. update the λ_i and E° analog storage units to the values of λ_{Mi} and E that resulted in improvement.
7. repeat steps 5 and 6 until no improvement is obtained or until a preset number of "proceed" steps have occurred.
8. repeat steps 3-7 until a stop—recycle condition occurs.
9. clear the results of previous logic calculations when a stop—recycle condition exists, make another analog run with the present λ_{Mi}, and store the resulting value of E as the new E°; start with step 1 again.

The hybrid program is shown schematically in Figure 6 with emphasis placed on the interface. Various portions of the analog program are represented symbolically by a single integrator; parameters are represented by a potentiometer symbol. High-speed portions of the program are indicated by a slash line drawn through the integrator symbol. The storage units employed analog integrators, programmed as accumulators. Selection of the magnitude and sign of the λ perturbations ($\delta\lambda_i$) was done by electronic switches under control of the DOS logic elements. Control outputs of the DOS subroutines are indicated; their interconnections are given in Reference 5.

RESULTS

Responses of the adaptive control system to a variety of test conditions are presented in Figures 7-11. The standard aircraft and reference model parameters given in Figure 2 were used unless otherwise specified.

Convergence: Figures 7a and 7b illustrate the speed of convergence to an optimum state from

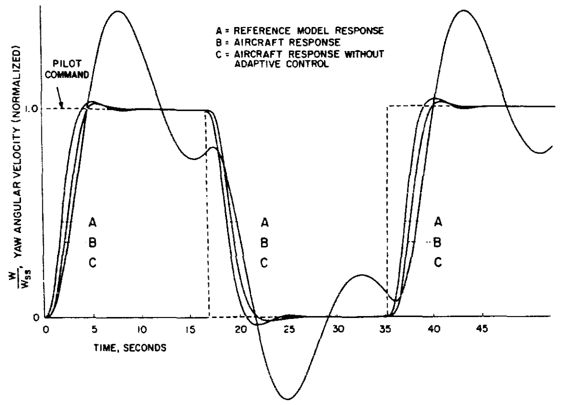


Figure 7a. Aircraft Response to Pilot Inputs. Controller Gains Initially Non-Optimum.

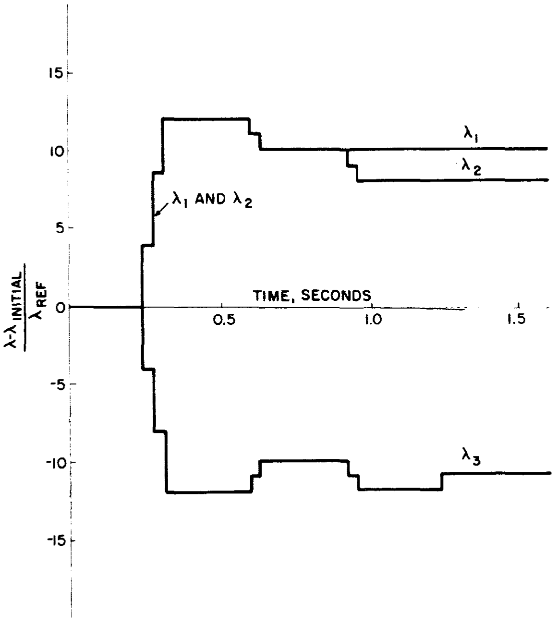


Figure 7b. Adaptation of Gains During Response of Figure 7a.

an arbitrary initial set of parameter values. The optimization procedure brought the parameter values for λ_1, λ_2, and λ_3 from 60, 10 and 20 (dimensionless as shown in Figure 7b), respectively, to 70, 18, and 11 in approximately 1.2 seconds. The index of performance E was reduced to 5.0% of its original value. The input function was a series of steps applied by the pilot in an attempt to perform roll-in and roll-out maneuvers. Although a perfect correspondence between actual and reference model responses was not obtained, the actual response was a drastic improvement over the marginally stable response that would have obtained in the absence of adaptation. Note also that the marginally stable solution is the first response obtained in the iterative optimization program, and the final response is the one shown for the actual aircraft. Approximately 40 high speed prediction runs were made in the intervening 1.2 second period. The prediction was made with $\tau = 20$ seconds (real time), corresponding to 20 ms of computational time in the high speed loop. The speed of adaptation is seen to be so rapid that actual performance is scarcely affected by the poor initial values of λ. It was assumed in this experiment that $p_i = p_{Mi}$ and that no environmental effects were operative. The speed of adaptation found here is considerably faster than that reported by Whitaker for essentially the same aircraft control system but different adaptive technique.

Environmental Effects: Using a slightly altered reference model ($q_1 = 1.25$), the response to a change in the aircraft's operating characteristics was obtained. At time $t = 0$, events occurred within the aircraft leading to a step change in velocity (increase). This change is sensed by Adaptive Process No. 2 in the manner shown in Figure 8a, i.e., p_{M1} changes. Without this parameter adjustment a poor response would result (curve C). With no adaptive control operating at all a poorer response obtains (not shown).

The response to an unanticipated input to the aircraft, in the form of a large, constantly applied deflection of the control surface, is shown in Figure 9. This was a most severe test of the adaptive system and demonstrated the need for improvement in defining an index of performance. Parameter cross-coupling with the present E was significant in this experiment. It is felt that the modifications suggested above would constitute a significant improvement. In view of the magnitude of the disturbance and the fact that exact compensation for an undesirable

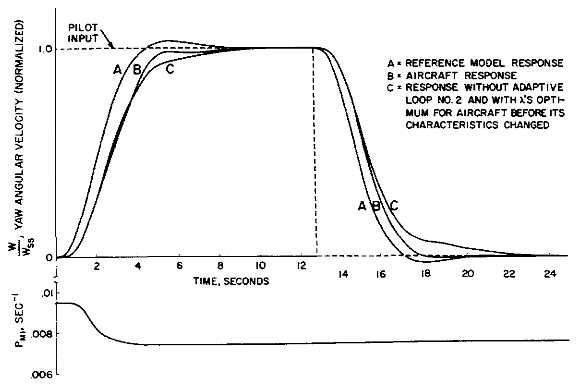


Figure 8a. Aircraft Response to Pilot Inputs. A Change in Aircraft Speed @ t=0 Initiates Adaptation.

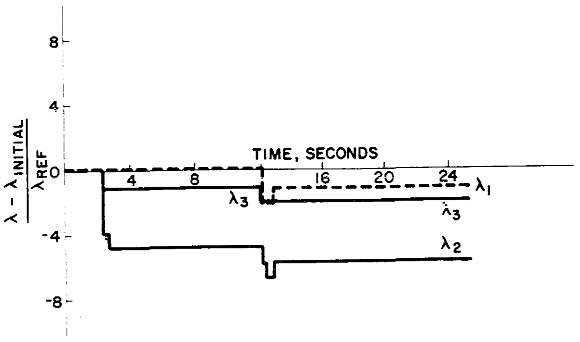


Figure 8b. Gain Adjustments During Response of Figure 8a.

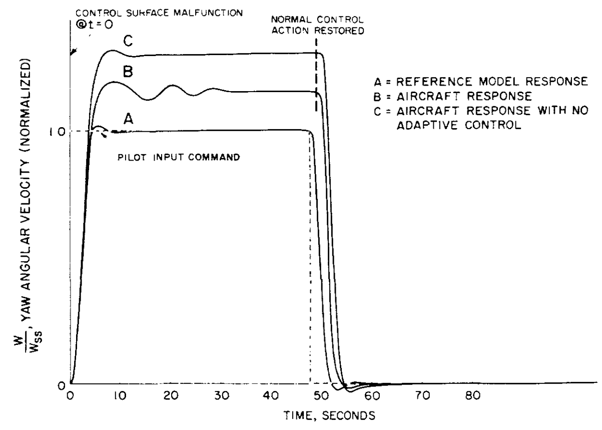


Figure 9. Aircraft Response to Pilot Inputs. Adaptation Initiated by Control Surface Malfunction.

input may be impossible by adjusting controller gains, the response obtained is considered adequate and certainly better than that obatined without adaptive control.

The response to a failure condition is shown in Figure 10. At time t = 0, the main servo becomes "locked" in an inactive state and remains in this condition for three seconds. This is an "open-circuit" type of failure. During the three second period the remainder of the control system, which operates correctly, reaches some limit condition, because no feedback signals from the aircraft outputs are forthcoming. At

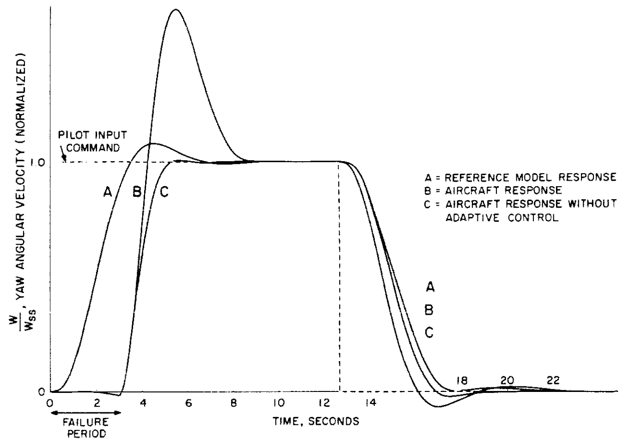


Figure 10a. Aircraft Response to Pilot Inputs When Main Servo Experiences Temporary Failure.

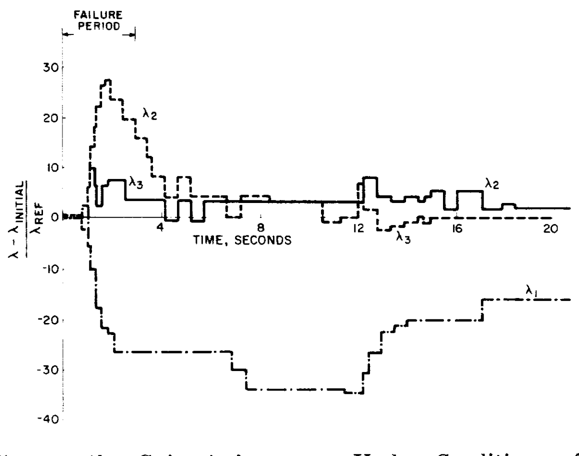


Figure 10b. Gain Adjustments Under Conditions of Temporary Servo Failure.

t = 3, the servo operation becomes normal. The response of the system from the limit condition then reached is of interest. Adaptive control is seen to institute corrective action very rapidly and avoids the large overshoot associated with no adaptive action.

Changing Goals: Intentional changes in the parameters q_i, should lead to new aircraft performance, since the reference model is now changed. The successful adaptation to these changing specifications is shown in Figure 11.

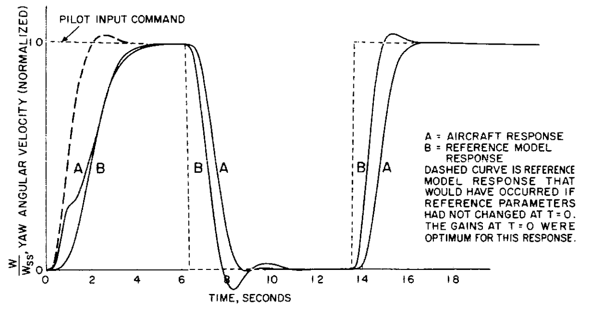


Figure 11. Aircraft Response to Pilot Inputs. Reference Model Changes During Maneuvers.

CONCLUSIONS

Hybrid computation had advanced to the point where the feasibility of using fairly sophisticated adaptive control techniques can be easily and economically investigated. Extensions of this illustrative problem to others requiring multispeed operations with iterations of differential equations can easily be made. The effectiveness of the analog computer in the solution of such problems is greatly enhanced by a flexible logical computer, which permits greater utilization of analog time and equipment and which can be easily tailored to fit each application.

REFERENCES

1. EVELEIGH, V. W., "Adaptive Control Systems", Electro-Technology, April, 1963.
2. WIDROW, B., "The Speed of Adaptation in Adaptive Control Systems", ARS Guidance, Control, and Navigation Conference, Stanford, California, August, 1961.
3. WHITAKER, H. P., YAMRON, J., and KEZER, A., "Design of Model-Reference-Adaptive Control Systems for Aircraft", Report R-164, Instrumentation Laboratory, Massachusetts Institute of Technology, September, 1958.

4. WHITAKER, H. P., and KEZER, A., "Use of Model-Reference-Adaptive Systems to Improve Reliability", ARS Guidance, Control, and Navigation Conference, Stanford, California, August, 1961.
5. WITSENHAUSEN, H. S., "Hybrid Techniques Applied to Optimization Problems", Spring Joint Computer Conference, San Francisco, California, May, 1962.
6. WITSENHAUSEN, H. S., LANDAUER, J. P., et al., "Applications Reference Library on Hybrid Computation", Electronic Associates Inc., Princeton, New Jersey.
7. FAVREAU, R. R., "Dividing Circuit Obtained by Applying Method of Steepest Ascents", PCC Report No. 132, Electronic Associates, Inc., Princeton, New Jersey.
8. PURI, N. N., and WEYGANDT, C. N., "Multivariable Adaptive Control System", IEEE Transactions, May, 1963.
9. HALBERT, P. W., LANDAUER, J. P., and WITSENHAUSEN, H. S., "Hybrid Simulation of Adaptive Path Control", AIAA Proceedings of the Simulation for the Aerospace Conference, Columbus, Ohio, August, 1963.

ACKNOWLEDGMENT

The author gratefully acknowledges the technical assistance and suggestions afforded by Mr. Omri Serlin and Mr. Hans Witsenhausen. In addition, Mr. A. I. Katz, Mr. J. Kennedy, and Mr. D. F. Welsher offered helpful comments on this paper.

A COMPUTER DRIVEN SIMULATION ENVIRONMENT FOR AIR TRAFFIC CONTROL STUDIES

E. A. Robin, R. S. Pardee, D. L. Scheffler, and F. C. Holland
TRW Computer Division
A. G. Halverson—Federal Aviation Agency

The technique of real-time simulation for studying air traffic control problems has been employed for many years. This technique has enabled investigators to simulate live air traffic situations for the purpose of evaluating new concepts and their relationship to the air traffic controller. The modern high-speed digital computer has increased the scope of these real-time simulation studies to include large scale air traffic control systems employing such computers in the control process.

Up to the present time each aircraft used in real-time simulation studies (conducted by the F.A.A. at the National Aviation Facilities Experimental Center) has been generated by a special purpose analog simulator. An individual "pilot" would be assigned to "fly" each simulated aircraft in a study. A radar simulator would transform these X, Y aircraft coordinates into proper signals to display this target on a standard radar display. (See Fig. 1.) The air traffic controller uses voice communication links to communicate with the simulator pilots.

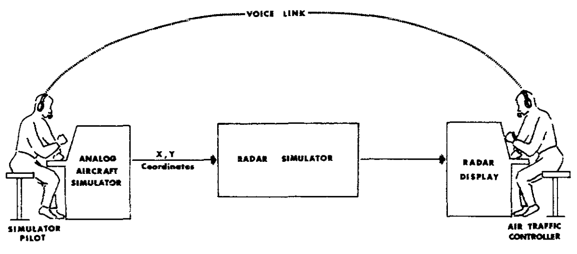


Figure 1. Analog Type Air Traffic Simulation.

Two of the major functions of a computer aided air traffic control system are scheduling and control. A sequence of aircraft is derived to deliver aircraft to a destination based on some criterion such as to safely maximize the aircraft acceptance rate at an airport, or reduce the average delay, controller workload, etc. The control function is then exercised to minimize the difference between the actual time the aircraft arrives at the destination and the schedule time. In the systems to be simulated this control is effected by the air traffic controller whose control decisions are based on computer-derived information displayed to him on some display media.

The increased controller workload caused by growth of air traffic and the influx of large high-speed jet aircraft in today's system necessitates quicker and more accurate decisions. Thus the computer is being proposed as an integral part of future air traffic control systems. Control suggestions may be derived by the computer and displayed to the controller for action. Many data processing functions, based on complex logic, must be performed in the process of scheduling and controlling aircraft. A general purpose digital computer with its flexibility can be easily used to implement this logic.

The question of "Why use a digital computer in real-time simulation?" may be answered in terms of its many advantages, e.g.:

Study of Computer Aided Systems. If the applicability of a digital computer in an air traffic control system is under investigation, then it follows that the simulation environment required to study solutions to this problem must also employ a digital computer. If more than one application is to be studied, then it is important that a general purpose simulation environment incorporating a general purpose digital computer should be available.

Generation of Aircraft Targets. Recent interest in introducing automation to terminal area near the airport air traffic control concepts has created stringent requirements on the realism of the analog aircraft simulators in terms of aircraft heading, position, and velocity. A digital computer can be used to generate these targets within this framework of requirements. It permits the inclusion of controlled navigation and speed error distributions so the effect of errors on the simulated system can be studied. The versatility of digital computer generated targets also allows additional functions to be programmed in the aircraft simulation that were unavailable in previous analog simulation equipment, e.g., the ability to realistically navigate the aircraft using the instrument landing system (ILS), or realistic simulation of take-off acceleration, etc.

Simulation of Sub-Systems. A digital computer can be used to simulate a large variety of complex sub-systems that might occur in present or future air traffic control systems. Simulation of these sub-systems avoids the necessity of developing these often complex equipment until their value has been reasonable well determined. Some examples of these sub-systems might be the radar or other form of position acquisition system, the radar or beacon tracking system, or a beacon attitude receiving and display system.

Flexible Programmable Situation Displays. Previous experience has shown that buffered general purpose displays are needed for controller and pilot situation displays. These units should also be capable of displaying tables and other alphanumeric information to the controller. By building the simulation around a general purpose digital computer the flexibility of a programmed display system with no "built in" format restriction can be readily obtained.

Data Collection and Analysis. The computer can be used to collect and analyze data while performing the other functions previously described. Thus the data collection can be integrated into any simulation.

Rapid Changes from Problem to Problem. A stored program enables rapid changes to be made from run to run. By changing the program an entire new simulation can be started with minimum effort in a few minutes. Thus one facility can be utilized on a time-shared basis to study many different air traffic control problems.

EQUIPMENT CONFIGURATION

Based on early experience it was decided that a large high-speed digital computer would be required to properly conduct these simulation experiments.[1] At that time the IBM 709 was installed at the National Aviation Facilities Experimental Center and was therefore recommended for use with this Computer Driven Simulation Environment (CDSE). Subsequently, the IBM 709 was replaced by an IBM 7090. The 7090 computer was equipped with two data channels,[2] a Direct Data Connection,[3] and a real-time digital clock.[4] (See Fig. 2.) The simultaneous read-write-compute feature of this computer made possible the transfer of large

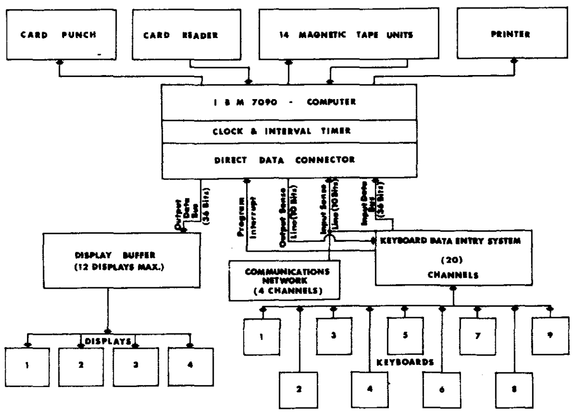

CARD PUNCH
CARD READER
14 MAGNETIC TAPE UNITS
PRINTER
I B M 7090 - COMPUTER
CLOCK & INTERVAL TIMER
DIRECT DATA CONNECTOR
Output Data Bus (36 Bits)
Program Interrupt
Output Sense Line (10 Bits)
Input Sense Line (10 Bits)
Input Data Bus (36 Bits)
DISPLAY BUFFER (12 DISPLAYS MAX.)
COMMUNICATIONS NETWORK (4 CHANNELS)
KEYBOARD DATA ENTRY SYSTEM (20) CHANNELS
DISPLAYS
1
2
3
4
KEYBOARDS
1
2
3
4
5
6
7
8
9

Figure 2. IBM 7090 Computer Equipment Configuration.

Figure 3. Charactron Display—Air Traffic Controller Position.

Figure 4. Simulator-Pilot T.V. Monitor Display.

blocks of data without excessively increasing program execution times. The Direct Data Connection provided high-speed data transfer to the Display Buffer system and from the Keyboard Data Entry system.

The displays chosen were of the Charactron type (19″ diameter) and had ability to display alphabetic and numeric information (fixed character size), special symbols (for aircrafts, fixes, etc.) as well as lines. Since the display format was under program control, flexibility existed for simulation on these displays for any desired situation (Fig. 3). Initially, there were four displays connected to the computer through the Display Buffer system. Three of these are presently used for air traffic controller consoles and one provides data for the simulator pilots by means of a closed circuit TV system. Four high resolution cameras transmit the data from the Charactron to six simulator pilots' TV monitor displays (Fig. 4).

A set of input keyboards are required to allow controllers and pilots to communicate with the computer. Six keyboards are used for simulator-pilot input functions (Fig. 5) and three for the air traffic controller positions (Fig. 6).

Four voice communication channels are provided between the controllers and the pilots. Each channel is monitored via the 7090 sense lines in order to derive the distribution of

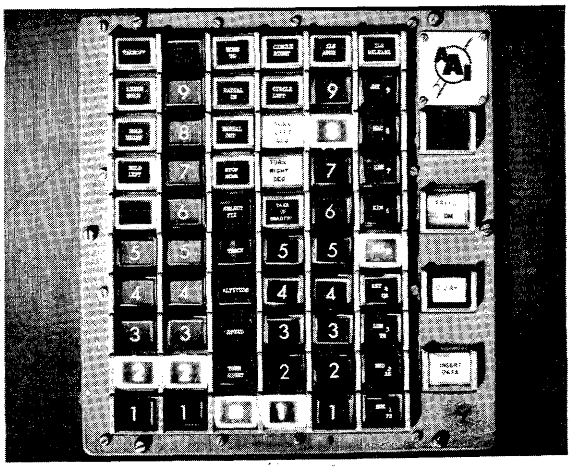


Figure 5. Simulator-Pilot Input Keyboard.

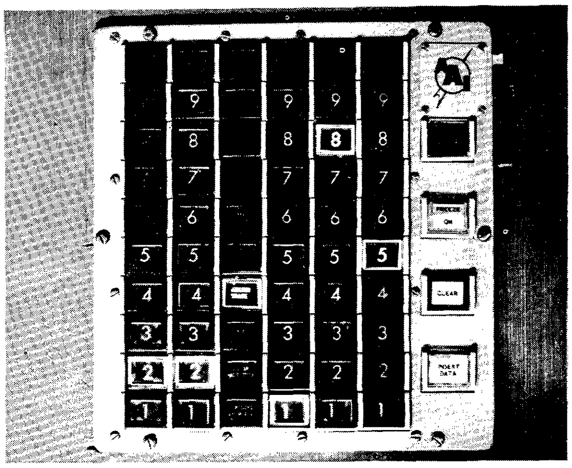


Figure 6. Air Traffic Controller Input Keyboard.

message lengths thereby enhancing the communication workload and delay studies.

PROGRAM ORGANIZATION AND TECHNIQUE

A major requirement in the selection of a programming language used was that it allow the programmer to utilize to the fullest extent the total capabilities of the IBM 7090 computer, including simultaneous input-output-compute, program interrupts, and all machine registers and instructions. At the same time it was desirable to have available a compiler which would facilitate the translation of mathematical statements into machine language since significant portions of simulation programs were to consist of mathematical formulae. Also, the use of a compiler would allow immediate programming contributions from highly technical personnel with limited programming experience.

The programming languages selected were the FAP Assembly Program and the FORTRAN Compiler. The main reason for the selection of these languages was that they were available and proven in daily operation. Programs written in either language could be used with those written in the other. Thus, persons of varying backgrounds and programming experience were able to work together.

An effort was made to write all real-time programs in the FAP assembly language and the off-line analysis programs in the FORTRAN language. The reason for this was that it was possible to write more efficient operational programs in machine language; and it was important to have real-time programs consume as little running time as possible, while this requirement was not necessary in the off-line programs.

For the purposes of these simulations the FORTRAN/FAP system proved to be adequate and imposed no limitations on the programmer's ability to exploit fully the capabilities of the computer. The problem of lengthy subroutine linkages which are inefficient in respect to both space in the computer memory and in running time during the operation of the problem was solved simply by using the COMMON pseudo-operation of FAP and FORTRAN. By the time a simulation program became operational, this list had grown to a size of over 10,000 words of memory, or about one-third of the total core memory. Since most subroutines made many references to data in the COMMON list, the use of this device saved thousands of memory locations. The maintenance of the COMMON list was at worst an inconvenience but certainly was not a hardship.

Three levels of interrupts were used in the simulation program. In order of decreasing priority they were the real-time clock interrupt, the external keyboard entry interrupt and the display transmission interrupt which was the 7090 data channel trap. The controllers or pilots entered information into the computer via the keyboards which triggered the keyboard entry interrupt. In general, no matter what function the computer was performing at the time, this function was interrupted for the time it took the keyboard interrupt program to read the keyboard message, check it for validity and store it in the message input buffer area of core; after which control was returned to the program in progress at the time of the interrupt. This whole process was completed in less than one millisecond. The clock interrupt operated in a similar manner and was set to interrupt the main program every ½ second. The third type of interrupt was used to transmit data to the displays without tying up the computer while sending this information. Had it been necessary to suspend other computer functions while updating displays, 20% of the available computing time would have been required.

One of the most significant subroutines in the Computer Driven Simulation Environment was the program to generate the simulated targets under the control of the pilot keyboards.[5] This technique resulted in a high degree of precision but more importantly, it permitted the inclusion of controlled navigation and speed error distributions. A number of additional functions were programmed in the digital aircraft simulation that were unavailable in previous simulation equipment.

Each aircraft flight plan was assigned a simulator number when it entered the problem.

A simulator assignment program maintained a dynamic list of available simulator numbers.

The simulator number was the key to all information concerning a given flight. Every flight had about fifty descriptive parameters associated with it, such basic things as X, Y, and Z coordinates, indicated air speed, performance characteristics, and many other parameters associated with the computer program. Each distinct parameter was stored in blocks of words equal to the number of simulators so that once the simulator was known, any piece of information concerning the flight could be obtained by indexing the initial location of the block by the simulator number.

For example, suppose a program required the altitude of the flight with the simulator number of 13. The desired altitude would be found in the 13th word of the block of altitudes associated with the simulators. This method of data storage proved to be very efficient in both target generation and display programs and provided a convenient method of reference between programs.

The collection of data on the movement of the simulated aircraft, performance measures and operational data was greatly simplified by using the same computer to generate targets and collect data. Data was written on magnetic tape during the simulation run and saved on tape for analysis after completion of the simulation. Tape buffering using the simultaneous write-compute of the 7090 Data Channel was used. This method made the control processing unit of the computer available at all times for the simulation program.

A TYPICAL SIMULATION PROBLEM

A typical simulation problem consisted of three main groups of programs arranged in a chain of three links. The purpose of Link 1 was to read into the computer some 94 parameters which specified the conditions of the run. On the basis of the parameters, input traffic flight plans were selected by the program from tape and stored away in the core. Also, data and tables needed by the operational control program were computed.

After completion of Link 1, the real-time simulation program, Link 2 was called. Link 2 consisted of 18,000 instructions and made use of a total of 28,000 memory locations. The initialization program (Link 1) ran for less than one minute compared to two hours or more for a simulation run (Link 2).

Link 2 consisted of the following main subroutines.

(1) Generation of Simulated Aircraft	1900 instructions
(2) Data Acquisition	160 instructions
(3) Display	2000 instructions
(4) Data Collection	1300 instructions
(5) Keyboard Entry	1200 instructions
(6) Operational Control Logic	3500 instructions

Upon completion of the simulation run the final data was recorded and Link 3 was called. Link 3 was read into the memory over Link 2. It consisted of some 52 different analysis programs to process the data collected during Link 2 and stored on magnetic tape.

Fig. 7 shows a functional diagram of the CDSE. The 7090 program consisted of four functional areas, simulated aircraft targets, data acquisition, operational control systems, and data collection. Some of the parameters read into the computer in Link 1 are shown in the figure. For example, the estimated aircraft characteristics, geometry of the area simulated, and the forecast wind are used in the operational control system logic. The characteristics of the data acquisition system, e.g., sampling

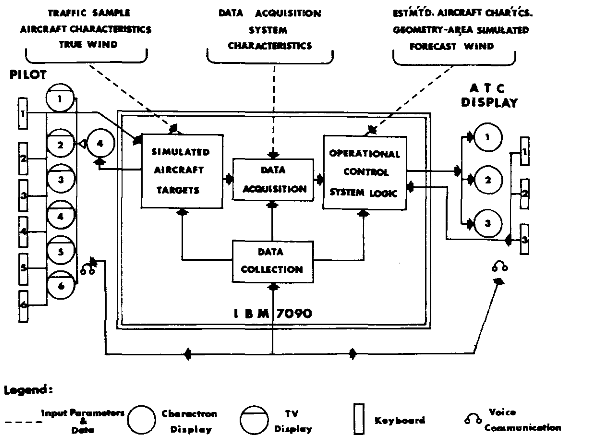


Figure 7. Functional Diagram of the Computer Driven Simulation Enviroment (CDSE).

rate of the radar or standard deviation of radar error, are used in the data acquisition subroutine. The external displays and keyboards provided the command and control hardware to integrate the air traffic controllers into the operational control system. They also allow the pilots to "fly" the simulated targets. Voice communication between the controllers and the pilots closes the control loop. The data collection program is able to monitor the communication lines, the simulated aircraft keyboard workload, and control system performance.

CONCLUSIONS

An extensive real-time simulation program can be handled most efficiently and expeditiously by maintaining a small group of specialists working closely together. Overall efficiency is increased if each individual in the group is given responsibility for a portion of the problem. This responsibility should begin with systems analysis and synthesis, and carry through to the flow charting, programming and debugging. The small group size and the full responsibility for major portions of the problem result in greatly reduced communication and coordinate requirements among members of the group. In addition, the knowledge and use of the computer machine language (FAP) or equivalent by all members of the group will make possible great savings in both program size and execution time.

The CDSE proved to be a versatile and reliable vehicle for carrying our real-time simulations of air traffic control systems. The keyboard and displays were both completely under program control, thus making it possible to change or improve formats in a matter of hours (or minutes) without any hardware modifications. Currently, two operational control systems are being evaluated. The ability to control displays and keyboards from the computer makes this use of the equipment possible

The use of programmed digital aircraft simulators proved to be completely successful. With the digital simulators there was never any problems of drift or alignment that were present in the analog simulators.

The use of the computer for the set-up and operational control of the real-time simulation proved to be a reliable and efficient operation. Input traffic was generated by the computer and stored on magnetic tape. At the beginning of a run, the traffic sample number was entered in the computer console keys, and pre-punched cards containing the simulation parameters read into the card reader. These were the only human operations required since from this point on, the computer controlled the entire simulation. This method resulted in a smooth and efficient operation plus a considerable saving in set-up and analysis time.

BIBLIOGRAPHY

1. R. S. PARDEE, F. C. HOLLAND, E. A. ROBIN, et al., "Air Traffic Control Studies Report No. 10, Project TASC (Terminal Area Sequencing and Control), Final Report," TRW Computers Company, February 1961. Contract FAA/BRD-112) available from Office of Technical Services, U. S. Department of Commerce, Washington, 25, D. C.
2. IBM Reference Manual, "709–7090 Data Processing System".
3. IBM Special Systems Feature Bulletin, "Direct Data Connection for IBM 7090", RPQ M90976.
4. IBM Special Systems Feature Bulletin, "Core Storage Clock and Interval Timer", RPQ F89349.
5. F. C. HOLLAND, "A Digital Aircraft Simulation Program for the IBM 709–7090", TRW Computers Company ATC Technical Memorandum 62–10, September 1962.

HYBRID TECHNIQUES FOR REAL-TIME RADAR SIMULATION*

Roger L. Boyell
Pennsylvania Research Associates, Inc.
133 South 36th Street
Philadelphia, Pennsylvania 19104
and
Henry Ruston
Moore School of Electrical Engineering
University of Pennsylvania
Philadelphia, Pennsylvania 19104

This paper summarizes results of a study of the design and application of special-purpose computers for the simulation of air-to-ground radars. Specifically, the computers must generate information in real-time to simulate the display of a scanning radar in a vehicle flying at Mach 3 and arbitrary altitude. The radar can view 10^5 square miles of terrain per second with a resolution of 60 points per mile in range (five per microsecond). This paper describes a hybrid simulator that exploits both the redundancy in the terrain and the repetitiveness caused by the radar scan pattern. An overall design is presented that affords reductions of 10^3 in storage capacity and computation speed, compared with a straightforward digital approach for generation of topographic profiles from which the display is prepared. Details are presented of data formats and transfer schemes through the memory hierarchy, and of algorithms for the reconstruction of analog profiles from a digitally stored contour map.

INTRODUCTION AND SUMMARY

The training of personnel in the operation of air-to-ground radars and in the interpretation of radar displays has always been a difficult problem. It is extremely expensive to fly a modern aircraft for the primary purpose of training only one or two personnel at a time in the use of a radar system. The art of radar interpretation can be learned only gradually, through continuing experience in the physical manipulation of controls that affect the operation of the radar and the appearance of its display. For the newer low-level, high-speed missions in particular, where a radar and its operator are an integral part of the vehicle's control loop, the real-time nature of the display is an important factor in the effectiveness of the training. Motion pictures and other conventional training aids are of limited utility, because of the highly preprogrammed nature of the training situation that would result. Furthermore, modern radars and modern aircraft have an overwhelming variety of modes of operation and extremely diverse sets of performance capabilities.

It is obvious that a radar simulator, operating in real time, is required. The simulation of

*This work is supported by Contract N61339–1025 between the U.S. Naval Training Device Center and Pennsylvania Research Associates, Inc. "The information contained herein is being disclosed in a confidential relationship."

air-to-ground radars historically has been performed by special-purpose, fully analog equipment. The current generations of simulator systems compress the important factors (height and reflectivity) of the terrain over which a pilot flies his aircraft, by a factor of several million, onto relief maps or density-modulated photographic plates. The plate or map is scanned by a point of light, in direct analogy with the scan pattern of the radar being simulated.

However, the Navy has been interested for some time in the applicability of digital computing techniques to this problem. Just as digital flight simulators were shown to have several important advantages over the older trainers, so it would be expected that digital radar simulators would have a greater degree of flexibility of changing the characteristics of the vehicle, the radar, and the terrain being simulated. This flexibility is important to avoid obsolescence of the simulator under continuing development of radars and aircraft.

For example, if the map of a target area were stored in numeric form on magnetic tape, a change dictated by new intelligence data would allow a revision of the map at the speed with which the tape could be updated by a computer. This change could be made orders of magnitude faster and cheaper than the revision of a relief map or photographic plate, yet retaining the accuracy and resolution usually required. However, even cursory investigation leads to the conclusion that straightforward application of conventional general-purpose computers is out of the question, because of the tremendous amount of data collected by an air-to-ground radar per unit time and the rapidity with which this data changes by virtue of the speed of the vehicle in which the radar is mounted.

Study of the form and amount of data collected and displayed by a radar has, on the other hand, led to the formulation of a model or overall logic design of a special-purpose analog-digital computer system for generating a realistic simulation. This paper describes one such hybrid computer, showing how the capabilities of both analog and digital equipment can be exploited to obtain a feasible model of a digitally oriented simulator, whose production of data is at a rate comparable to the megacycle bandwidths of modern airborne radars, but not restricted to the simulation of any particular radar system or set of performance specifications.

The map of the terrain and cultural features is stored in digital form, with numbers representing location, height, and type of terrain and structures. Redundancy in the data and repetitiveness in the radar scan pattern are taken advantage of by passing the data through several memories in succession, each memory acting as a transformation, or digital filtering, from greater to lesser data but lesser to greater speed of access. Ultimately, the digital data is used to set analog function generators, which reconstruct the terrain height and reflectivity profiles in synchronism with the radar display at speeds comparable to radar video bandwidth. The analog portions of the equipment are of relatively low accuracy and high speed, while the digital portions of the equipment are used for storage, control, and selection of data.

The digital map is represented in the form of lines on a contour map which are approximated by polygons, and tabulated in the form of the positions of successive vertices of each polygon. Certain important properties of contour lines, such as their nesting relationship, are used to afford a convenient method of accessing data from memory on positions of contour polygons and their sides. Easily implemented algorithms are developed for continuous interpolation of height and reflectivity as the simulated radar "scans" this map with the speed of light, while "flying" with the speed of an aircraft. Just as the digital portions of the equipment are used to select data for setting and controlling the analog portions of the equipment, so the analog portions perform calculations to generate controls for the digital portions. For example, whenever a contour line is crossed by the radar sweep as it traverses the digital map of the terrain, the sign of an analog signal changes, causing activation of a digital circuit which selects data on the next contour and feeds it in turn to the analog circuitry.

This hybrid model, based on a close marriage of analog and digital computing techniques and designed to exploit the characteristics of a wide class of airborne radars, reduces the severity of the radar simulation problem by three orders of magnitude. Computing times that would be in the order of nanoseconds (10^{-9}) by a straightforward digital approach are relaxed to microseconds (10^{-6}), and storage devices that would hold gigabits (10^{9}) are reduced to megabits (10^{6})—within the state-of-the-art of modern computing equipment.

Such a digitally oriented simulator will achieve all the desirable advantages of digital equipment, such as repeatability, flexibility, expandability, and programmability. The basic limitation on the capabilities of the simulator becomes the economics of user needs. For example, the precision of representation of a digital word can be made larger merely by adding additional digit positions. In analog equipment there is a physical limitation imposed by the resolution that can be obtained by a photographic reduction or relief map of a portion of the earth's surface. Thus, a simulator which is basically digital in nature, using the techniques described in this paper, removes or relaxes some of the stringent physical limitations on simulators designed to mimic a diverse class of radars and areas of terrain. The physical limitations are replaced by economic limitations on the amount or complexity of equipment required to achieve a multiple-purpose or universal simulator.

Having established the technological feasibility of the application of digital simulators to high-data-rate devices such as airborne radars, research is continuing in two fundamental associated areas. Digital maps and encoding methods are now being studied to determine the information content of a map, or the theoretical minimum number of bits required to represent the data from which the radar display is built. In addition, research is being carried out on design tradeoffs and optimization of conflicting cost-performance requirements, in order to formulate definitive principles for the logical design of efficient hybrid computing systems.

STATEMENT OF THE PROBLEM

It is generally far less expensive to utilize a special-purpose simulator for training of personnel in the operation of radars and interpretation of radar displays, than to fly special missions for this purpose. For effective training, the radar simulator must be integrated with the weapon system simulator which is piloted in real time, and thus photographic reproductions of previous flight plans are invalid. Further, training may be required for missions that cannot be physically flown in advance.

A generally stated goal or problem for advanced radar land mass simulation is to generate signals to mimic the display presented by an airborne ground-mapping radar having a range of 200 miles, range resolution of 60 points per nautical mile, mounted in an aircraft flying at speeds up to Mach 3 and altitudes up to 10 miles, over a problem area of 600×1200 miles. The signals for the display equipment of the radar must be produced in real time, and they must provide accurate replicas of the actual radar returns based on the instantaneous position and orientation of the aircraft and an up-to-date knowledge of the terrain height, radar reflectivity, etc.

Of the various effects observed by the radar, the most difficult to reproduce appears to be the height profile as a function of ground range, which must be synchronized with the radial radar sweep pattern. Other effects such as reflectivity, attenuation, moving targets, and the like are second-order problems, to be superposed on the terrain profiles proper.

Historically, two generations of simulators for airborne mapping radars used respectively optical and ultrasonic analogs to the actual radar. That is, a transducer modeling the radar antenna was physically moved over a relief map, and the usual problems such as linearity, accuracy, size of model, and the like limited the accuracy of simulation to that of the earlier airborne radars.

The analog device currently in vogue is the "factor transparency", used in an equipment in which flying-spot scanners inspect photographic plates. Each plate carries a separate factor of interest to the radar—height, reflectivity, etc.—in the form of density modulation. Again there is a limitation on the number of resolvable points that can be stored on a manageable photographic plate, and on the accuracy with which they can be scanned in synchronism with an arbitrary radar to be simulated.

Cursory solution of the problem, based on straightforward use of digital computers, leads to a requirement for nanosecond (10^{-9}) computing times and tens of gigabits (10^{10}) for total memory capacity. Accordingly, a study was initiated of a digital simulator, designed to reproduce height and reflectivity factors in synchronism with the radar display. The resolution was to be considerably better than that of the pure analog devices, but no attempt was made to incorporate special effects or the "second-order problems" mentioned. That study gave rise to the overall logical design discussed herein.

Digital techniques provide not only potential improvement of accuracy and resolution, but also greater ease of changing terrain models and aligning them properly by storing the terrain model on, say, magnetic tape and performing digital selection and processing of the data. Desired changes in simulated radar parameters are inserted with the facility of a "patch" in a program.

In describing the simulator a basic unit of length is chosen, and all operations are referenced to it. The length unit, named a, is defined as equivalent to 14 microseconds of radar round-trip travel time, or approximately 7/6 nautical miles. In the computer the problem area contains on the order of 2×10^9 resolvable points. The area displayed in a single radar scan (several hundred profile sweeps) includes on the order of 10^8 resolvable points, and the area to be added for each new scan (roughly 1 per second) is in the order of 10^6 resolvable points. For each radial sweep of length 188 a (corresponding to 200 miles) the radar sees 64 resolvable points per unit length a, for a total of 12,000 resolvable points per sweep.

The problem can be looked upon as one of selection of data for a particular a by a square, from the problem area measuring 512 a by 1024 a, followed by the reconstruction of the terrain height profile to lateral resolution of $a/64$ as the sweep passes through that square. Note that if the sweep just cuts the corners of such squares (or passes diagonally through the square), the maximum rate of data access is given by the 20 microseconds required to traverse the diagonal of one square and an infinitesimal piece of a square adjacent to it (or diagonally through 2 squares). This averages one square in 10 microseconds, and it represents a compromise between speed of access and length of word required of a high-speed memory. The value of a represents a tradeoff in the equipment, analogous to scale factor of a map.

The resulting digital-analog (hybrid) simulator must employ techniques that exploit the redundancy and repetitiveness found in the display signals produced by a scanning radar. These features significantly reduce the requirements on computational speed and memory capacity, and they have been found to be useful in the reconstruction of the terrain height profiles at the accuracy and resolution of advanced scanning radars.

Redundancy is apparent in several guises. In the terrain it is apparent that heights of points a fraction of a mile apart have a great deal of correlation, and a good estimate of the height of an arbitrary point is obtained by a straightforward interpolation from points which are a significant distance from it. This feature is not so true in cavernous cities and in the Grand Canyon, but natural terrain generally tends to be reasonably smooth and flat. For example, in a problem area 512 a by 1024 a it is expected that the height variation will not exceed 4 a. Further, in cultural areas (e.g., cities) the defining features tend to be lines (streets, railroads, boundaries, etc.), which are not truly random, but in fact tend to form continuous, smooth curves. In short, the number of resolvable elements of length or lateral position is effectively lower than the 2×10^9 grid points derived above, and the number of resolvable reflectivities is even lower yet.

Additional redundancy or repetitiveness is introduced by the radar and the vehicle in which it is mounted. For one thing, the aircraft moves extremely slowly compared with the speed of the radar sweep (traverse of the length unit a by a Mach 3 aircraft takes two seconds). There are great similarities among successive frames or scans, although no two range sweeps reconstruct the same terrain profiles. In short, a large area is displayed, but a small amount of new area is added each successive scan. However, there are variations in shadowing phenomena, and there are radar effects due to aspect (angle of the line of sight with respect to a normal to the surface seen) that change considerably from scan to scan, even though the basic terrain data varies slowly by comparison.

There are limitations on the rapidity with which an aircraft can maneuver, and reasonable predictions may be made on the portions of data the simulator will need a short time in the future, based on the present aircraft heading and knowledge of its maximum turn and climb rates. These predictions caused by maneuverability limitations or redundancy allow, for example, the advance organization of future data ("look-ahead") with subsequent saving in data access time.

Again, some of the redundancy in the data is due to the radar itself. The antenna of a scanning radar sees near points with relatively high resolution in the cross-range dimension, but this resolution decreases linearly with range, because of the constant angular beamwidth of the antenna. Accordingly, the longer range data is smeared as the radar sees it, and the information content per unit area of terrain is reduced with increasing distance from the radar. Similarly, since the elevation angle with which the radar beam sees the terrain becomes more and more a grazing angle at long range, fine detail is lost and longer shadows occur at longer range.

Finally, the ultimate display places a limit upon the number of resolvable points that can be portrayed at any instant. No modern radar display has the capability of maximum resolution at maximum range at which the radar is capable. Rather, the operator can manipulate the display to show a portion of the area in view at certain desired magnifications. In every case there is a limitation on the total number of resolvable points that can be displayed, although the operator has the option of switching display centers and scales almost at random. Thus the simulator needs the capability of high resolution at all ranges and azimuths, but the operator selects the data to be displayed only as a subset thereof.

PRESENT APPROACH

The present approach utilizes the capabilities of digital storage, transmission, and processing devices for storing, retrieving, organizing, and preparing radar data for ultimate display. It takes advantage of the high-speed computing capabilities of analog devices where the simulator calls for successively processing different data in accordance with the same mathematical transformations. The approach also takes advantage of several techniques for exploiting redundancy of the data, in format, in algorithms, and in implementation. The techniques are noted here, and described below is the overall computer organization embodying them.

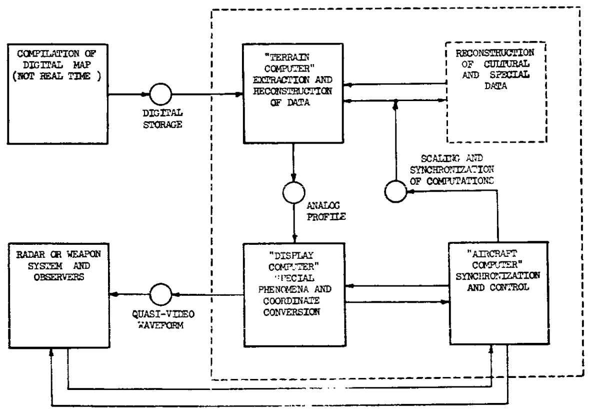

COMPILATION OF DIGITAL MAP (NOT REAL TIME)
DIGITAL STORAGE
"TERRAIN COMPUTER" EXTRACTION AND RECONSTRUCTION OF DATA
RECONSTRUCTION OF CULTURAL AND SPECIAL DATA
SCALING AND SYNCHRONIZATION OF COMPUTATIONS
ANALOG PROFILE
RADAR OR WEAPON SYSTEM AND OBSERVERS
QUASI-VIDEO WAVEFORM
"DISPLAY COMPUTER" SPECIAL PHENOMENA AND COORDINATE CONVERSION
"AIRCRAFT COMPUTER" SYNCHRONIZATION AND CONTROL

Figure 1. Organization of Digital Radar Land Mass Simulator.

The functional organization of the real-time simulator is pictured in Fig. 1. It is designed to work closely with the display portions of real radars which are to be simulated, and it consists of three conceptual divisions: "aircraft computer", "terrain computer", and "display computer". The purposes and functions of each are discussed below.

The terrain computer is by far the most complex; accordingly, it receives the most attention in this paper. Basically, it transforms a terrain map in the form of *digital storage* of data to appropriately sequenced information on terrain height, reflectivity, etc., in the form of *analog profiles*. By means of the aircraft computer, the profiles are synchronized with the radar scan pattern and controlled in accordance with aircraft motion; the terrain computer feeds signals to the display computer for preparation of waveforms for the radar equipment.

The terrain computer can further be divided into its major conceptual components, as shown in Fig. 2. This organization and terminology has evolved as being useful for discussion but need not represent the physical or functional segregation of any actual equipment. The fuller

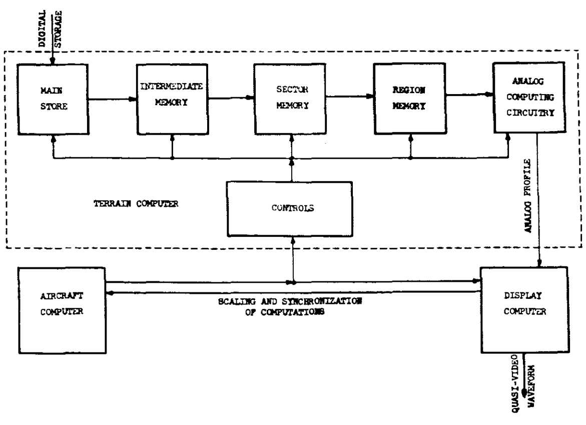


Figure 2. Organization of the Terrain Computer.

descriptions of these components, their purposes, and their implementation, and the finer logical and organizational details of the radar simulator, are covered in succeeding portions of the present paper.

A portion of the terrain having a standardized size, shape, orientation, and amount of data, is termed a *region* (the basic region size is a by a, and larger regions of the terrain are in multiples of this size). For the application at hand, three region sizes have been adopted; size A regions are $a \times a$, size B regions are $4a \times 4a$ and size C regions are $16a \times 16a$. The amount of data in a region, whatever its size, is relatively constant for any given type of terrain (city, plains, mountains, etc.). For those relatively small portions of the problem area on which high-resolution data can be obtained by radar, size A regions are utilized. When lowest resolution is sufficient, size C regions are utilized, with a consequent saving in amount of data per unit area of terrain. Fig. 3

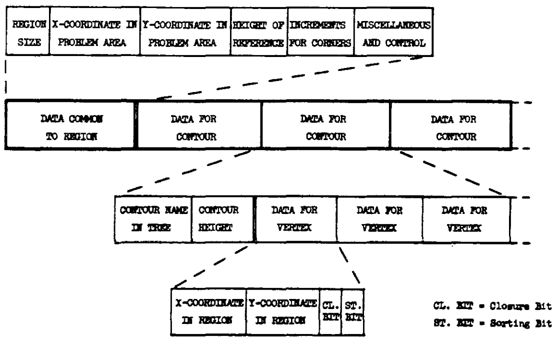


Figure 3. Format of Data on Any Region.

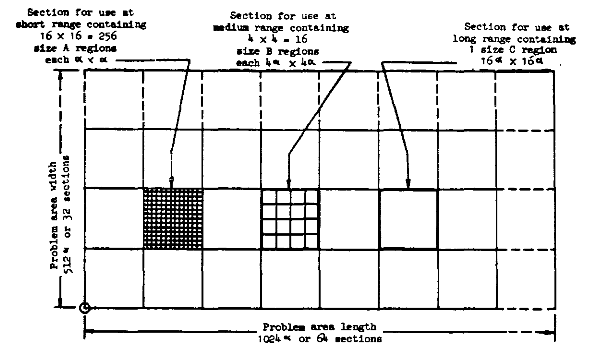

Section for use at short range containing 16 × 16 = 256 size A regions each α × α
Section for use at medium range containing 4 × 4 = 16 size B regions each 4α × 4α
Section for use at long range containing 1 size C region 16α × 16α
Problem area width 512α or 32 sections
Problem area length 1024α or 64 sections

Figure 4. Regions of Different Resolutions for Use at Different Radar Ranges.

illustrates the format of data for any given region, to be discussed in greater detail below.

A *section* of terrain, as shown in Fig. 4, is defined as the largest region size to be used. Thus a section, of dimension $16a \times 16a$, may contain data on one C region, 16 B regions, or 256 A regions. In the process of selecting data, the section corresponding to the portion of terrain needed is selected at the appropriate resolution, and then the regions within that section are selected. Up to this point there is no necessity for consideration of the format of data in the regions. Now, however, the simulator is concerned with obtaining the height profile corresponding to the position and orientation of the radar sweep.

The technique used to exploit the redundancy (relative flatness) of natural terrain, simultaneously allowing a simplified method of reconstructing height of the terrain at a series of closely spaced points, is that of *contour interpolation.* By analogy with a conventional topographic map, the height of any point can be interpolated between the height of the contour that immediately encloses that point, and the heights of those other contours that are also immediately enclosed. The continuum of points on the radar sweep can be found by interpolation based upon relative distances from the enclosing and enclosed contours. The crossing of a contour by the radar sweep can be sensed by a change in sign of the directed distance between the point in question and the contour. Fig. 5 shows how a profile may be reconstructed to any desired accuracy, given the heights and locations of all contour lines in a region.

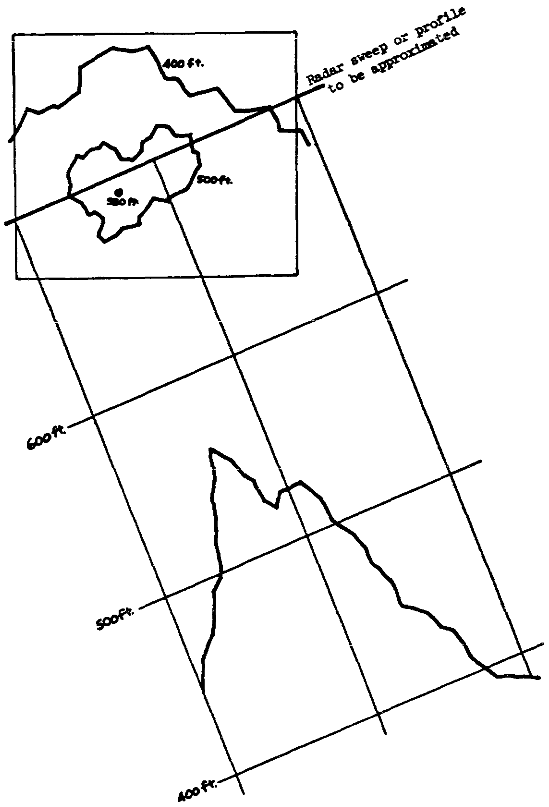


Figure 5. Reconstruction of a Vertical Profile from Horizontal-Plane Contour Lines in a Region.

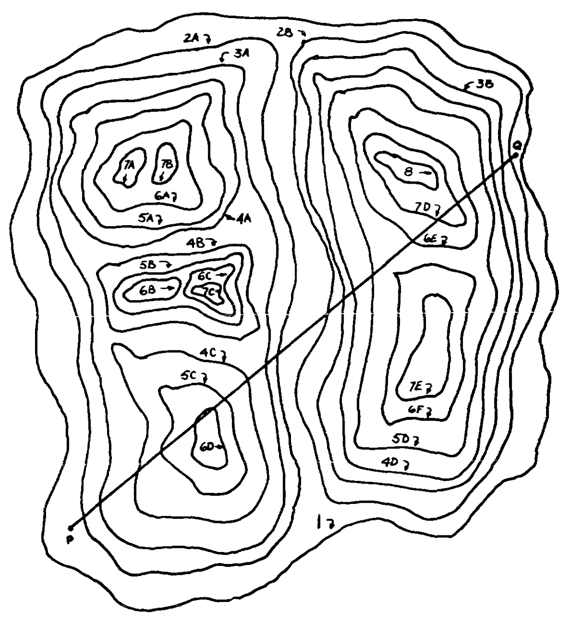


Figure 6. A Complete Contour Map.

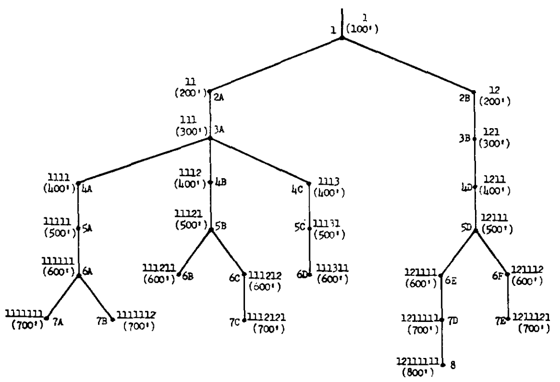


Figure 7. Enclosure Tree Corresponding to Preceding Figure.

Contour lines on a single-valued surface have the particularly convenient property that they form nested closed curves. As shown in Fig. 6, any given contour line encloses an arbitrary number of other contour lines, but is in turn enclosed by only one contour line, and these lines never intersect. (A saddle point can result in two contour lines being tangent, but they can still be treated as two separate closed curves which touch but do not intersect each other). This enclosure relation can be represented in the form of a tree. As shown in Fig. 7, any given contour is enclosed by the contour above it on the tree and immediately encloses all those contours whose branches emanate from that node on the tree.

Thus when a contour is sensed as being crossed from the outside to the inside, the position on this tree moves down one branch. When a contour is crossed from inside to outside, the position moves up one branch on the tree. Specifically, one can take the representative radar sweep drawn on Fig. 6 and trace out the successive branches on the tree of Fig. 7, representing successive memory accesses to information on enclosing and enclosed contours at any point of the radar sweep. Thus having selected the *section* and then the *region* of interest at any instant, and having determined within which *enclosing contour* the radar sweep point lies at any instant, it remains to find the distance between that point and the enclosing and enclosed contours. This can be done by approximating with specified algorithms each contour line to an arbitrary accuracy using linear segments and storing in digital form the *positions of the vertices of the polygon* thus constructed. The instantaneous distance between the point and any contour can be taken as the minimum of the distances between that point and all segments composing the polygon that approximates that contour. Format-controlling digits are included to signify a contour *leaving the edges* of the map and reappearing later, and to signify the *termination of a list of vertices* representing the polygon that approximates any given contour in the enclosure tree.

The foregoing description leads directly to a reasonable hybrid implementation. The storage of information on the terrain can be performed in a digital manner, the section, region, and contour data being digitally addressable. On the other hand, the relatively low-accuracy but high-speed computations for interpolation of distance as the radar sweep point moves across a region, can be performed in an analog manner. That is, while regions can be picked up at a maximum rate of 10^5 to 10^6 per second, successive interpolated points within the regions must come out at fractional microsecond increments. There are at most 512×1024 such regions, regardless of the data content, but the distance interpolation requires only 6 or 7 bits of precision.

The digital storage of contour polygon vertices enables a significant compression of the amount of data required to represent terrain height. The ultimate display is desired in analog form; the interpolation procedure among digitally described contours can be rather simply performed with high-speed, medium-accuracy, parallel analog equipment.

LOGICAL ORGANIZATION

A radar simulator using the foregoing techniques may conceptually be divided into three parts, representing the computations based on the *aircraft*, the *terrain*, and the *display*. The "aircraft computer" has facilities enabling it to keep track of the velocity, position, and orientation of the aircraft with respect to the terrain. The effect of wind on the position of the aircraft, and the overall response of the system to changes in the aircraft position that are initiated by the pilot, are handled by the aircraft computer.

The "terrain computer" provides the data concerning height (or reflectivity, or other factors) for the portion of the terrain within the range setting of the radar, utilizing as input the position of the aircraft, its orientation, and the direction of the radar sweep with respect to the aircraft. The terrain computer consists of a main store, some buffer memories, and the high-speed computational components. The terrain computer, described in greater detail below, provides analog representations of terrain profiles in synchronism with the radar sweep. (These relations are diagrammed in Figs. 1 and 2.)

The "display computer" transforms the data provided by the terrain computer to the final display after making the necessary modifications. Its computations consist of those related to shadow determination, weather noise, antenna pattern, etc. The display computer also handles the weighting of data from calculations on regions of different size, to allow a smooth transition during the decrease in resolution from minimum to maximum radar range.

While the aircraft computer constitutes control and synchronization, and the display computer constitutes rather straightforward analog transformations, the terrain computer does the actual reconstruction of the terrain profiles in a hybrid implementation. With reference to Fig. 2, the terrain computer may be broken down further to illustrate the techniques embodied in the design.

The Main Store contains the source data in digital form for the complete problem area. The Intermediate Memory contains the data on the terrain within the radar range, only at the necessary resolution. The Sector Memory stores data for several adjacent radar *sweeps* (resolvable azimuth increments) within the radar scan. Controls are provided for selection of data from the Main Store to the Intermediate Memory, and from the Intermediate Memory to the Sector Memory, as well as for selection of data on successive regions from the Sector Memory for the High-Speed Computations. This overall logical design is shown in Fig. 8, with successive figures used in this paper for convenient exposition. (The data preparation process and the evolution of this model are discussed in detail in the referenced report.)

For the problem area of dimensions $512\ a \times 1024\ a$, the Main Store consists conceptually of two serial-access units. As shown in Fig. 4, one unit of the Main Store contains 32 east-west

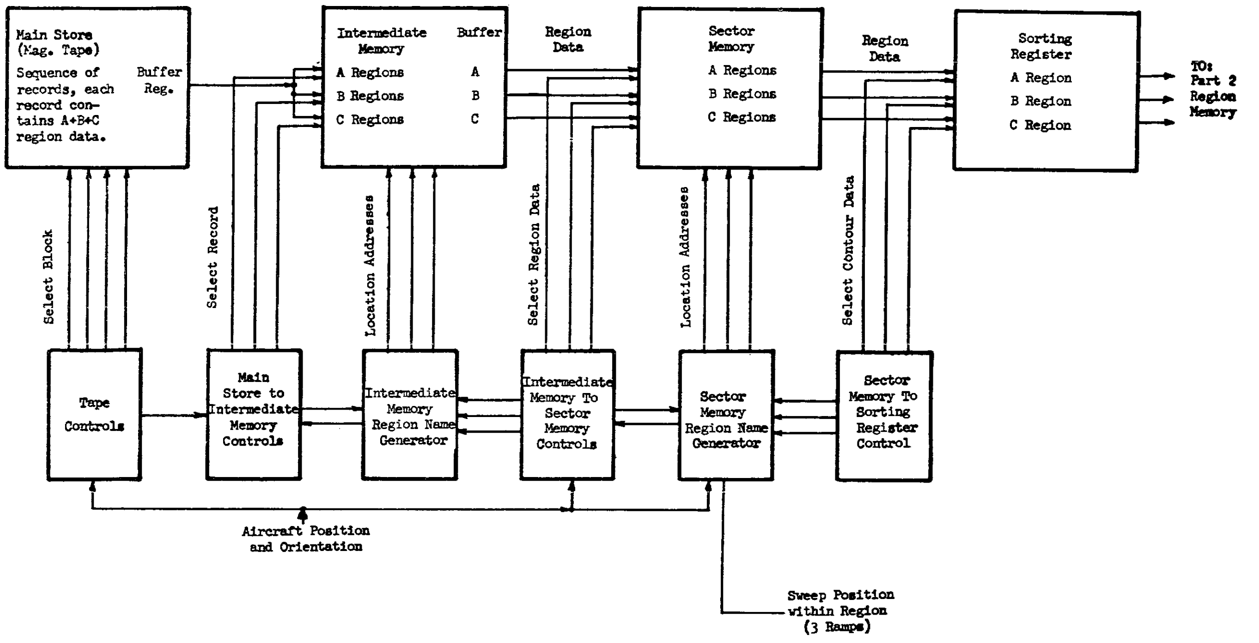

Main Store (Mag. Tape)
Sequence of records, each record contains A+B+C region data.
Buffer Reg.
Intermediate Memory
Buffer
A Regions A
B Regions B
C Regions C
Region Data
Sector Memory
A Regions
B Regions
C Regions
Region Data
Sorting Register
A Region
B Region
C Region
TO: Part 2 Region Memory
Select Block
Select Record
Location Addresses
Select Region Data
Location Addresses
Select Contour Data
Tape Controls
Main Store to Intermediate Memory Controls
Intermediate Memory Region Name Generator
Intermediate Memory To Sector Memory Controls
Sector Memory Region Name Generator
Sector Memory To Sorting Register Control
Aircraft Position and Orientation
Sweep Position within Region (3 Ramps)

Figure 8. Overall Logical Block Diagram of the Simulator—Part 1.

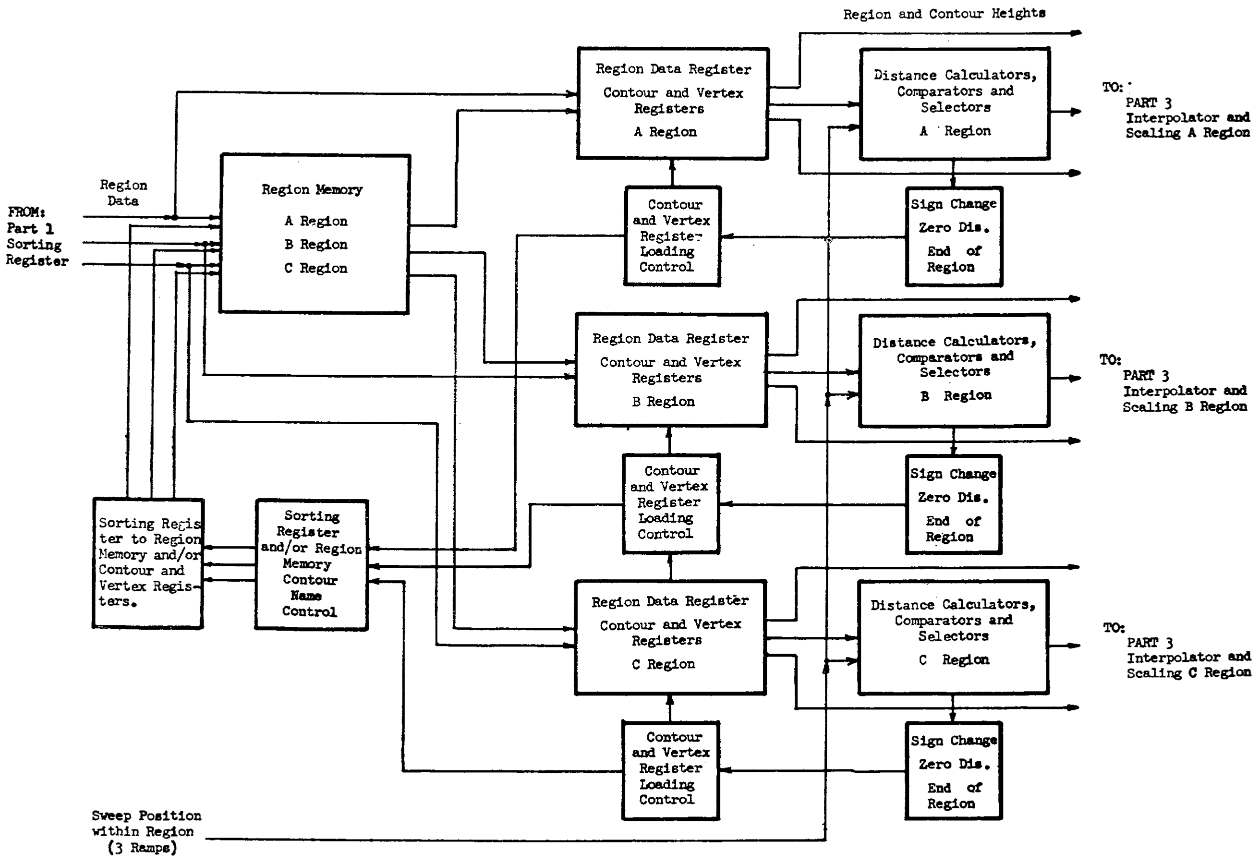

Region and Contour Heights
FROM: Part 1 Sorting Register
Region Data
Region Memory
A Region
B Region
C Region
Region Data Register
Contour and Vertex Registers
A Region
Distance Calculators, Comparators and Selectors
A Region
TO: PART 3 Interpolator and Scaling A Region
Contour and Vertex Register Loading Control
Sign Change
Zero Dis.
End of Region
Region Data Register
Contour and Vertex Registers
B Region
Distance Calculators, Comparators and Selectors
B Region
TO: PART 3 Interpolator and Scaling B Region
Contour and Vertex Register Loading Control
Sign Change
Zero Dis.
End of Region
Sorting Register to Region Memory and/or Contour and Vertex Registers.
Sorting Register and/or Region Memory Contour Name Control
Region Data Register
Contour and Vertex Registers
C Region
Distance Calculators, Comparators and Selectors
C Region
TO: PART 3 Interpolator and Scaling C Region
Contour and Vertex Register Loading Control
Sign Change
Zero Dis.
End of Region
Sweep Position within Region (3 Ramps)

Figure 8. Overall Logical Block Diagram of the Simulator—Part 2.

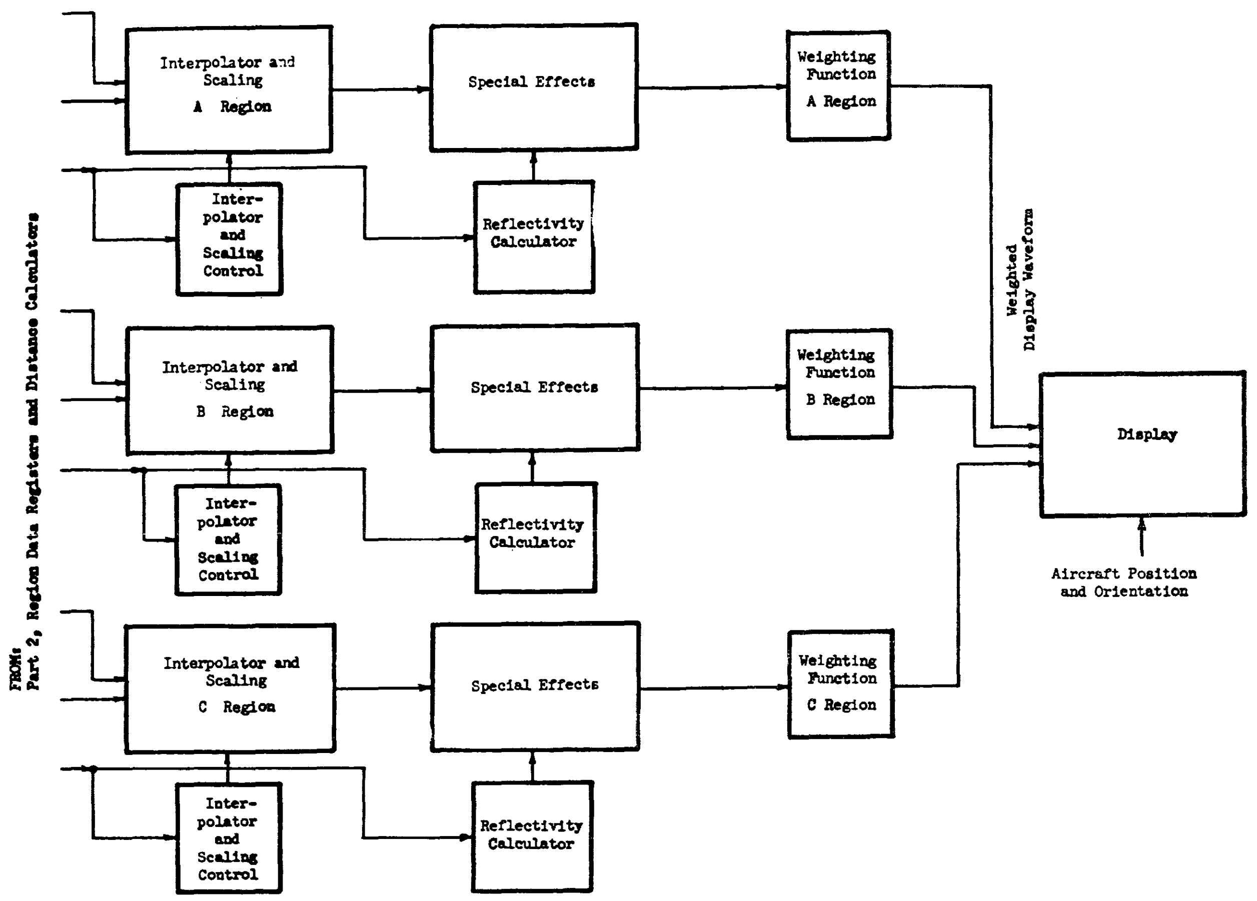

FROM: Part 2, Region Data Registers and Distance Calculators
Interpolator and Scaling A Region
Special Effects
Weighting Function A Region
Inter-polator and Scaling Control
Reflectivity Calculator
Interpolator and Scaling B Region
Weighting Function B Region
Interpolator and Scaling C Region
Weighting Function C Region
Weighted Display Waveform
Display
Aircraft Position and Orientation

Figure 8. Overall Logical Block Diagram of the Simulator—Part 3.

strips of terrain, each containing 64 sections at all three levels of resolution. The other unit of the Main Store contains 64 north-south strips of terrain, each containing 32 sections at the three levels of resolution. Thus if the aircraft component of motion is primarily northward, an east-west strip is selected; if the aircraft motion is predominately eastward, a north-south strip of terrain data is selected. Some statistics have been obtained on the amount of data per region required by the contour interpolation method to obtain the resolution specified above, and it appears that this amount of data for the overall problem area will fit comfortably on a conventional digital magnetic tape. (The mechanics of the contour interpolation method are also discussed in the referenced report.) Similarly, the aircraft can fly the width of one strip (16 *a*) in only half a minute, and there is sufficient time to move the magnetic tape at conventional speeds to select information from any *section* that is coming within radar view. The storage of data in the form of contours affords more than an order of magnitude reduction in data, over tabulation of the heights of all potentially resolvable points in the problem area.

The Intermediate Memory is fed with data from the Main Store, including only those sections whose resolutions correspond to the resolution desired up to the given radar range. As shown in Fig. 9, the Intermediate Memory contains data on 4129 regions, divided among sizes A, B, and C. While conceptually consisting of three sub-memories, it is doubly cylindrically symmetric, so that information added at one side of any sub-memory overrides the information contained at the other.

As the Main Store is serially accessed for sections in, say, an east-west direction, those sections due north of the center of the Intermediate

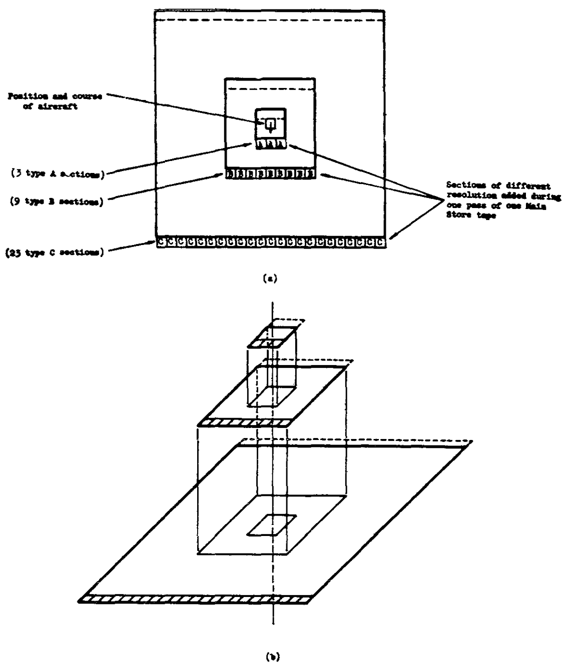

Position and course of aircraft
(3 type A sections)
(9 type B sections)
(23 type C sections)
Sections of different resolution added during one pass of one Main Store tape
(a)
(b)

Figure 9. Geometric Representation of the Three Digital Maps of the Intermediate Memory.

Memory (corresponding to the aircraft position) feed the Intermediate Memory with data for regions of all three resolutions. A little east and west of the due-north point, the Intermediate Memory is fed with only B and C regions. Finally, at the extremes of the radar scan, where only low-resolution data is anticipated, only size C regions are taken from the Main Store and fed to the Intermediate Memory. Actually the Main Store is organized so that it needs to be traversed once in an east-west direction for the short-range A regions, intermediate-range B regions, and long-range C regions, the range measured in the north-south direction. That is, for any section of terrain within which the aircraft is located, the Main Store east-west unit superposes information on A regions just north and just south of that section, B regions several strips north and south of that section, and C regions the maximum distance north and south of that section. (Actually for convenience of control in the presence of operator manipulation of scale factors and ranges, it turns out to be easier to divide each unit of the Main Store into two units, so that there are a total of four serial access units, each corresponding to motion of the aircraft predominately to one point of the compass.) Data from the Main Store is obtained at about 10^3 regions per second. As the aircraft flies, the tapes move to furnish data for loading the Intermediate Memory at three different resolutions simultaneously, for three different radar viewing ranges.

This retention of data in the Intermediate Memory only to the resolution at which it is expected to be needed (by extrapolation of the aircraft motion) allows a significant reduction in the size of the Intermediate Memory. Realizing that data on regions may be required at quite a rapid rate, it is necessary to keep the Intermediate Memory as small as possible. While it represents about 20 percent of the entire problem area, the technique of using three different resolutions allows the Intermediate Memory to retain at any instant only 1 percent of the data in the Main Store—a factor of 20 in amount of data. The data is placed in the Intermediate Memory section by section, the appropriate resolution level being selected at entry corresponding to radar range to each section. However, the data is transmitted from the Intermediate Memory to the Sector Memory region by region, at the appropriate region size —A, B, or C—with a maximum output rate of about 10^4 regions per second.

The Sector Memory makes a further transformation of speeds and capacities by recognizing that several successive radar sweeps will pass through any given region. Averaged over the three different region sizes specified, the "region usage factor" is about eight. That is the data for an arbitrary region will be used on the average eight times during the course of profile reconstruction for successive radar sweeps within a scan. The Sector Memory thus retains information on a sector of the scan (about 125 regions corresponding to about 65 successive sweeps, divided among the three region sizes or resolutions).

Information enters the Sector Memory at about 10^4 regions per second, but is fed from the Sector Memory to the High-Speed Computations at a maximum rate of about 10^5 per second. As noted above, these speeds are scaled to the length unit a, equal to distance traversed by a two-way radar sweep in 14 microseconds, and the synchronizing signals for these computations are based upon this scale factor.

Selected from the Sector Memory is the data for the *region* containing the profile line to be traced at any instant, in the format of Fig. 3. The reconstruction of the profile consists of computing the terrain height at the coordinates of the sweep path as it traverses the map, on and between contour lines. The contours are approximated by polygons. Hence any arbitrary points of the sweep is either on the approximated contour or between two contours. If the point is on a contour, the height is obviously the height of the contour, which is part of the data for the region at hand. If the point is between contours, the minimum distance to all segments of these contours in that region is calculated, and interpolation is performed between the contour heights using these distances as constants of proportionality.

The limitation is artificially imposed that no contour may immediately enclose more than three other contours, this limitation being implemented by inserting extra contours where necessary in the data preparation phases of the Main Store. The makeup of the data for any region, as shown in Fig. 3, includes the *closure bit,* used to signify a contour leaving the edge of a region and returning to it elsewhere; there is no line segment within the region between the vertices so marked. The *sorting bit* is used to signify the end of a list of vertices for any one contour.

Fig. 10 diagrams the computation of minimum distances to one enclosing and three immediately enclosed contours, upon knowing where in the contour enclosure tree (of any region) a point on the sweep path lies. Fig. 11 derives the mathematics of determining the distance between a point and a line segment, this

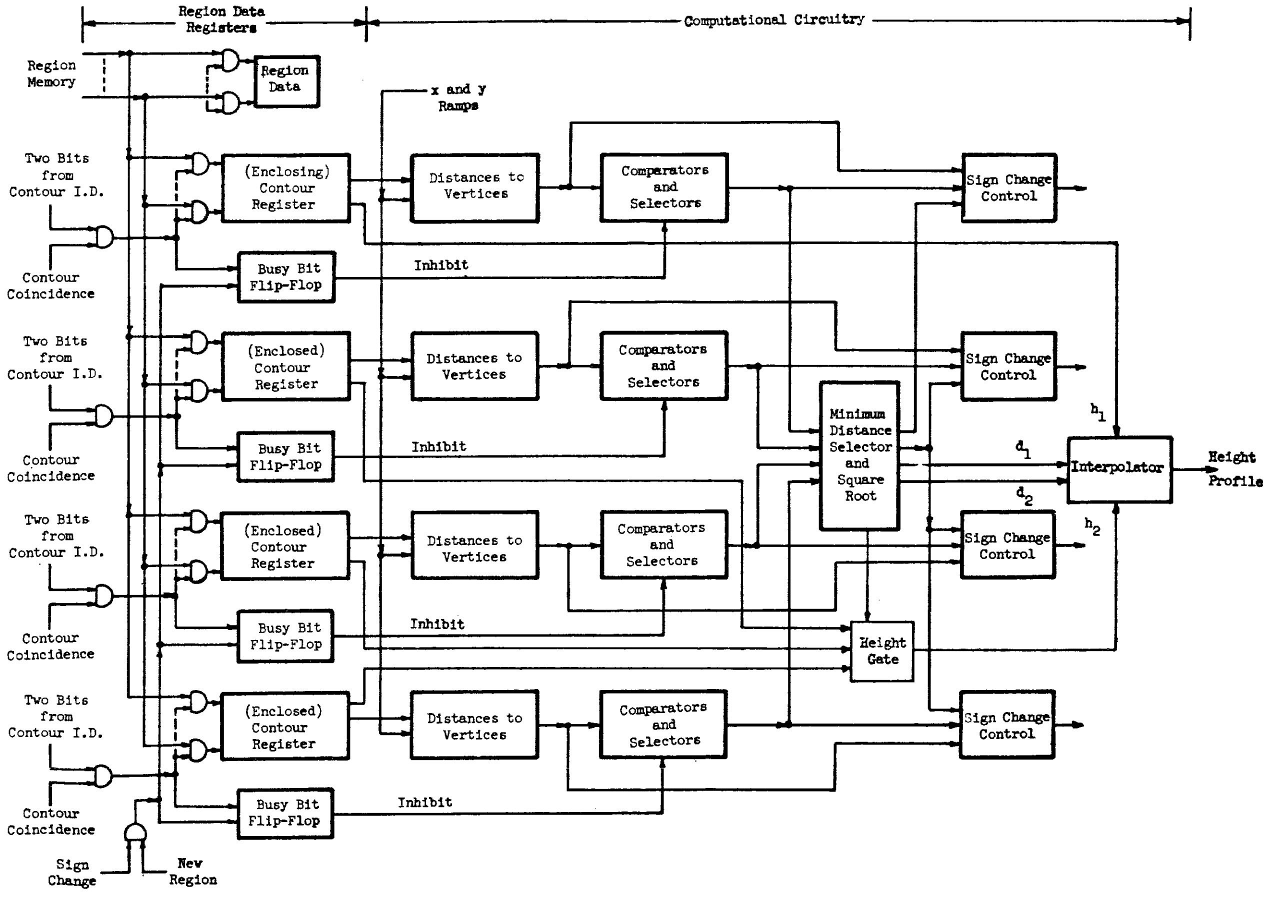
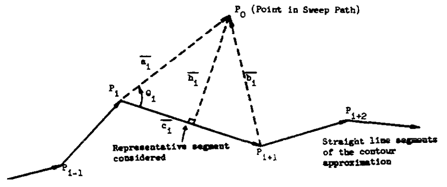


Figure 10. Block Diagram of Region Data Registers and Computational Circuitry.

computation being replicated for all line segments of the contour in the region. Finally, Fig. 12 shows a block diagram of the combined analog/digital equipment that may be used for implementing these distance-measuring and height-interpolating algorithms.

It is seen that the digital storage of contour polygon vertices within a region (with format and control information to define the enclosure tree structure) is sufficient to set analog computing equipment as any given region is entered by the radar sweep. The analog equipment then utilizes relatively low-accuracy signals for the position of the radar sweep point within that region, and calculates in a continuous manner the distances to enclosing and enclosed contours, from which the height is continuously calculated by combined analog/digital interpolation. The output profile is obtained


P_0 (Point in Sweep Path)
$\overline{a_i}$
$\overline{h_i}$
$\overline{b_i}$
P_i
θ_i
P_{i+2}
$\overline{c_i}$
Representative segment considered
P_{i+1}
Straight line segments of the contour approximation
P_{i-1}

Points:

P_0 = a point in the radar sweep path specified by coordinates (x_0, y_0).

$P_{i-1}, P_i, P_{i+1}, P_{i+2}$ = points or vertices of a contour polygon specified by coordinates (x_{i-1}, y_{i-1}), (x_i, y_i), etc.

(x_b, y_b) = region reference coordinates

$(x_i - x_b, y_i - y_b)$ = the coordinates of vertex i, relative to the region reference coordinates

Vectors:

$\overline{a_i}$ = vector from P_0 to P_i of length $|\overline{a_i}|$

$\overline{b_i}$ = vector from P_0 to P_{i+1} of length $|\overline{b_i}|$

$\overline{c_i}$ = vector from P_i to P_{i+1} of length $|\overline{c_i}|$

$\overline{h_i}$ = vector of length $|\overline{h_i}|$, representing the altitude from P_0 to side $\overline{c_i}$

$\overline{n_i} = \overline{c_i} \times \overline{a_i}$ = vector directed normal to the paper of length $|\overline{n_i}|$

Definition of Terms:

$$\Delta x_i = (x_{i+1} - x_b) - (x_i - x_b)$$
$$\Delta y_i = (y_{i+1} - y_b) - (y_i - y_b)$$
$$\delta x_i = (x_0 - x_b) - (x_i - x_b)$$
$$\delta y_i = (y_0 - y_b) - (y_i - y_b)$$

Length of Vectors:

$$|\overline{a_i}| = \sqrt{|\delta x_i|^2 + |\delta y_i|^2}$$
$$|\overline{b_i}| = |\overline{a_{i+1}}|$$
$$|\overline{c_i}| = \sqrt{|\Delta x_i|^2 + |\Delta y_i|^2}$$

$$|\overline{h_i}| = |\overline{a_i}| \sin\theta = \frac{|\overline{n_i}|}{|\overline{c_i}|} = \frac{\Delta y_i \delta x_i - \Delta x_i \delta y_i}{\sqrt{|\Delta x_i|^2 + |\Delta y_i|^2}}$$

$$|\overline{n_i}| = |\overline{c_i}|\,|\overline{a_i}| \sin\theta = \Delta y_i \delta x_i - \Delta x_i \delta y_i$$

The sign of the cross products of $|\overline{n_i}|$ above signifies whether P_0 is to the inside or outside of the segment, $\overline{P_i P_{i+1}}$, of the contour. When $|\overline{n_i}| = 0$, P_0 is on the line segment, $\overline{P_i P_{i+1}}$.

Selection of Minimum Distance:

The lengths of the vectors, $\overline{a_i}$, $\overline{c_i}$, $\overline{h_i}$ and $\overline{n_i}$, are computed. The distance from the point, P_0, to the line segment, $\overline{P_i P_{i+1}}$, is either the altitude, $|\overline{h_i}|$, of a triangle from P_0 or is equal to either the side $|\overline{a_i}|$ or $|\overline{b_i}|$. The vector dot product is used to determine which of the three cases is the correct distance.

$\overline{a_i} \cdot \overline{c_i}$ represents the projection of $\overline{a_i}$ on $\overline{c_i}$ and $= \Delta x_i \delta x_i + \Delta y_i \delta y_i$

If

a) $\overline{a_i} \cdot \overline{c_i} < 0$; the minimum distance from P_0 to $\overline{c_i}$ is $|\overline{a_i}|$.

b) $0 < \overline{a_i} \cdot \overline{c_i} < \overline{c_i}^2$; the distance is the altitude, $|\overline{h_i}|$, from P_0 to side $\overline{c_i}$.

c) $\overline{a_i} \cdot \overline{c_i} > \overline{c_i}^2$; the distance is $|\overline{b_i}| = |\overline{a_{i+1}}|$.

Figure 11. Vector Diagram and Derivation of Distance Computation.

with a frequency response approximating that of the video bandwidth of advanced airborne radars—several megacycles.

RECAPITULATION

A method has been described for real-time generation of topographic profiles for constructing simulated radar displays. The redundant features of natural terrain and radar systems are exploited, and the resulting overall logic design, as described by the illustrations, allows data compression of three orders of magnitude.

Many of the details of control signals and data formats within the memory hierarchy are not contained in this discussion. Neither are any of the details of preparation of data for the Main Store by line segment approximations to contour maps. (This material is included, however, in the referenced report.) The work reported herein is continuing, concerned with modification of the logical design of the simulator to take account of the effects caused by phenomena associated with real radars and real terrains. That is, many anomalous effects occur merely by virtue of the high frequencies at which radar signals operate, and of the peculiar characteristics of cultural areas and man-made structures. The preparation of maps of the locations of these phenomena from a radar point of view, and also the problems of reconstruction of the eventual display and not just the height profile, are reserved for future discussions.

Acknowledgement must be made to Dr. Morris Rubinoff and to other members of the staff of Pennsylvania Research Associates, Inc. who contributed to the project on which this paper is based. The exploiting of redundancy by a factor of 20 in the Intermediate Memory, by a factor of 8 in the Sector Memory, and by a factor of 10 in the terrain representation by contours, has brought the problem of real-time simulation of advanced airborne radars within the realm of technical feasibility, based on the present state-of-the-art of special-purpose digital and analog computing equipment.

Ref: "Interim Report Nr. 2: Investigation of Digital Techniques for Radar Land Mass Simulation", U.S. Naval Training Device Center technical report 1025-2, AD , incorporating in turn other pertinent references.

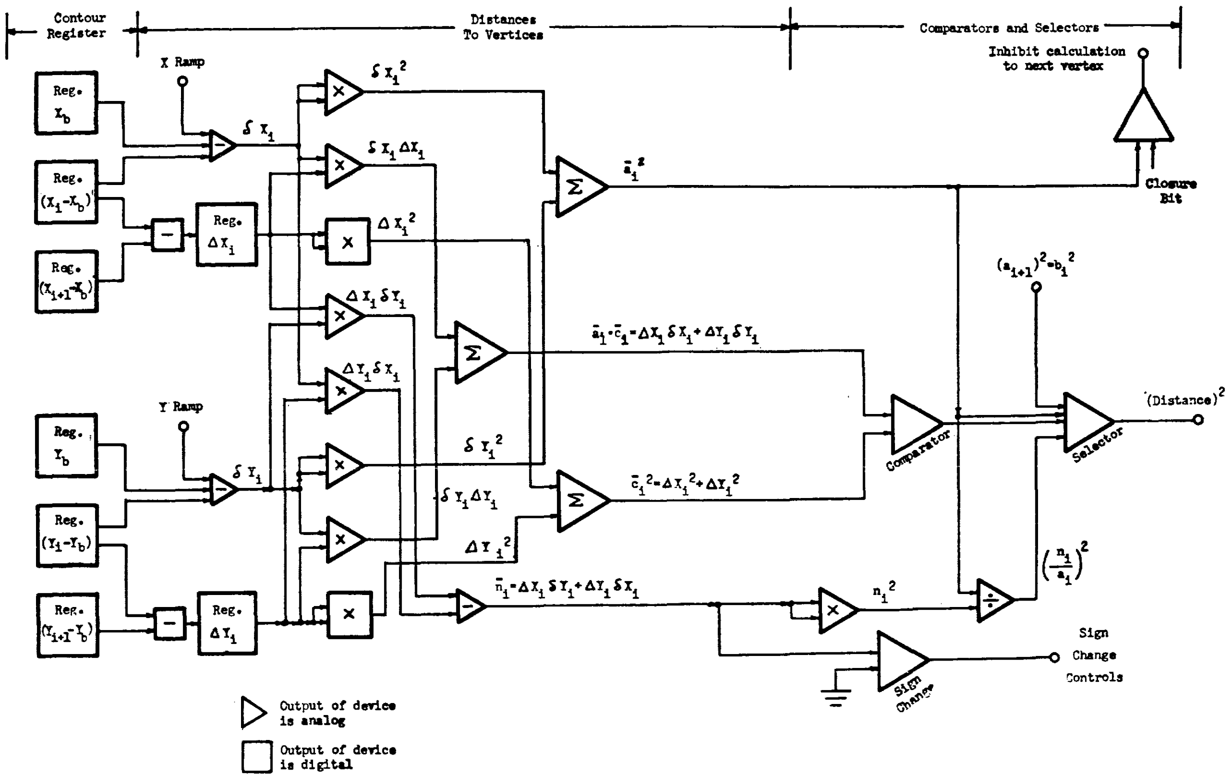


Figure 12. Distance Computation Circuitry.

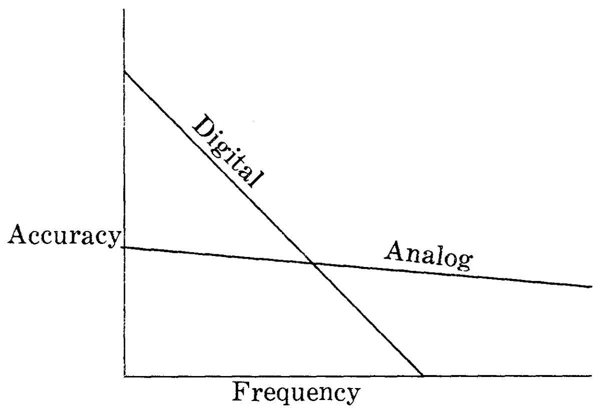
THE DES-1—A REAL-TIME DIGITAL SIMULATION COMPUTER

Max Palevsky and J. V. Howell
Scientific Data Systems, Inc.
Santa Monica, California

INTRODUCTION

Historically, there have been two lines of development in the attempt to apply digital techniques to the solution of those types of differential equations that have traditionally been solved by analog methods. The first started at Northrup Aircraft in approximately 1948 and led to the development of the digital differential analyzer.[1] The original digital differential analyzers were slow but, in context of the state of digital technology at the time, provided a computing method not otherwise available; in 1948, high-speed internally-programmed general purpose computers were not in general use. As digital techniques developed, digital differential analyzers were increased in speed and in complexity. The most recent of these, the TRICE, is a very high-speed parallel digital differential analyzer that operates at approximately ten thousand times the speed of early digital differential analyzers. The second line of development was started at the Moore School[2][3] in the investigation of general purpose digital techniques for operational flight trainers. This line of development has been carried on into the present at M.I.T. The emphasis has been on stability considerations and algorithms which would permit computers with a given speed to simulate high performance aircraft.

The DES-1 is an attempt to apply present day technology to the problem of differential equation solving in view of both of the lines of development. The need for this type of computer is shown in a qualitative way by the following graph.



Analog computers—high-precision DC analog equipment—have, for most problems, sufficient frequency response but insufficient accuracy, particularly (and this is not shown by the graph) if large systems of non-linear equations are involved. Digital techniques, on the other hand, are slow. If a third axis—cost—were shown, digital techniques with reasonable frequency responses would be prohibitively expensive. The area above and to the right of the two lines represents performance that is more and more in demand in view of the increasingly difficult problems posed by our expanding technology, and these demands are not met either by digital differential analyzers or by any other available techniques. It can be shown that high-speed digital differential analyzers, in fact, will not yield the performance of presently available medium-sized general purpose computers for highly non-linear sets of equations.

With this starting point, then, the DES-1 has been evolved.... "evolved" since the design has passed through a considerable number of stages starting from a large number of computing elements operating simultaneously, to the present solution which is a general purpose, digital computer with a structure suited to the classical analog problems. For example, the M.I.T. studies[4] have shown that, in using a digital computer for flight simulation, something approaching half the time is employed for the generation of arbitrary functions. The DES-1, therefore, has extensive Search commands that permit functions to be generated with a minimum number of instructions and at a very high speed. A 15-segment function of one variable, using unevenly-spaced increments, can be generated in 77 microseconds. In order to meet the frequency response requirements for simulation, the computer is extremely fast: ADD requires 1.75 microseconds, MULTIPLY requires 7 microseconds. Both of these times include all accesses and indexing.

The DES-1, then, is a general purpose computer with a specially oriented instruction set. It also has a special set of software and a control and display console. These three elements—computer, software, console—are integrated in such a way that the DES-1 "acts" like an analog computer. The central concept that unifies the design is the notion of an "operator." An "operator" is a set of instructions that performs an operation which is analogous to the operation performed by a computing element in an analog computer. The "operators" are processed serially so that the comparison between the DES-1 and an analog computer takes the form of a given amount of time for the DES-1 as compared to a given amount of equipment for the analog computer.

1. *Comparison with Analog Computers*

A breakdown of such a comparison is shown in Table 1. These numbers and types of operators are processed in approximately 11 milliseconds (which corresponds to about 90 integration cycles per second). Problems involving full-scale resonant frequencies of up to 6 cycles per second can be handled at 75 integration cycles per second with an error of less than one part in 10^5. The operators listed below require approximately four thousand words of memory.

2. *Operators*

The notion of an "operator" will now be described in detail. Note than an "operator" is effectively a subroutine and so need not be fixed. The "operator" concept merely permits the machine to be programmed using an analog language as opposed to the standard digital language of the general purpose computer. An "operator" is characterized by an output which is functionally related to an input. For example, if

$$\chi = \cos \theta,$$

χ is the output, θ is the input, and the "operator" is COSINE. If θ is assumed as the output of another operator, for example—operator 550, and this cosine operator is assigned an identifying number, for example—430, the operator in the example is specified as

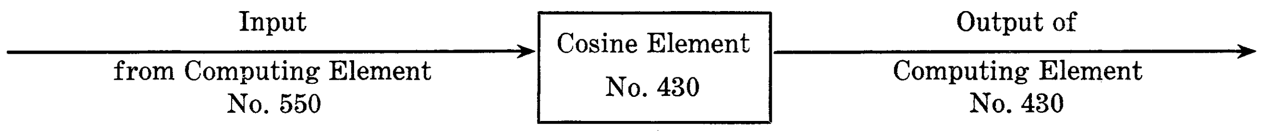
COS	550	430
↑	↑	↑
Operator	Input	Output Identification Number

Specification of operators in this form is also equivalent to a multiple address instruction in a digital computer. The operation portion of the operator, COSINE, may be considered an Operation Code, 550 may be considered the address of the input, and 430 the address of the output. Since the DES-1 is a digital device, the latter interpretation is more nearly correct. However, use of the "operator" programming structure lends itself to easy transition regardless of whether the user is analog-minded or general purpose computer-minded. The general purpose computer features of the DES-1 are provided to retain the inherent flexibility of digital computer programming which is particularly suitable for representation of discrete processes and problems involving logical decision and control. The DES-1 possesses a full complement of general purpose computer single address instructions.

Table 1. Direct Comparisons

Analog Computing Elements Type	*Qty*	*DES–1 Equivalents Type*	*Qty*	*Time per Operator μsec*	*Total Time msec*
Operational Amplifiers (30 summer-integrators, 45 summers, 25 inverters)	100	Sum (7 input)	130	14	1.82
		Integrator	30	80.5	2.415
Potentiometers	150	Multiplier	150	10.5	1.575
Multipliers					
Time division (10 independent)	20	Multiplier	20	10.5	0.210
Quarter square (10 independent)	30		30	10.5	0.315
Servo (10 independent)	50		50	10.5	0.525
Resolvers	5	Sine	5	80.5	0.4025
		Cosine	5	82.25	0.41125
		Arctan	5	171.5	0.8575
		Square Root	5	57.75	0.28875
Comparators including amplifiers and double-pole, double-throw switches	10	Comparator (2 switched inputs)	10	26.25	0.2625
Diode Limiters	15	Comparator	15	26.25	0.39375
Diode Function Generators (10 segment)	20	Function 1 variable (15 segment)	20	77	1.54
					11.00625

The specification of operators in this form is equivalent to a wire list giving the connections of computing elements in an analog computer as indicated below:



The basic operators are:

Sum
Scale
Multiply
Divide
Integrate
Sine
Cosine
Arctangent
Square Root
Function of One Variable
Function of Two Variables
Compare

The operators described here are based on fixed point, single precision (24 bits) number manipulation. However, the methods used can be extended to fixed point, double precision (48 bits) and floating point (39-bit mantissa and 9-bits exponent) at a reduction in speed. Methods used to implement these operators are based on a compromise between maximum speed and maximum generality and accuracy consistent with 24-bit representation of fixed point numerical information. It should be noted that all operators can, and should, exhibit flexibility in their mechanization. For special applications, it may be appropriate, for example, to use integration formulae other than the ones chosen here, or, if greater speed is essential, the number of terms in the sine series may be decreased to trade accuracy for greater speed.

All operators can be modified to suit special applications by general purpose programming techniques.

2.1 *Sum Operator*

The sum operator is specified as follows:

$$\text{SUM}, \pm x_1, \pm x_2, \ldots, \pm x_n, y.$$

Contents of the memory locations specified by $x_1, x_2, \ldots x_n$, are summed in accordance with the sign preceding each x_i. The result is stored in the location specified by y. The sum operator requires 1.75 $(n + 1)$ microseconds, where n is the number of inputs.

2.2 *Scale Operator*

The scale operator is specified as follows:

$$\text{SCALE}, x, n, y.$$

Contents of the memory location specified by x are binary-scaled and the result,

$$(y) = 2^n\, x(x)$$

is stored in the location specified by y. The scale operator requires 5.25 microseconds for execution.

2.3 *Multiply Operator*

The multiply operator is specified as follows:

$$\text{MUL}, x, y, z.$$

Contents of the memory location specified by x are multiplied by the contents of the memory location specified by y and the result stored in the memory location specified by z. The multiply operator requires 10.5 microseconds for execution.

2.4 *Divide Operator*

The divide operator is specified as follows:

$$\text{DIV}, x, y, z.$$

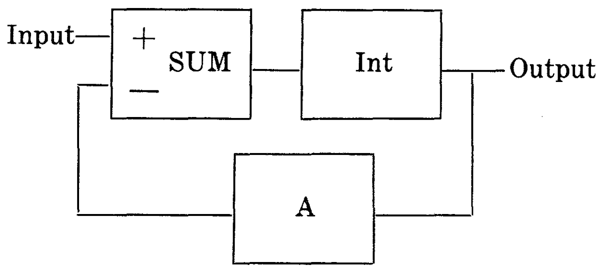
Contents of the memory location specified by x are divided by the contents of the memory location specified by y and the result stored in the memory location specified by z. The divide operator requires 22.5 microseconds for execution.

2.5 *Integration Operator*

The integration operator is specified as follows:

$$\text{INT}, \ddot{Y}, Y$$

Contents of the memory location specified by $\dot{Y}$ is the input to the selected integration formula. The integral is computed and stored in the memory location specified by Y. All integration operators are predictors; that is, inputs evaluated at time $t = nh$ (where h is the step size) are used to produce outputs at time $t = (n + 1)h$. The evaluation of all derivatives at $t = nh$ is insured by placing integration operators at the end of the integration cycle. Inputs to the integrator should not be taken from other integrator outputs. A sum operator or constant operator can be used to achieve separation. The integration sub-program can be modified by general purpose programming methods to obtain any numerical integration process. Integration formulae used in the standard integration sub-program are chosen for general application. Five integration methods are provided and selectable from the DES-1 console. These integration formulae are indicated in Table 2. Truncation errors are indicated for all formulae, except RK4, which has a truncation error of the order of h^5 but cannot be expressed in the same form as others. All formulae except RK4 make use of past information to achieve a form suitable for high-speed computation. However, the use of past information introduces extraneous roots in the integration. In the first order loop indicated below,



these extraneous roots can cause instability if the product Ah is too large. In practice, this means that difficulty is encountered if the step size h is made too large. Stability regions for each predictor formula is indicated below:

formula	stable for
P2	$Ah < 1$
P3	$Ah < 6/11$
P4	$Ah < 3/10$

Table 2. Integration Formula

P2 2 point predictor

$$y_{n+1} = y_n + \frac{h}{2}\left[3\dot{y}_n - \dot{y}_{n-1}\right] + \frac{5}{12}y'''(\theta)h^3$$

P3 3 point predictor

$$y_{n+1} = y_n + \frac{h}{12}\left[23\dot{y}_n - 16\dot{y}_{n-1} + 5\dot{y}_{n-2}\right] + \frac{3}{8}y^{iv}(\theta)h^4$$

P4 4 point predictor

$$y_{n+1} = y_n + \frac{h}{24}\left[55\dot{y}_n - 59\dot{y}_{n-1} + 37\dot{y}_{n-2} - 9\dot{y}_{n-3}\right] + \frac{251}{720}y^{v}(\theta)h_5$$

P4, C4 Adams-Bashford Predictor Corrector

$$y^{(p)}{}_{n+1} = y_n + \frac{h}{24}\left[55\dot{y}_n - 59\dot{y}_{n-1} + 37\dot{y}_{n-2} - 9\dot{y}_{n-3}\right] + \frac{251}{720}y^{v}(\theta)h^5$$

$$y^{(c)}{}_{n+1} = y_n + \frac{h}{24}\left[9\dot{y}_{n+1} + 19\dot{y}_n - 5\dot{y}_{n-1} + \dot{y}_{n-2}\right] - \frac{19}{720}y^{v}(\theta)h^5$$

Truncation error in corrector

$$\epsilon^{(c)}{}_T = \frac{19}{270}(y^{(p)}{}_{n+1} - y^{(c)}{}_{n+1})$$

RK4 4th order Runge Kutta

$$y^{(1)}{}_{n+1/2} = y_n + \frac{h}{2}\dot{y}_n$$

$$y^{(2)}{}_{n+1/2} = y_n + \frac{h}{2}\dot{y}^{(1)}{}_{n+1/2}$$

$$y^{(3)}{}_{n+1} = y_n + h\dot{y}^{(2)}{}_{n+1/2}$$

$$y_{n+1} = y_n + \frac{h}{6}\ \dot{y}_n + 2\dot{y}^{(1)}{}_{n+1/2} + 2\dot{y}^{(2)}{}_{n+1/2} + \dot{y}^{(3)}{}_{n+1}$$

The Predictor Corrector Integration Method, P4C4, is very useful for problems in which a linear relation between problem time and real time is not required. The integration cycle is divided into two subcycles. During the first subcycle, P4 is used in all integration operators to move ahead in time by h. During the second subcycle, C4 is used to obtain a second value of the integrated quantities. The truncation error in C4 is estimated as indicated in Table 2. If the truncation error in any integration operator exceeds a preassigned amount ϵ_T max, the step size is halved. If the truncation error in all integration operators is less than a second preassigned quantity ϵ_T min, the step size is doubled. The truncation error is controlled by use of the P4C4 integration technique to remain approximately within the range ϵ_T min to ϵ_T max by slowing the problem down (taking smaller steps when rates of change are large) and by speeding the problem up (taking larger steps when rates of change are small). For example, the Predictor Corrector technique can be applied to missile trajectory calculations, where rates of change are high during powered flight and re-entry but small during free flight, to achieve a significant net saving in run time.

The Runge Kutta Integration methods possess the property of requiring no past information. Runge Kutta methods are thus very useful in determining starting values for the other four integration methods. Starting values are computed by use of RK4 when the DES-1 is put into the RESET mode and during computation when step size changes are called for by P4C4. The RK4 method is also useful in obtaining an alternate problem solution for checking against any of the other integration methods. The integration cycle is divided into four subcycles during which new values of all derivatives are computed. In effect, this implies that a problem will run about four times slower using RK4 than when using any of the predictors. The use of RK4 is in general restricted to obtaining check solutions.

Execution times for integration methods are shown in Table 3.

Table 3. Execution Times for Integration

Method	*Time in μsec*
P2	42
P3	61.25
P4	80.5

Table 3—Continued.

Method		*Time in μsec*
P4C4	P4 cycle	84
	C4 cycle	97
RK4	Cycle 1	35
	Cycle 2	35
	Cycle 3	35
	Cycle 4	49

The following table is an approximation to the accuracy band-width performance of the DES-1 with various integration schemes. It assumes that the step size is 10 milliseconds and that the step error is 10^{-4}:

Order of Integration	*Maximum Resonant Frequency*
2	8.5 rad ~ 1.35 cps
3	22 rad ~ 3.5 cps
4	41 rad ~ 6.53 cps

2.6 *Sine Operator*

The sine operator is specified as follows:

SIN, θ, Y.

Contents of the memory location specified by the sine of θ is computed and stored in the memory location specified by the contents of Y. The operator input (θ) is in revolutions, $-\frac{1}{2} \leq (\theta) < \frac{1}{2}$. Second and third quadrant angles are reduced to the equivalent first or fourth quadrant angle φ. The sine is computed from the power series,

$$\sin 2\pi\varphi = C_1\varphi + C_3\varphi^3 + C_5\varphi^5 + C_7\varphi^7 + C_9\varphi^9$$

where $\frac{1}{2} \leq \varphi < \frac{1}{2}$

and $C_1 = 1.5707963$

$C_3 = 0.6459637$

$C_5 = 0.0796897$

$C_7 = 0.0046738$

$C_9 = 0.0001515$

with a maximum error of approximately three parts in 10^7. Execution time for the entire process is 80.5 microseconds.

2.7 *Cosine Operator*

The cosine operator is specified as follows:

COS, θ, X.

The cosine of the contents of the memory location specified by θ is computed and stored in the memory location specified by the contents of X. The operator input is in revolutions, $-\frac{1}{2} \leq (\theta) < \frac{1}{2}$. The cosine of (θ) is obtained by computing $\sin 2\pi((\theta) + \frac{1}{4})$ as described under sine operator. The execution time for the cosine operator is 82.25 microseconds.

2.8 *Arctangent Operator*

The arctangent operator is specified as follows:

ATAN, Y, X, θ.

The arctangent of (y)/(x) is computed and stored in the location specified by the contents of θ. The operator output is in revolutions, $-\frac{1}{2} \leq (\theta) < \frac{1}{2}$. The principal angle φ is computed from the power series,

$$\varphi = C_1U + C_3U^3 + C_5U^5 + C_7U^7 + C_9U^9 + C_{11}U^{11} + C_{13}U^{13}$$

where $U = x/y$ if $|x| \geq |y|$

$U = x/y$ if $|x| < |y|$

and $C_1 = 0.15914533$

$C_3 = 0.05302625$

$C_5 = 0.03152520$

$C_7 = 0.02106178$

$C_9 = 0.01267292$

$C_{11} = 0.00534860$

$C_{13} = 0.00108423$

The output angle (θ) is obtained from the principal angle φ by quadrant correction. All four quadrants are resolved and the error in θ is less than 3 parts in 10^7. Execution time for the arctangent operator is 171.5 microseconds.

2.9 *Square Root Operator*

The square root operator is specified as follows:

SQRT, X, Y.

The square root of the contents of the memory location specified by x is computed and stored in the location specified by the contents of Y. The square root of (x) is obtained as follows: The input (x) is normalized such that $x_N = 2^N(x)$ lies within the range $\frac{1}{2}$ to 1. This range is divided into four equally spaced intervals and within each interval $\sqrt{x_N}$ is representable by a third degree power series in x_N. The square root of x_N is obtained by evaluation of one of the four power series and the square root of (x) is obtained from

$$\sqrt{x} = 2^{N/2}\sqrt{x_N}.$$

Execution time for the square root operator is 57.75 microseconds and the maximum error is three parts in 10^7.

2.10. *Function of One Variable Operator*

The function of one variable operator is specified as follows:

FUN 1, X, Y.

The contents of the memory location specified by X is the input. The function of one variable is computed and the result stored in the memory location specified by Y. The method of straight line approximations is used to construct the function. Fifteen values of the independent variable and dependent variable are stored relative to the location of the operator output.

x_1	y_1
x_2	y_2
x_3	y_3
.	.
.	.
.	.
x_{15}	y_{15}

The function y is computed from the relation

$$y = y_i + \frac{y_{i+1} - y_i}{x_{i+1} - x_i}(x - x_i)$$

where $x_i < x \leq x_{i+1}$

The independent variable points can be arbitrarily spaced. During computation, the interval within which x lies is determined by the interval used during the previous integration cycle by checking this interval plus intervals on either side. If x does not lie in any of these three intervals, all intervals are scanned linearly. In practice, the variable x should never change so rapidly that the linear scan is required. The linear scan is used primarily when the DES-1 is put into the RESET mode. During computation, the function of one variable operator requires 77 microseconds for execution. A slightly longer period is needed if the linear scan is necessary. The accuracy in this method of function generation is dependent upon how well the function can be approximated with straight line segments.

2.11 *Function of Two Variables Operator*

The function of two variables operator is specified as follows:

FUN 2, X, Y, Z.

The contents of memory locations specified by X and Y are the inputs. The function of two variables is computed and the result stored in the memory location specified by Z. The method of straight line approximations is used to construct the function. Values of dependent and independent variables are stored relative to the location specified by Z indicated, as follows:

	y_1	y_2	.	.	.	y_7
x_1	z_{11}	z_{12}	.	.	.	z_{17}
x_2	z_{21}	z_{22}	.	.	.	z_{27}
.	.	.	.	.	.	.
.	.	.				.
.	.	.				.
x_7	z_{71}	x_{72}	.	.	.	z_{77}

The function z is computed from the relation,

$$z = z_{ij} + \frac{z_{i+1,j} - z_{ij}\ (x - x_i)}{x_{i+1} - x_i} + \frac{z_{i,j+1} - z_{ij}\ (y - y_i)}{y_{i+1} - y_j}$$
$$+ \frac{(z_{i+1,j+1} - z_{i+1,j}) - (z_{i,j+1} - z_{i,i})}{(x_{i+1} - x_i)\ (y_{j+1} - y_i)}\ (x - x_i)\,(y - y_j)$$

where $x_i < x \leq x_{i+1}$

$y_i < y \leq y_{j+1}$

The above relation is exactly satisfied on all four corners of the square

x_i, y_j

x_{i+1}, y_j

x_i, y_{j+1}

x_{i+1}, y_{j+1}

The independent variable points can be arbitrarily spaced so long as a square mesh is maintained. The region within which x and y lie, is determined by the region used during the previous integration cycle by an extension of the method used for a function of one variable. During computation, the function of two variables operator requires 231 microseconds for execution.

2.12 *Comparator Operator*

The comparator operator is specified as follows:

COMP X, Y, X_1, Y_1, Z_1, X_2, Y_2, Z_2, ''' X_k, Y_k, Z_k

The comparator operator simulates a K pole double-throw switch. The comparator operator functions as follows:

if $x \leq y$

$z_1 = x_1$

$z_2 = x_2$

.

.

.

$z_k = x_k$

if $x < y$

$z_1 = y_1$

$z_2 = y_2$

.

.

.

$z_k = y_k$

where x represents the contents of the memory location specified by X

y represents the contents of the memory location specified by Y

x_1 represents the contents of the memory location specified by X_1

etc.

Execution time for the comparator operator is 12.25 + 7k microseconds where k is the number of switched inputs.

2.13 *Other Useful Operators*

The operators previously described are considered to be most universal in application. However, there are many others which are useful on occasion. For a particular application, a requirement may exist for Bessel Functions, or Legendre Polynominals, or exponentials or logarithms.

In addition to operators which are useful for computation, it is necessary to have operators which provide a convenient means of input/output. A number of these are provided for analog input/output, for typewriter output, etc.

3. *Programming Example*

An example will illustrate the programming of the DES-1. This example is complex in order to illustrate, first of all, the method in which the analog language of the DES-1 is used to program the machine and, secondly, to illustrate the homogeneity of the analog language with the standard DES-1 general purpose language.

The Pulsed Jet Control System problem is an example of this type of problem because a portion of the system (the rigid body dynamics) is continuous, and a portion of the system (the pulsed jet control) is digital in nature. Differential equations expressing the angular orientation of the vehicle are handled by operators. Jet commands and jet characteristics are digital in nature and are handled by the general purpose computer instructions.

The equations to be mechanized for solving this problem are given in Table 4. Problem mechanization is shown in Figures 1 and 2. A listing of operator connections is given in Table 5. It should be noted that Table 5 consists of a mixture of operators and general purpose computer instructions. The general purpose instructions are used to mechanize the Sampler indicated in Figure 2. Operators and

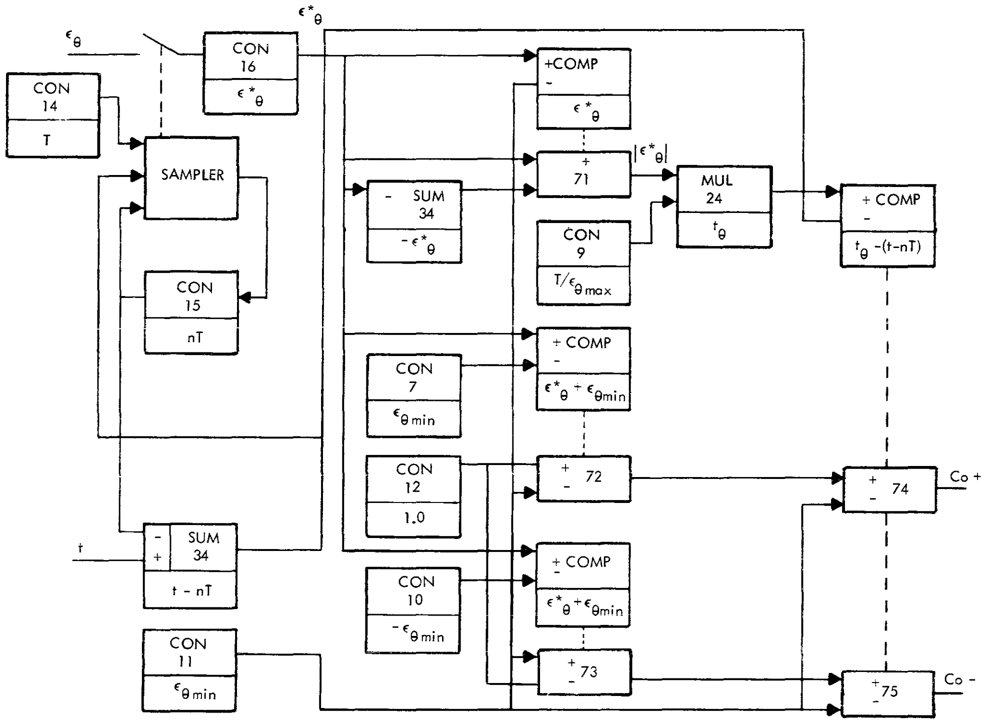

ϵ_θ
ϵ^*_θ
CON 14 T
CON 16 ϵ^*_θ
SAMPLER
CON 15 nT
SUM 34 t - nT
t
CON 11 $\epsilon_{\theta min}$
SUM 34 $-\epsilon^*_\theta$
CON 7 $\epsilon_{\theta min}$
CON 12 1.0
CON 10 $-\epsilon_{\theta min}$
+COMP ϵ^*_θ
71
$|\epsilon^*_\theta|$
CON 9 $T/\epsilon_{\theta max}$
+ COMP $\epsilon^*_\theta + \epsilon_{\theta min}$
72
± COMP $\epsilon^*_\theta + \epsilon_{\theta min}$
73
MUL 24 t_θ
+ COMP t_θ -(t-nT)
74
Co +
75
Co -

Figure 2. Jet Control.

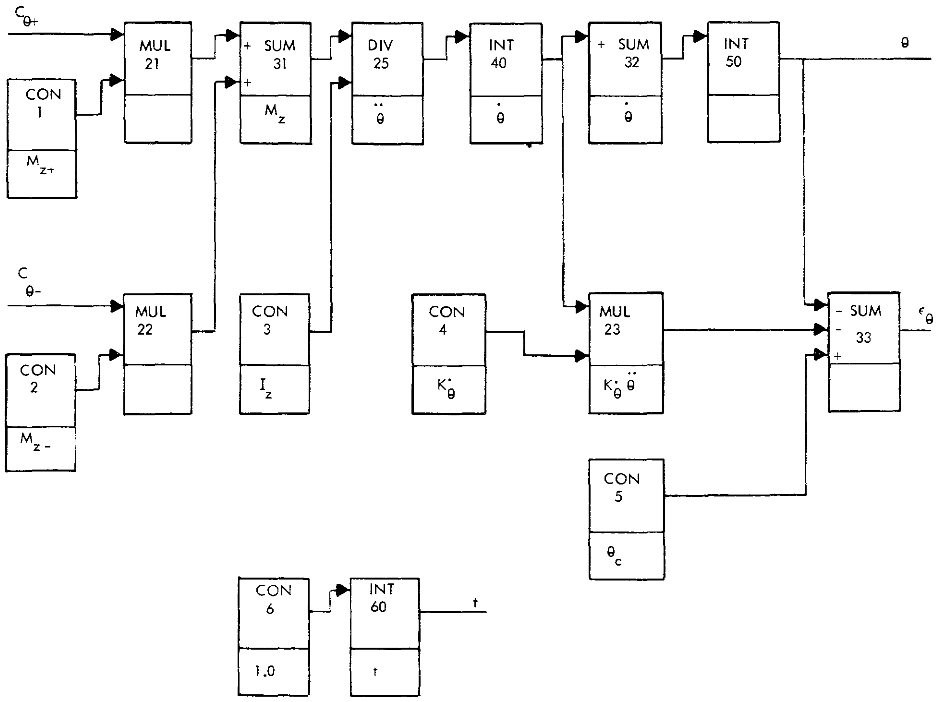


Figure 1. Angular Dynamics.

general purpose instructions can be mixed in any fashion in the DES-1, thereby retaining decision and logical capabilities of general purpose computers.

Table 4. Equations

$1_z\ddot{\theta} = M_z$	Pitch angular motion
$\epsilon_\theta = \theta_c - \theta - K_{\dot\theta}\dot{\theta}$	Pitch error
$\epsilon^*_\theta = \epsilon_\theta$ at $t = nt$	Sample pitch error
$t_\theta = \dfrac{\lvert\epsilon^*_\theta\rvert}{\epsilon_{\theta\,\max}} T$	Jet pulse duration
$C_{\theta+} = \begin{cases} 1 \text{ if } \epsilon_\theta > \epsilon_{\theta\,\min} \text{ and } t - nt < t_\theta \\ 0 \text{ if } \epsilon_\theta \leq \epsilon_{\theta\,\min} \end{cases}$ $C_{\theta-} = \begin{cases} 1 \text{ if } \epsilon_\theta \leq -\epsilon_{\theta\,\min} \text{ and } t - nt < t_\theta \\ 0 \text{ if } \epsilon_\theta > -\epsilon_{\theta\,\min} \end{cases}$	Moment commands
$M_z = M_{z+}C_{\theta+} - M_{z-}C_{\theta-}$	Control moment

The meaning of Table 5 should be emphasized. The initial instructions are in the analog language and are analogous to a list prepared for plugboard programming in an analog machine. Starting with instruction "LDA", the instructions are in digital language since the operations to be performed are appropriate to a digital computer rather than analog elements. The analog language starts again with the instruction "SUM". Thus, this example illustrates first that the computer is programmed like an analog machine and, secondly, the homogeneity of the analog and digital languages. Of course, in addition to the program interconnections, initial condition data would also have to be entered into the computer. This is analogous to setting up Potentiometers or feeding tapes into an automatic analog initial condition set-up system.

Table 5. Operator Connections

SUM,	+40,	32.	
MUL,	40,	4,	23.
SUM,	+ 5,	—50,	—23, 33.
SUM,	+60,	—15,	34.
LDA	34		Load accumulator with t — nt
SKG	14		skip if t — nt > T
TRU			transfer (do not sample)
LDA	14		load accumulator with T
ADM	15		add to memory nT + T → nT
LDA	33		load accumulator with $\epsilon\theta$
			store $\epsilon\theta$ in
STA	16		$\epsilon^*\theta$
SUM,	—16,	34.	
COMP,	16,	11,	16, 34, 71.
COMP,	16,	7,	12, 11, 72.
COMP,	16,	10,	11, 12, 72.
MUL,	71,	9,	24.
COMP,	24,	34,	72, 11, 74, 73, 11, 75.
MUL,	1,	74,	21.
MUL,	2,	75,	22.
SUM,	21,	22,	31.
DIV,	31,	31,	25.
INT,	25,	40.	
INT,	32,	60.	
INT,	6,	60.	

4. *The Control and Display Console*

The two elements of the DES-1 thus far described—the computer proper, and the operators are tied together and operated from the control and display console. This console aids the operator in solving and debugging problems in conjunction with the SDS 9300 Computer console and uses the typewriter, photo-reader, and paper tape punch belonging to that console. A drawing of the DES-1 console is shown in Figure 3.

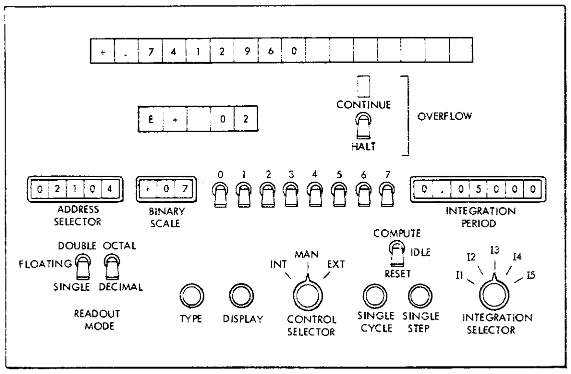


Figure 3. DES–1 Control Console.

Mode Selector

Three operation modes are provided by the mode selector: COMPUTE, IDLE, and RESET. The problem solution is generated in the COMPUTE mode. IDLE mode is provided for stopping the problem solution at any time to inspect various parts of the solution. RESET mode is used to automatically insert initial conditions for re-run of the problem. In addition, single cycle and single iteration control are provided to facilitate problem checkout.

Control Selector

Three sources of control are provided by the CONTROL SELECTOR: internal, manual and external. INT refers to program control of the operation modes. MAN gives problem control to the mode selector. EXT gives problem control to an external device allowing the DES-1 to be slaved to the external device.

Integration Selector

A means of selecting one of five types of integration is provided by INTEGRATION SELECTOR. These integration types are normally:

1) Predictor—2nd order
2) Predictor—3rd order
3) Predictor—4th order
4) Predictor-corrector—4th order
5) Runge Kutta—4th order

Integration Period Selector

A means of synchronizing machine iterations to real time is provided by the INTEGRATION PERIOD selector. The minimum integration is dependent upon problem size, but can be extended to any convenient period up to 10 seconds in increments of 10 microseconds. An on-off switch is provided for switching out the integration period selector in problems requiring no correlation to real time.

Readout

The output display and ADDRESS SELECTOR provide a convenient means of monitoring operators and intermediate results, especially during program debugging. An operator output or memory location is selected by the ADDRESS SELECTOR. A binary exponent is selected by the BINARY SCALE to allow the information readout to be presented in a properly scaled form. The DISPLAY pushbutton causes the contents of the selected address to be scaled by the binary exponent, converted to decimal, and presented on the output display in the form of a signed fraction and decimal exponent. The information presented on the display can be typed out on the typewriter by operation of the TYPE pushbutton. A READOUT MODE control allows all combinations of DECIMAL or OCTAL and SINGLE or DOUBLE fixed point or FLOATING point number readout.

Sense Switches

A group of eight sense switches located on the DES-1 console are for program-determined use.

Arithmetic Overflow

Arithmetic overflow is indicated by the OVERFLOW light. The DES-1 can be set to HALT on overflow or CONTINUE on overflow.

Program Input/Output

The photo-reader or typewriter is used for program input. Normally, the photo-reader is used to load the program and changes or corrections are made from the typewriter. The corrected program can be punched and/or typed out for future use.

Data Input/Output

Normally the photo-reader is used to load initial conditions, constants and other data into the computer. Changes or new values can be loaded for either the photo-reader or typewriter.

Digital Potentiometer

Continuously variable 16-turn digital potentiometers can be supplied to permit continuous variation of a problem variable during actual solution.

5. *Input/Output*

A typical input/output configuration which can be used to integrate the DES-1 into a system is shown in Figure 4. Five types of input/output are used:

A. *Analog Input*

This type of input accepts analog signals, converts them to digital form and enters the results into the DES-1. In this example, a multiplexer provides for 96 inputs; these may be selected either by channel number or the channels can be sampled sequentially. The former case requires additional computer time, since in the latter case the sequencing can be performed independently of the computer. Note that both SET and ADVANCE controls are provided.

A sample-and-hold amplifier is provided for signals with higher frequency contents. In some systems the time lag caused by the sequen-

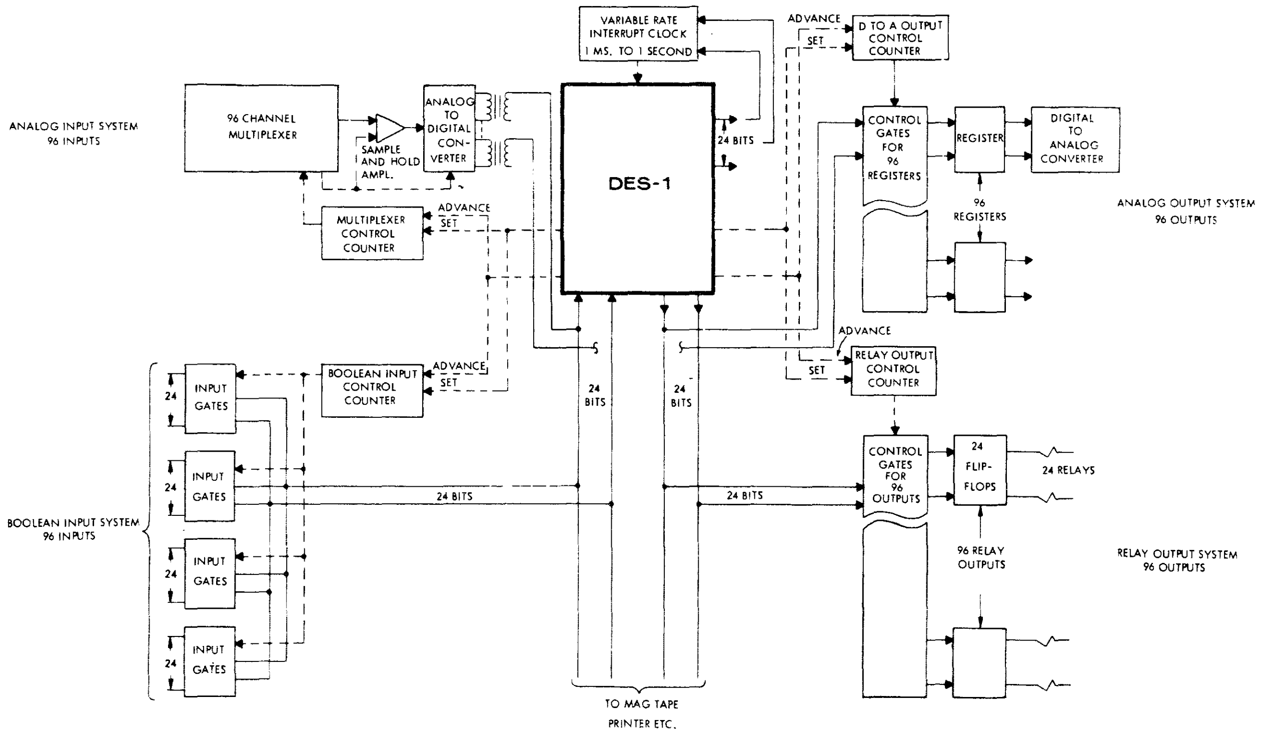

VARIABLE RATE INTERRUPT CLOCK I MS. TO 1 SECOND
ADVANCE
SET
D TO A OUTPUT CONTROL COUNTER
ANALOG INPUT SYSTEM 96 INPUTS
96 CHANNEL MULTIPLEXER
SAMPLE AND HOLD AMPL.
ANALOG TO DIGITAL CONVERTER
DES-1
24 BITS
CONTROL GATES FOR 96 REGISTERS
REGISTER
DIGITAL TO ANALOG CONVERTER
96 REGISTERS
ANALOG OUTPUT SYSTEM 96 OUTPUTS
MULTIPLEXER CONTROL COUNTER
ADVANCE
SET
RELAY OUTPUT CONTROL COUNTER
BOOLEAN INPUT CONTROL COUNTER
INPUT GATES
24
24 BITS
BOOLEAN INPUT SYSTEM 96 INPUTS
CONTROL GATES FOR 96 OUTPUTS
24 FLIP-FLOPS
24 RELAYS
96 RELAY OUTPUTS
RELAY OUTPUT SYSTEM 96 OUTPUTS
TO MAG TAPE PRINTER ETC.

Figure 4. Typical Input/Output Configuration.

tial nature of the conversion process may generate intolerable errors. A simultaneous sample-and-hold, which samples all channels simultaneously, can be used in these cases. Note that the analog and digital grounds are isolated by transformer coupling.

The total computer time required to process 96 channels into the computer is less than 180 microseconds.

B. *Boolean Input*

This type of input accepts on-off signals and enters them into the computer. These signals may be generated by various types of switch closures or by combinations of lines which represent other control information. Again, individual groups of 24 can be sampled or the sampling can be sequential. The total time required for 96 inputs is less than 10 microseconds.

C. *Analog Output*

This type of output accepts digital information from the computer and converts it to analog form. A total of 96 registers are provided to drive converters. Some of the registers are shown without converters to indicate that special devices, such as digital servos, can be employed. Alternately, a single register can be used sequentially with a single digital-to-analog and a number of holding circuits to provide slower, less accurate but more economical conversion. Again, SET and ADVANCE lines are provided and transformer coupling isolates the grounds.

The total time required to process 96 channels is less than 180 microseconds.

D. *Boolean Output*

This type of output generates on-off signals for driving relays, indicators, etc. Individual groups of 24 can be generated or sampling can be sequential. The total time required for 96 inputs is less than 10 microseconds.

E. *Clock*

The solution of differential equations by numerical methods proceeds by steps or cycles. A clock is provided to signal the start of each cycle. The cycle time is set into the clock by the computer to the nearest microsecond. The clock accuracy exceeds one part in 10^5. At the appropriate times, the clock activates an interrupt which sends the program to the start of the next cycle.

The total time for the entire input/output operation as shown in Figure 4 is less than 400 microseconds. If solutions are generated 25 times per second, this represents less than 1% of the available computing time.

Since the DES-1 is oriented towards classical simulation problems, and since in many simulations actual hardware must be tied into the computing system, the input/output system is a critical area.

6. *Summary*

This brief talk has attempted to indicate how the DES-1 can solve some of the classical problems of simulation, digitally, and to the greater accuracies that are required by today's technology. Further, the DES-1 is, in many cases, less expensive than analog equipment. Several other points should be made that are less technical but perhaps as important. Being a digital system, the maintenance problem of the DES-1 is considerably less severe than that of an analog machine. Further, set-up time is relatively insignificant since all information is stored on paper tape or punched cards which can be read rapidly and accurately. No initial "tuning" is required. It is certainly the case that in some problems the frequency response of the DES-1 is insufficient; however, the DES-1 represents the first step in what may be a change in the technology which will replace analog computers to a large measure by digital technique.

REFERENCES

1. Palevsky, Max: The Digital Differential Analyzer (pp. 19-14 to 19-72), Computer Handbook, McGraw-Hill Book Company (1962).
2. Dunn, W. H., C. Eldert, and P. V. Levonian: A Digital Computer for Use in an Operational Flight Trainer, IRE

Trans. on Electronic Computers, EC–4(2): 55–63 (June, 1955).

3. GRAY, H. J., Jr.: Numerical Methods in Digital Real-time Simulation, Quart. Appl. Math., 12(2): 133–140 (July, 1954).

4. CONNELLY, MARK E.: Real-Time Analog-Digital Computation (pp. R-29 to R-39), Simulation, Fall 1963, Simulation Councils, Inc., La Jolla, California.

SYSTEMS IMPLICATIONS OF NEW MEMORY DEVELOPMENTS*

Memoria ex Machina

Dr. Sullivan G. Campbell
Xerox Corporation

Computer design has historically been limited by, and even largely determined by, available memory capability: speed, size, cost, and operational characteristics. Just as Edgar Allen Poe could create the entire complicated plot and counterplot of Dickens's "Barnaby Rudge" after reading the first installment, so can a really competent systems designer describe a well-designed computer system with considerable precision if he only knows what memories were available to the designers. The classical problems, such as serial vs. parallel, synchronous vs. asynchronous, even decimal vs. binary, all relate to ways of getting around problems with memory.

1. *Memories and storage devices.* In the United States, it has become traditional to identify a certain section (usually a particular box) of a digital computer as "the memory"; in England, the same physical hardware would be called "the Store", after Charles Babbage, but with perhaps a touch of distaste for the anthropomorphism. If Webster[†] is to be followed (and he is the only reference which will be used here), the British term "store" does in fact describe the things called "memories" in most exisiting computers, while "memory" describes better much of the subject of this paper. Websters definitions are:

Store:	something that is stored or kept for future use; articles . . . accumulated for some specific object and drawn upon as needed.
Memory:	the power or process of reproducing or recalling what has been learned and retained especially through associative mechanisms; persistent modification of structure or behavior resulting from an organism's activity or experience.

In conformance with both Webster and Anglo-Saxon tradition, we shall use the term "storage" for devices in which information is (hopefully) unaltered by the process of being stored, retained, and accessed; and in which each physical unit of information (bit, character, word, record, etc.) has at least one (explicit or implicit) address which applies uniquely to it. The term "memory" will then be used for all types of information storage devices, assuming only that the purpose of such devices is the storage, retention, and access of information; e.g., we include tag addressed memories, associative memories, memories which do not remember for very long, memories which rearrange the bits stored in them, or memories which only remember what they are told to remember.

2. *Kind of memories:* speed and size; what they remember. It has become traditional to

[†] Webster's New Collegiate Dictionary, G. & C. Merriam Co., Publishers.

[*] Based on work done at International Business Machines Corporation, Poughkeepsie, New York.

evaluate computer storage in terms of various speed and size characteristics. The usual size characteristics include number of words, characters and bits. Obviously the number of bits will always define size precisely, but number of characters or words is more meaningful to users of character-oriented or word-oriented machines (and also allows for endless confusion). Speed is usually expressed in terms of the total cycle time required between successive accesses of the same memory box; for most memories, the various elements which make up the cycle time (read cycle, write cycle; selection, driving, sensing, amplifying, restoring) are of interest only to those particular engineers who design and build memories.

It would seem, then, that from the standpoint of the system designer, a statement of capacity (number of bits) and speed (total cycle time) would characterize memory satisfactorily; certainly these are the parameters which have been mostly used to characterize memories in the past. This has had the beneficial effect of allowing mass-production economics to be exploited fully in the memory field, since it allows widely divergent systems to use the same kind of memory array, often even the same physical box, so long as speed and size requirements are roughly the same; but at the same time these very economic factors have discouraged development of any competition to the conventional storage devices, have encouraged systems designers to draw memory as a separate box rather than as an integral part of the system, and have discouraged application of the kind of systems thinking in the memory area which has worked efficiently elsewhere.

For our purposes here, memories will be classified as conventional, unconventional, and oddball. This classification is determined by usage: by conventional memory, we mean the usual storage devices such as registers, ferrite arrays, tapes, disks, drums, punched cards, etc., used on most computing systems, or devices technically equivalent to them; by unconventional memories, we mean devices which still store, retain, and access information, such as tag-addressed and associative memories, self-sorting memories, etc., but which differ functionally from the conventional types. We reserve the term oddball for memories which do not altogether perform the three functions of storing, retaining, and accessing information, and will use the terms read-only memory (ROM), forgetful memory, and write-only memory (WOM) for memories which fail, at least somewhat, respectively in the ability to write, retain, and access information.

Since there are really memories in each of these categories, and since we really do evaluate memories in terms of speed, capacity, and cost, we are on moderately respectable ground up to this point. When we attempt to classify memories by other means, as qualitatively, or by what they remember, then the ground is no longer respectable; we run the risk of frightening small children and designers of well-ordered, well-regulated, well-constructed, and well-priced memories. We might indeed be able to classify human beings by what they remember; we have no interest whatsoever in a human being who remembers precisely everything he is told, except perhaps as a curiosity. When human beings are exposed to a great deal of information, they often do not remember any single item of it; but they do remember certain of its properties, and the most intelligent ones find and remember properties which are much more important than the physical information itself. Human beings also remember such things as how much information there was, how accurate it was, what it described, where it came from, what can be done with it, and even what it means. This is precisely what computers do not remember.

What is really the most important information that any human has? It is certainly not a collection of numbers or even a collection of formulas; it is probably not even the skills, techniques, and methodologies which have been learned and applied; but rather, it is a vague and probably ambiguous and inaccurate collection of memories from total experience, on the basis of which decisions are made and to the existence of which the individual owes most of his value.

Similarly, in many computer systems, the value of the answers produced is really much less than would be the knowledge of what went on in the system in the process of producing those answers. Almost inevitably, a computer system "sees", but has no way of interpreting, understanding, or remembering, relationships of the utmost importance, whether it is doing a hydrodynamic calculation or analyzing financial data. The information contained in the programs executed by a computing system may be the only valid operational characterization of a major activity, and of itself much more valuable than the numbers produced by those programs —just as your knowledge and experience may be of much greater value than all the things you have ever done.

It seems not too much to suggest that computer memories may some day be classified in terms of what sort of things they remember. Already control systems keep elaborate logs of gross activities, motivated usually by the noblest motive of them all: the necessity of charging each user the right amount of money. In some installations, the idea of keeping a permanent file of all information ever generated by the computer, even of being able to recover its total past history from any point in time, is being developed, and such development may well be essential to our eventually learning what part of this information is important, which is to say, how the computer can learn to profit from its own experience, or rather, how we can learn to profit more from the computer's experience.

3. *Normalized technologies and systems thinkers.* The term "normalized technology" has been used to describe a given mature electronic technology in which the usual circuit component, circuit, packaging, and memory problems have been well solved relative to that technology. Between normalized technologies, the ratios between logical circuit speeds and memory switching speeds will remain remarkably close to constant. This should surprise no one: information read into or out of a memory must be selected, driven, sensed, amplified, and (usually) restored; each of these is a substantial operation involving the same driving and switching circuitry which can perform logical operations. Improvement in driving and switching capability will therefore improve memory speeds more or less linearly; conversely, memories cannot be made faster than the associated logical componentry.

Without wishing to argue the details, we present reasonably well-chosen examples of two normalized technologies:

Circuit Delay (per decision)	Register Memory Size	Local Store	1 k Words	32 k Words	k^2 Words
20	150	300	500	2000	4000
1	10	20	100	500	4000

where times are given in miniseconds and sizes are given in words assumed to be between 36 and 72 bits ($k = 2^{10}$). (1 minisecond = 1 hemidemisemiquaver.)

These figures illustrate a well-known problem: that circuit speeds are improving faster than memory speeds, and that at some point out toward the periphery of the system, effective speeds are improving very little, if at all. With new components, with modern micro-circuit and transmission line techniques, it is possible to speed up the logical hardware much more than the bulk store; if it is possible to speed up circuits by a factor of 20, and to solve the related packaging problems, there is no discernible ability to achieve comparable speedup of bulk storage, and there is in fact very little performance improvement apparent in peripheral storage devices which are basically mechanical. For applications which are already limited by the speed of these devices (and most applications are limited by them to some extent) it is difficult to see how very much performance improvement can be achieved just by improving the speed of logical operation.

During the past five years or so, systems designers have made some valiant attempts to get around the problem that for simple, efficient designs in normalized technologies, memories are just not fast enough to keep up with the rest of the system. First they multiplexed the memories and buffered the input-output (I/O) through multiple channels; but then there arose a class of designers who were determined to sop up the excess memory speed and capacity thus generated. They drew systems with multiple processors, multiple controls, multiple arithmetic units, but most of these did not get built.

Under this general polyploid concept, the more advanced thinkers left the polystichous organization for the polysyndetic: they observed that memories are like glue, in that they can be used to stick things together, and with the concept of local storage, which could proliferate through the system like sarcoma, the polysyndetic concept became possible. At the same time that such advanced concepts were developing into very sophisticated specific system designs, such as the Polymorphinic, the opposing memnocentric school was developing the concept that memories might be more like gravity than like glue; they accordingly placed the memory in the center of the system, and let the other components circle around it in the appropriate orbital trajectories.

Despite the brilliance and ingenuity of all these approaches, the problem of what to do about memories which are too slow and too small does not appear at present to have been satisfactorily solved, at least from the standpoint of the people who use computers, as opposed to those who make them, and even those who purchase them. In part this may be because we have not concentrated enough on how to use large memories, fast memories, or special memories; and in part it may be because the system designer rarely looks inside the memory box.

4. *Some unconventional use of conventional memory.* Let us suppose that we have some conventional memories: say, a stack of core arrays; and that we have flown down from the lofty perch of systems designer and actually taken a look at what is inside the box; further, let us even assume that we have gone so far as to look at what goes on inside the computer itself in terms of the statistical distribution of stores and fetches for various applications; perhaps we have even characterized types of computation by these statistical distributions. Perhaps we were initially surprised to observe that there were four or five fetches for each store; but then we realized that execution of most instructions involves fetching the instruction and fetching an operand (and possibly also an index register).

Examining instruction fetches and data fetches and stores, assuming a memory with capacity of n words, we can measure or calculate the probabilities p_{ij} that a memory reference to word i will be followed by a reference to word j, and we can identify the two extremes:

$$\text{(a)}\quad p_{ij} = \begin{matrix} 1 \text{ if } j = i + 1 \\ 0 \text{ otherwise} \end{matrix}$$

and (b) $p_{ij} = 1/n$ for all i, j.

The first of these represents the completely orderly calculation in which instructions and/or data are accessed in strict sequence; the other extreme represents a completely random situation. Obviously the p_{ij} can refer to either the instruction stream or the data stream, and certainly all problems lie somewhere in between the completely random and the completely orderly, although some important problems lie closely enough to one or the other to permit reasonable approximation.

For case (a), the orderly case, we can use interleaving and multiplexing techniques to make the apparent memory access as fast as we choose; in fact, we can do this whenever the matrix A(ij), $a_{ij} = p_{ij}$, consists of all 0's and 1's, but we cannot use such techniques at all efficiently when all the elements of A(ij) are equal; and the "flatter" the matrix becomes, the less use we can make of such techniques. For the orderly case, we can also use simpler techniques than interleaving; e.g., changing the addressing structure so that the consecutive addresses are not "distrub-sensitive" relative to one another, permitting the array to be run much faster and also permitting access of consecutive words to be highly overlapped. For both cases, it is clear that a well-balanced memory should have a read capability about five times that of its write cycle. It might be possible to mix orderly computations with disorderly ones in a multiprocessing sense with little memory interference; or even to use orderly memories in unconventional ways to solve disorderly problems.

For example, in many Monte Carlo calculations, which are frequently quite disorderly, we wish merely to catalog the frequency of occurrence of various physical events, whch may be accomplished with a great deal of conditional branching and counting. But it may also be accomplished by built-in memory counting facility, which permits the appropriate counters to be ticked up or down whenever the associated event occurs.

It is clear that it is not so much the problem which is "orderly" or "disorderly" as the program, which relates in turn to the programmer, the programmming system, and the instruction set. Programs may be made as disorderly as we desire, or more orderly. Instead of conditionally branching through a Monte Carlo program, the program may systematically "sort" through it, level by level; branching operations may be replaced by table lookup operations; but the attempt to apply such techniques may run afoul of the traditional relationships of the computer to its memory.

5. *Some conventional uses of unconventional memory.* Since table lookup techniques are perfectly general, and more important, since they really represent efficient means of attacking important problems, it would appear that much more attention should be paid to provision of adequate table lookup and table construction facilities both in programming systems and in the physical computing system. Such facility does not necessarily involve development of new memory components or arrays, but it does involve improving the accessing (or accessibility) of whatever elements are available. To the greatest possible extent we prefer to process addresses rather than data. The way in which information is stored is not usually the way in which it is accessed; in fact, if one is systematic, the other usually is not, as in the simple example of generating a table, then looking up arbitrary information in it; or of generating a table of the frequency of occurrence of various events, then systematically analyzing it. In either case, we may need a particular memory access mechanism for storage, and a completely different mechanism for fetches from the same physical array.

For many applications, incidence matrices can be used to provide very efficient control (e.g., routine operations on sparse matrices) but it usually turns out that the incidence arrays need to be read in by rows and out by columns, or vice-versa. In this case, we still can use the same physical array, but we need again a special access mechanism. Or, again, we may be able to calculate $y = f(x)$ when we really want $x = f^{-1}(y)$; accessing x by the mechanism used to store y is at best cumbersome.

It seems, indeed, that the most unconventional memories are likely to be designed for the most conventional applications, and particularly for the most conventional applications of commercial data processing, such as sorting, merging, updating, etc. In part this is because such problems are sufficiently well understood that more efficient memory structures for dealing with them can be defined; in part it is also because he who would sell the world a new kind of memory must be able to demonstrate its performance upon some well-understood and important application.

Associative memories, for example, have already found application inside computer control. For associating individual or small amounts of information, they have obvious application; but if the object is to associate everything with everything, then a conventional sort may be more efficient. Push-down stores might be viewed as a new concept in memory; but they also accomplish only what programmers have been doing with conventional memories for years, nor is it clear that they can match multiple accumulators and well-designed programming systems in performance. Indeed, since unconventional memories can do nothing at all which conventional memories cannot do, the burden of proof is upon the unconventional memory to prove that it is faster or less expensive. Speed must be achieved through the way in which the memory operates rather than by component performance; and cost reduction is automatically up against the mass production economic factor referred to earlier. There remains the possibility that absolute cost, rather than relative cost, will ultimately permit unconventional memories to be sold on the basis of convenience and simplicity to the user.

6. *Oddballs.* Whether a memory is read-only (ROM) or write-only (WOM) depends upon the viewpoint: from the point of view of the computer, key punched cards are a read-only memory but hard printed copy is a WOM; to the human being, each is just the opposite, leading to the suggestion that ROMs and WOMs are often involved in the man/machine interface. What is WOM and what is ROM also depends upon the state of the technology. From the viewpoint of the computer, an effective optical character reader could convert a WOM into a ROM.

Since a ROM is useless unless it has been written into at least once, we really use the term for fast-read, slow-write memories. The concept has been with us for a long time, and with a few notable exceptions, it has remained just that—a concept—despite the important and obvious uses of ROM storage. A few of the more obvious uses include:

- Stored Translators
  - Compilers
  - Simulation Languages
  - Macro Generators
  - Assembly Programs
- Stored Utilities
  - Loaders
  - Conversions
  - Editing Programs
- Stored Control
  - Machine Control Programs
  - Operational Control Programs
  - Multiprogramming
  - Multiprocessing
  - IOCS
- Stored Logic
  - Pseudo Machine Instructions
  - Simulators
  - Simulations
- Stored Programs
  - File Operators
  - Problem Programs
  - Special Functions
- Stored Tables
  - Lists
  - Function Tables
  - Branch Tables
  - Probability Distributions
  - System Descriptors

ROM's have not been used extensively for one reason: they have not been able to compete economically against conventional storage. Clearly a ROM has no real advantage over a read-write memory of the same size, speed, and cost; in order to compete, it must have either substantially higher speed, or lower cost per bit. Because of the lack of disturb sensitivity and the need only for a read cycle, it is obvious that something can be done to increase speed. The saving from eliminating only a small amount of associated circuitry is apt to be minor. But most existing systems cannot gain much performance simply by having faster memories hung on, and it appears doubtful that ROMs will ever amount to much until entire systems are designed around them, allowing them to be used with maximum efficiency, and lowering production costs. But it has already been observed that conventional memory can also be read faster by various techniques, so that the ultimate advantage of ROM may in fact be its lack of forgetfulness.

All memories are to some extent "forgetful" memories; what we are referring to here is memories which "forget" either as a function of time or a function of use. It has been often observed that a reproduction process which would allow the message to disappear at a certain time might be highly desirable; that many documents are of value only for a specified period of time; that most documents in most hands are useful only for a limited period. If the information on such documents were simply to disappear after an appropriate time, then people would not keep so much junk in their files. There would be nothing to do with a piece of blank paper but throw it away or use it again.

In a computing system we can afford to use more selective criteria. In most cases, it is really information which is not used which should be destroyed. Keeping track of the age of informations, much less its usage, is a formidable and costly task, which might better be served by a device which "forgets" and returns itself to the pool of available storage if it has not been accessed for some given period of time.

As computer systems and the associated memory capabilities become larger, more complex, and more sophisticated, it is not at all clear that our requirement that every single bit of information everywhere in the system must always be perfect will continue to be realistic; in fact, it is not realistic now. It is doubtful whether there exists any large, complex information system, or even any really large major programming system, which does not contain numerous errors. Yet the human operators and users cannot really tell whether these systems contain errors or not. It seems rather doubtful that human beings could function at all if they had always to remember everything perfectly. If errors had never been made in the transmisson of genetic information, human beings would not exist. In order to achieve some of the latent capabilities of computing systems, it may be necessary to go to memory technologies which do not permit perfect reconstruction of stored information. It seems entirely likely that as computers get smarter, as they get older and more experienced, they may be allowed to become in some sense forgetful.

The earliest computer memories were ROMs (punched cards and punched tapes) and WOMs (printers and later CRT displays); some of the earlier internal memories were also of the forgetful type. The punched card, in particular, was such a useful ROM that its existence may have contributed substantially to the lack of development of other suitable types of ROM. At the present time, forgetfulness is not highly esteemed by the designers of conventional storage; yet only the ROM really lacks the capacity to forget, and it is partly the ability of the conventional memory to forget selectively upon demand that makes it so valuable. Forgetfulness is considered to be a critically important property to displays; yet for some purposes the present ones are too forgetful. If humans had perfect visual memories, this would not be a problem. What is frequently desired in problems involving real-time display is the ability to make a change and then see what happens; to compare the old with the new. Often we do not desire a permanent record except for exception situations, but we do want to retain the previous display in some form long enough to be able to compare it with the new one. The most desirable degree of forgetfulness for the display depends upon the type of problem and the human reaction times involved.

Memories which cannot forget (ROMs), memories which forget selectively upon demand (conventional storage), and memories which forget always unless they are reminded (WOM displays) are all conceptually capable of much further development in hardware, in systems usage, and in application. More efficient systems can be designed by taking advantage of their unique characteristics and such designs ultimately may be required to effect the extension of information processing into entirely new areas of application.

"A MODIFIED HOLLAND MACHINE"

W. T. Comfort
Development Laboratory, Data Systems Division
IBM Corporation, Poughkeepsie, New York

I. INTRODUCTION

A highly parallel machine is, in concept, a multiprocessor with more than one hundred semi-independent, program-controlled computing elements. One type of highly parallel machine is the parallel-network computer, wherein the computing elements are arranged in a geometrical array and direct communication among elements is restricted for each to a local neighborhood. (See Section II.) Most of the postulated network systems have a separate central control unit that directs the operation of the network. Examples of such are Unger's machine (References 17 and 18), McCormick's machine (References 5 and 13), and SOLOMON (References 1, 14, and 15).* The only published example of a network machine without central control is that proposed by Holland (References 4, 9, 10, and 16), and is the organization to be considered here.

When one is first introduced to Holland's concept, two extremely significant problems are immediately brought to mind:

1. The almost insuperable programming problems involved;
2. The potentially prodigious amounts of hardware implied.

   Because of this, the goal of the work represented here has been to study the reorganization of Holland's machine, attempting to:

1. Improve the programmability of the machine, and thereby increase its ability to solve problems;
2. Reduce significantly the amount of hardware involved, without losing the unique capabilities which provide the potential computing power.

The Holland machine has three distinctive features which combine to make it a unique organization—its parallel network of computing elements, its method of instruction sequencing, and its method of data accessing. These features are basically the same in the organization proposed here, but have been significantly modified in details, in line with the goals cited. Each will be discussed in some detail.

II. PARALLEL-NETWORK COMPUTER

A parallel-network computer is an array of modules, each of which is provided with some capability for communicating with its four nearest neighbors. Each module can be thought of as having two relatively independent sections (Figure 1). The upper section is the memory and control section, while the lower section governs the general communication functions, as will be described in Section IV. The result is, for the Holland machine, a dual-network effect, as in Figure 2, where all modules are identical.

The network will be of fixed size and rectangular in shape. At one side of the array

* A pilot model of McCormick's machine is currently under construction at the University of Illinois. SOLOMON is being designed and developed at Westinghouse in Baltimore. These are the only such machines known by the author to be under construction.

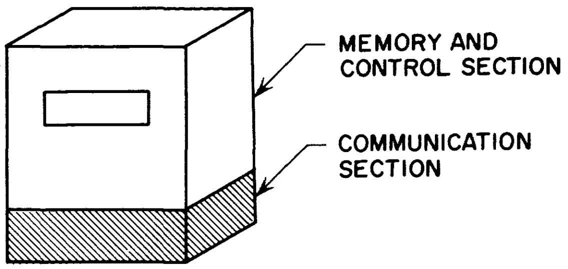


Figure 1. Module sections.

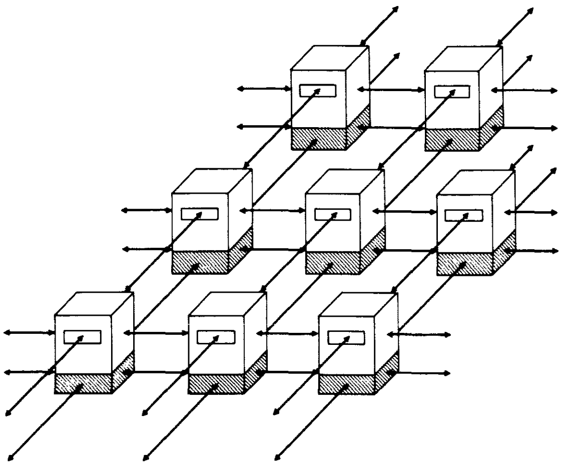


Figure 2. Holland machine network.

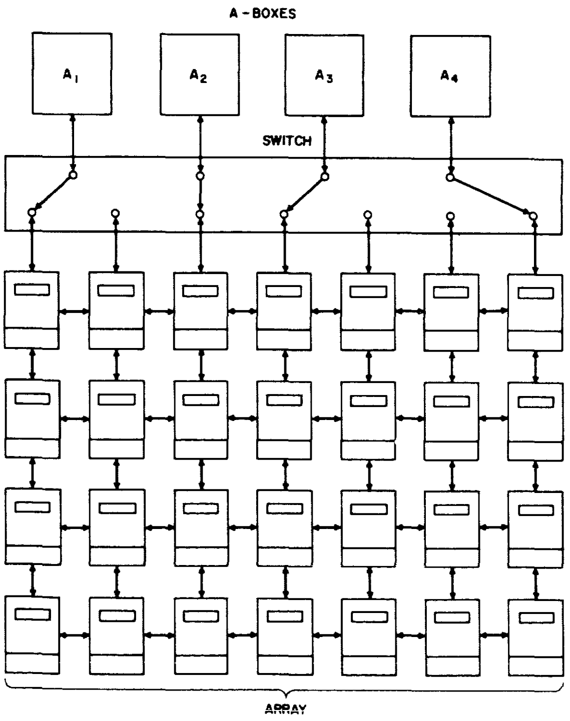


Figure 3. Network computing system.

will be a set of arithmetic units. Figure 3 shows four arithmetic units (hereafter referred to as A-boxes) connected through a program-controlled switch to a 4 $\times$ 7 array of modules.† The switch allows an A-box to be connected to the first module in any one column (no two A-boxes to the same column, no two columns to the same A-box). The array modules provide data and instruction storage, communication with and between the A-boxes, and instruction sequencing, while the A-boxes do all the actual arithmetic and logical computations. The switch is strictly a passive device. Various types of control capability are provided at the local level (as will be discussed below); however, system control exists only with individual program sequences.

III. INSTRUCTION SEQUENCING

In the Holland machine, since a module contains only one word which represents one instruction, a sequence of instructions must be correlated with a sequence of modules. The positioning of instructions is restricted such that the next instruction must be in one of the four nearest neighbor modules of the module containing the current instruction. The sequencing algorithm here is that each instruction includes a two-bit address selecting one of the four neighbors as the one with the next instruction.‡ This sequence of instruction modules is termed the "Line Of Successors," while the reverse sequence is the "Line Of Predecessors."

Since there is no central control unit, each module must execute its own instruction when its turn comes up. Thus the execution of a sequence of instructions involves the activation and deactivation of the successive modules; i.e., when a given module is turned on by its predecessor, it executes its instruction, then turns on its successor and turns itself off.

Since the execution of a (one-address) instruction normally involves an operand and the use of an accumulator, these functions must be

† It is not intended to imply that a 4 $\times$ 7 array could be considered typical in size. See Section VII.

‡ While this is a "next instruction address" of sorts, it is not specified in the instruction format. Rather, a pair of program-controlled triggers within the module is used.

provided by other modules, through some type of communication with the module containing the current instruction. This is where the A-box comes in.

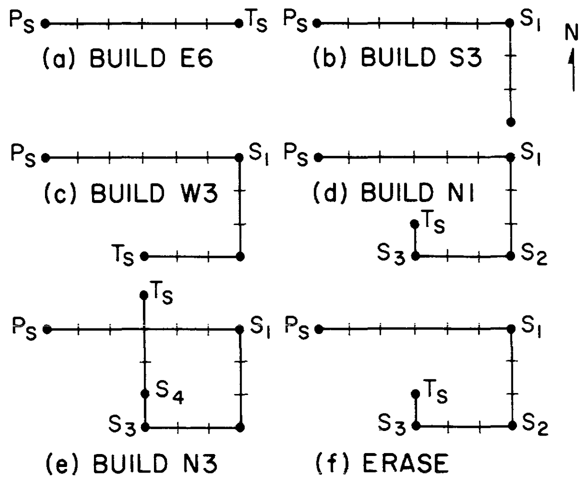
An A-box is a basic fixed-point arithmetic unit, containing an accumulator, multiplier, and operation decoder. It serves as the focal point for the execution of most instructions. An instruction sequence must start with an A-box. The line of successors (LOS) is then determined to the array by the switch setting, and through the array by the two-bit "next instruction address," as indicated previously. During instruction execution, the instruction in an active module is sent back the line of predecessors (LOP) to the A-box at the beginning of its sequence. From there an operand is fetched (if necessary), by a procedure to be described below, and the operation executed.

IV. PATH BUILDING

A path shall be defined to consist of the following: (1) A starting point, P; (2) A sequence of connected, straight-line horizontal or vertical segments; (3) A termination point, T. Such a path can be specified by: (1) the location of the starting point; and (2) the direction and length of each segment. The termination point is then defined as the end of the last segment. The length of a path segment is measured by the number of modules through which it passes. Based upon this, Holland has proposed the concept of *building a path* to the module containing the operand for a given instruction. The main difference here is in the generality of implementation.

Since paths of arbitrary length and structurè are required, simple processes have been defined whereby paths can be constructed and erased. Path building will occur one segment at a time, from some specified starting point. The process of building a path segment involves the following three steps: (1) Locate the starting point, which in this case will be the A-box; (2) If a path already exists there, move to its termination point; (3) Build the segment in the direction and to the length specified.

Figures 4a through 4e show the five steps required to build a typical path, S. Note particularly the successive definitions of the termi-

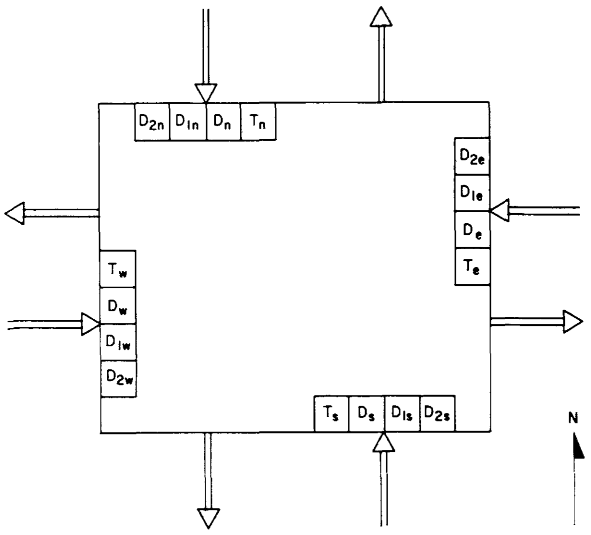


P_S T_S (a) BUILD E6
P_S S_1 (b) BUILD S3
N
P_S S_1 (c) BUILD W3 T_S
P_S S_1 (d) BUILD N1 T_S S_3 S_2
T_S P_S S_1 S_4 S_3
(e) BUILD N3
P_S S_1 T_S S_3 S_2
(f) ERASE

Figure 4. Path building.

nation point, T_S. Erasing will also occur one segment at a time. Figure 4f shows the result of one erasing step.

To implement this path-building capability, the lower section of each module (in Figure 1) is conceptually organized as shown in Figure 5. Four paths enter and leave each module. (Double lines are used in the figure to indicate that, although it has a particular direction, a path must be electrically bidirectional.) When a path enters a module, it enters a switching network which is controlled by four triggers. These triggers will have been set previously under program control, possible at program



D_{2n} D_{1n} D_n T_n
D_{2e} D_{1e} D_e T_e
T_w D_w D_{1w} D_{2w}
T_s D_s D_{1s} D_{2s}
N

Figure 5. Module communication section.

load time. The configuration of these four triggers determines the direction of flow of information coming down the path after it enters the module.

Consider the path entering the module from the west (W) side. The four triggers are labeled T_W, D_W, D_{1W}, and D_{2W}, and have the following effect:

1. If $D_W = 1$ (i.e., is set), then the two triggers D_{1W} and D_{2W} decode into the selection of one of the four directions (N, E, S, W) as the direction in which the incoming information will leave the module. By definition, $D_W = 1$ marks the first module of a segment.
2. If $D_W = 0$ and $T_W = 0$, then the incoming information passes straight through the module and out (in this case)in the E direction. All modules in a segment except the first are coded this way.
3. If $D_W = 0$ and $T_W = 1$, the path terminates at this module, and is connected to the data path in the upper memory and control section of the module.

A similar network is at the input of each of the other three paths, and all are independent of each other, and of the contents of the memory and control section of the module.

It is important to note that once the triggers are set, the selected path through the network is fixed, and will remain so until an instruction is given to change the settings of the triggers. The memory-clearing function (normally performed on a computer before a new program is loaded and executed) is also planned to reset all control triggers; specifically, all path-control networks will be set to condition 2 above. This means that a given sequence of path segments is marked only at the first module of each segment, and at the termination of the last segment.

This particular implementation has been designed to facilitate path sharing, which is necessary to get around the inherent restriction that only four paths can termniate at one module at one time, thereby limiting the number of instructions that can reference the operand at one time. To this end, the concept of a *union* is introduced. Suppose a path, A, has been built through a module, and another path, B, wishes to use it, and forms a union at that module. This union has the following characteristics:

1. Either path A or path B—or both—may fetch the operand from the termination module.
2. Either path may store a new quantity in the termination module.
3. Either path can cause a new segment to be built at the end of path A; i.e., starting at the termination module.
4. Path A can erase its last segment. This will change the union indication for path B to a termination, which is necessary to avoid leaving path B in an undetermined state.
5. If path B tries to erase, it simply disconnects from path A; i.e., it changes its union indication to a termination.

Thus the union concept allows many paths to share another.

Considering the idea of the union from a different point of view leads directly to the concept of a *channel,* wherein the path segment to be shared by other paths is only implied. Here, termination bits are loaded into the desired termination module at the time the program was loaded. Thus any path can access that operand by building a segment to that row or column of modules and forming a union in the proper direction.

While this generalized path capability with unions and channels is of itself very useful, the final goal has not been achieved. Since it is (at this point) possible to build only one segment at a time, and since in the general case it takes two segments (one horizontal and one vertical) to get from one arbitrary module to another, inevitably some extra instructions must often be inserted into a program sequence, just to provide some connecting path segments. Thus the ability to build two path segments at one time would be desirable. To provide this, in the framework of the previously discussed capabilities, an instruction will be allowed to build a path segment of arbitrary direction and length (limited only by word length) and also optionally form a union in a second arbitrary direction at the end of the segment. Thus, in combination with prespecified terminations, the double segment is achieved, without having to provide two length fields in the instruction format.

As indicated previously, paths are semi-permanently defined; i.e., once specified they are fixed until specifically changed. An alternative is provided—*temporary* path segments can be built, whereby a path segment (or double segment) can be built for the use of one instruction and automatically erased upon completion of that instruction.§

By far the most significant point to be noted here is that the path-building scheme provides the potential of accessing arbitrarily many words in the "memory" simultaneously.

V. PROGRAM EXECUTION

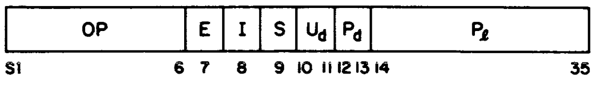
Now the concepts discussed in Sections II, III, and IV must be tied together, and the method of executing instructions and program sequences must be shown.

Figure 6 shows the instruction format, which is basically a one-address form where the "address" is the specification for a path segment. A 36-bit word is shown, although determination of a best word length has not been made. While there is not a large number of operations, a long operation-code field is desired to allow for a fair amount of direct predecoding; e.g., "horizontal microprogramming."|| Also, the path-length field must be long enough to accommodate the largest dimension of the network array. These characteristics are subject to change based upon actual design considerations.

The "address" in this instruction is not a means for directly accessing the operand. Rather, it specifies a path segment which, when built, will make the operand accessible. Thus the segment must be built before the operation can be executed. Therefore instruction execution is broken into two phases: path building and operation execution. In fact, to avoid

§ Compiled programs are expected to use temporary double segments very extensively. See Reference 4.

|| See Beckman, Brooks, and Lawless, "Developments in the Logical Organization of Computer Arithmetic and Control Units," *Proc. IRE*, Jan. 1961, p. 57.



OP	-	OPERATION CODE
E	-	ERASE
I	-	IMMEDIATE ADDRESS
S	-	UNION SELECTION
U_d	-	UNION DIRECTION
P_d	-	PATH DIRECTION
P_ℓ	-	PATH LENGTH

Figure 6. Instruction format.

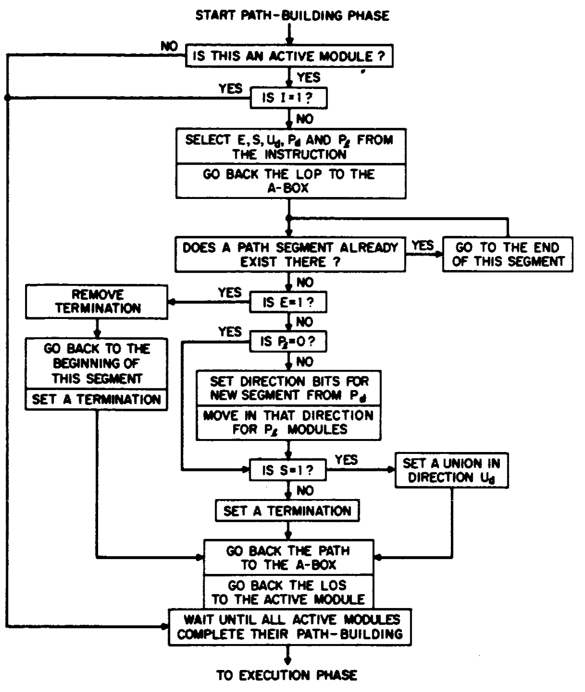
severe synchronization problems, the execution of instructions in all modules will be synchronized by phase; i.e., no module can proceed to the operation-execution phase until all active modules have completed the path-building phase, and vice versa.

The "address" portion of the instruction includes everything but the operation code. The various fields are defined as follows:

E	E = 1 means to erase the last segment of the current path, and the rest of the address is ignored. However, if a union is encountered somewhere down the path, it is simply converted to a termination.
I	I = 1 means an immediate address. The operand consists of bits 9–35 of this instruction.
S	S = 1 means that a union is to be formed after the new segment has been built.
U_d	Selects one of four directions for a union. (Ignored if S = 0.)
P_d	Selects one of four directions for the new path segment to be constructed.
P_l	Defines the length in modules of the new path segment to be built.

Note that with $P_l = 0$ and $S = 1$, only a union will be built; with $P_l = 0$ and $S = 0$, no path building occurs. Figure 7 describes the general sequence of events during path building for any active module. While some details are lacking, the figure is a reasonably comprehensive picture of the path-building process.

Once the path segment has been built, the operation can be executed. Generally speaking,


START PATH-BUILDING PHASE
IS THIS AN ACTIVE MODULE ?
NO
YES
IS I=1 ?
YES
NO
SELECT E, S, U_d, P_d AND P_ℓ FROM THE INSTRUCTION
GO BACK THE LOP TO THE A-BOX
DOES A PATH SEGMENT ALREADY EXIST THERE ?
YES
GO TO THE END OF THIS SEGMENT
NO
IS E=1 ?
YES
REMOVE TERMINATION
GO BACK TO THE BEGINNING OF THIS SEGMENT
SET A TERMINATION
NO
IS P_ℓ=0 ?
YES
NO
SET DIRECTION BITS FOR NEW SEGMENT FROM P_d
MOVE IN THAT DIRECTION FOR P_ℓ MODULES
IS S=1 ?
YES
SET A UNION IN DIRECTION U_d
NO
SET A TERMINATION
GO BACK THE PATH TO THE A-BOX
GO BACK THE LOS TO THE ACTIVE MODULE
WAIT UNTIL ALL ACTIVE MODULES COMPLETE THEIR PATH-BUILDING
TO EXECUTION PHASE

(NOTE: LOP ≡ LINE OF PREDECESSORS; LOS ≡ LINE OF SUCCESSORS)

Figure 7. Path-building phase sequencing.

the operation code is sent back to the A-box, where most of the decoding and actual execution take place. Figure 8 shows the basic process.

VI. INSTRUCTION SET

Figure 9 lists the operation codes proposed for the machine. Most of them are reasonably standard for a one-address, fixed-point machine, and will not be discussed in detail. However, some are rather peculiar to this type of organization, and deserve more careful consideration:

STP This is a little different from the normal stop instruction, because it does not stop the whole machine. Only the particular instruction sequence of which this is a part will stop. All other sequences currently active are unaffected.

TMI
TZE If the proper condition (minus or zero, respectively) is not met by the quantity in the accumulator of the A-box, the next instruction in sequence follows. However, if it is met, then the module at the path termination, *instead of* the normal successor, is activated on the next cycle.

ACT The module at the path termination, *in addition to* the normal successor, is activated on the next cycle. This is

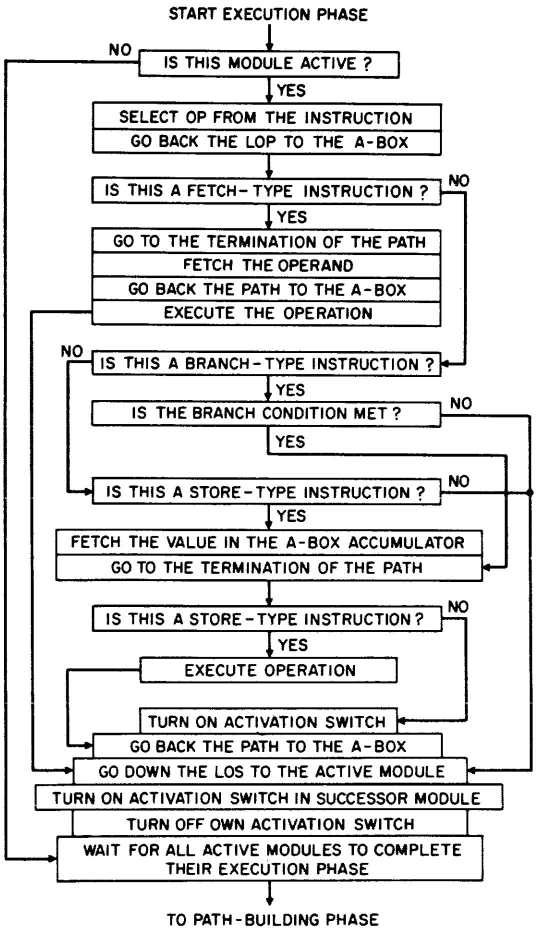

START EXECUTION PHASE
IS THIS MODULE ACTIVE ?
NO
YES
SELECT OP FROM THE INSTRUCTION
GO BACK THE LOP TO THE A-BOX
IS THIS A FETCH-TYPE INSTRUCTION ?
NO
YES
GO TO THE TERMINATION OF THE PATH
FETCH THE OPERAND
GO BACK THE PATH TO THE A-BOX
EXECUTE THE OPERATION
IS THIS A BRANCH-TYPE INSTRUCTION ?
NO
YES
IS THE BRANCH CONDITION MET ?
NO
YES
IS THIS A STORE-TYPE INSTRUCTION ?
NO
YES
FETCH THE VALUE IN THE A-BOX ACCUMULATOR
GO TO THE TERMINATION OF THE PATH
IS THIS A STORE-TYPE INSTRUCTION ?
NO
YES
EXECUTE OPERATION
TURN ON ACTIVATION SWITCH
GO BACK THE PATH TO THE A-BOX
GO DOWN THE LOS TO THE ACTIVE MODULE
TURN ON ACTIVATION SWITCH IN SUCCESSOR MODULE
TURN OFF OWN ACTIVATION SWITCH
WAIT FOR ALL ACTIVE MODULES TO COMPLETE THEIR EXECUTION PHASE
TO PATH-BUILDING PHASE

Figure 8. Execution phase sequencing.

ALS	Arithmetic Left Shift	STP	Stop
ARS	Arithmetic Right Shift	NOP	No Operation
LGL	Logical Left Shift	HLD1	Hold Level 1
LGR	Logical Right Shift	HLD2	Hold Level 2
RLS	Rotate Left Shift	HLD3	Hold Level 3
RRS	Rotate Right Shift	REL1	Release Level 1
CLA	Clear and Add	REL2	Release Level 2
ADD	Add	REL3	Release Level 3
SUB	Subtract	WAIT	Wait
MPY	Multiply	ACT	Activate
DIV	Divide	TMI	Transfer on Minus
STO	Store	TZE	Transfer on Zero
ANA	AND to Accumulator	REC	Record Triggers
ORA	Inclusive OR to Accumulator	SET	Set Triggers
ERA	Exclusive OR to Accumulator	RST	Reset Triggers
COM	Complement		

Figure 9. Instruction set.

	one way that multiple sequences can be started.
HLD REL	This is a method of global synchronization. Whenever a sequence executes a hold (HLD1, HLD2, or HLD3), that module enters a pseudo-active state, wherein it never activates its successor or builds any paths. As soon as some module in the network executes the corresponding release (REL1, REL2, or REL3), all those modules in hold status at the level are allowed to activate their successors on the next cycle. This allows a large number of widely dispersed modules to be easily and directly synchronized.
WAIT	This instruction always requires an immediate address, an integer, say N. The module will not activate its successor until it has been activated N times. This allows for local synchronization of program segments.
SET RST REC	In addition to its one word of memory, every module contains a complement of control triggers (as discussed in Sections III and IV). These instructions provide program control for them. REC is basically a CLA, except that the status of all control triggers is drawn together into a binary word and fetched to the A-box. SET (RST) causes the masked triggers in the path-termination module to be set to ones (zeros).

VII. CONCLUSION

This proposed organization is clearly a modification to Holland's machine, as described in Reference 9. However, the following results seem evident:

1. Programmability has been improved by several orders of magnitude. Admittedly, this is difficult to justify quantitatively, but a programming comparison of this machine with Holland's will probably be convincing. For the skeptic who does not wish to make such a comparison, however, justification probably depends on whether or not a compiler can be developed for this machine. It is anticipated that in the near future an associated paper will devote itself to just this problem.
2. Machine size (i.e., amount of hardware) has been reduced by a factor of five. While no detailed designs have been attempted, this is an estimate of the effect of the fact that the arithmetic and decoding capability has been removed from the array modules and centered in a relatively few A-boxes at one side of the array.
3. Hardware utilization has been improved by a factor of three. This also is primarily due to the fact that no longer are all modules provided with the arithmetic and decoding capability, which only about 5% are able to use in any one program.
4. System performance has possibly been degraded somewhat. Probably at least 10,000 modules would be required to accomplish any reasonably complicated task. However, the system performance is based quite heavily upon the number of program sequences that can be executed simultaneously. This is restricted directly to one sequence per A-box, so that performance depends upon the number of A-boxes provided.

This report represents the first results of one study of the Holland machine organization. There remain many interesting alternative approaches and unsolved problems. Some of these are:

I/O. What is a good way to provide input/output for a system like this?

Multiple-Word Modules. Holland picked one word per module; Slotnick picked 128. A proposed scheme for utilizing eight words per module looks as though it could decrease hardware by another factor of four or so.

Indirect Addressing. Applied to the path-building concept, indirect addressing is exactly an implementation of a list memory. How might it be used effectively?

Alternative Neighborhoods. What is the effect of selecting a different set of directly connected neighbors?

Queuing. A LIFO queue falls out very nicely. What is a clever way—or even a reasonably simple one—to implement a FIFO queue?

Locally Associative Memory. There is a lot of talk about associative memories these days. What results from considering the possibility of restricting the association to one path structure at a time?

There are clearly many unique and interesting problems (both in organization and in programming) which remain to be considered about Holland's basic approach—for anyone with some spare time.

ACKNOWLEDGMENT

The author would like to express his appreciation for the very capable assistance in this study of his colleague, Mr. Jerald S. Hughes.

REFERENCES AND BIBLIOGRAPHY
(on Highly Parallel Machines)

1. J. R. BALL, R. C. BOLLINGER, T. A. JEEVES, T. C. MCREYNOLDS, and D. H. SHAFFER, "On the Use of the SOLOMON Parallel Processing Computer," *Proc. FJCC*, 1962.
2. R. A. BROUSE, "The Data Sequenced Computer," System Development Corp. Field Note #4060.
3. C. K. CHOW, "A Recognition Method Using Neighbor Dependence," *IRE TEC*, October 1962.
4. W. T. COMFORT, "Highly Parallel Machines," *Workship in Computer Organization,"* Spartan Books, Inc., 1963.
5. J. L. DIVILBISS and B. H. MCCORMICK, "Tentative Logical Realization of a Pattern Recognition Computer," *University of Michigan Engineering Summer Conference on Parallel Computers, Automata and Adaptive Systems*, June 1962.
6. G. ESTRIN, "Organization of Computer Systems—The Fixed-Plus-Variable Structure Computer," *Proc. WJCC*, 1960.
7. G. ESTRIN and C. T. LEONDES, "Annual Summary Report of Investigations in Digital Technology Research," June 1959 —May 1960, UCLA Report No. 60–82.
8. R. GONZALES, "A Multi-Layer Iterative Circuit Computer," 1963 ACM National Conference, Denver, Colo., August 28.
9. J. H. HOLLAND, "A Universal Computer Capable of Executing an Arbitrary Number of Sub-Programs Simultaneously," *Proc. EJCC*, 1959.
10. J. H. HOLLAND, "Iterative Circuit Computers," *Proc. WJCC*, 1960.
11. C. Y. LEE, "Intercommunicating Cells, Basis for a Distributed Logic Computer," *Proc. FJCC*, 1962.
12. C. Y. LEE and M. C. PAULL, "A Content Addressable Distributed Logic Memory with Applications to Information Retrieval," *Proc. IEEE*, June 1963.
13. B. H. MCCORMICK, "The Illinois Pattern Recognition Computer," 1963 ACM National Conference, Denver, Colo., August 27.
14. D. SLOTNICK, "The SOLOMON Computer, Preliminary Report," *Workshop in Computer Organization*, Spartan Books, Inc., 1963.
15. D. L. SLOTNICK, W. C. BORCK, and R. C. MCREYNOLDS, "The SOLOMON Computer," *Proc. FJCC*, 1962.
16. J. S. SQUIRE and S. M. PALAIS, "Physical and Logical Design of a Highly Parallel Computer," *Proc. SJCC*, 1963.
17. S. H. UNGER, "A Computer Oriented toward Spatial Problems," *Proc. IRE*, October 1958.
18. S. H. UNGER, "Pattern Detection and Recognition," *Proc. IRE*, October 1959.

ASSOCIATIVE LOGIC FOR HIGHLY PARALLEL SYSTEMS

R. R. Seeber and A. B. Lindquist
Development Laboratory, Data Systems Division
IBM Corporation, Poughkeepsie, New York

INTRODUCTION

Some of the advantages of a highly parallel system are well known. With several computers and memories available, double or triple redundancy can supply extremely high reliability where such reliability is a necessity. The loss of one computer will mean a reduced computing rate but no complete breakdown of the system; this desirable attribute has been called graceful deterioration. We can also have modular enhancement when the user who wishes to solve his problems more rapidly has but to add another computer module or two in parallel with his existing installation, if his problems are responsive to parallel treatment. Complex real-time problems of supreme urgency will require a high degree of parallelism if they are to be solved. Of course all of these advantages of parallel systems assume an exceedingly flexible means of programming and control for the system.

What is not so generally well known is that a high degree of parallelism will likely be the key to the economic mass fabrication of logic units in a mass fabrication technology such as cryogenics. Here the cost of a design is largely determined by its complexity rather than by the total number of elements involved. A memory plane seems economically feasible because of its low complexity, i.e., repetitive pattern. But the much less repetitive patterns of conventional arithmetic and logic units become excessively expensive for the complex design. This argues for an elementary type of arithmetic unit, repeated many times, provided that the simple controls and programming are available.

Several proposals within the field of parallel computer organization have been made. Most of these have been in the direction of parallel programming, i.e., multiprogramming, where a single computer deals with several programs, hopefully to increase throughput. H. Hellerman[1] proposes, in addition, a multiprocessing function, taking advantage of a small associative memory for memory address transformation. Unger[2] has proposed a parallel network system with central control; problems of a high degree of symmetry could be solved, with different parts of the problem being done, synchronously, in different modules. The programming for this system appears reasonable, but the very stringent symmetry relationship limits the type of problem. The Slotnick[3] proposal for the Solomon Computer proceeds along somewhat similar lines. Holland[4] has proposed a machine with distributed control for solving more general types of problems. His organization appears to have very severe difficulties in programming.

THE ALPS ORGANIZATION

The use of associative memories for data and instructions, and associative logic to control the parallel computers, offers an attractive solution to the programming difficulties usually encountered with conventional systems. The result is an organization that will permit a very high degree of parallelism, both in processors and

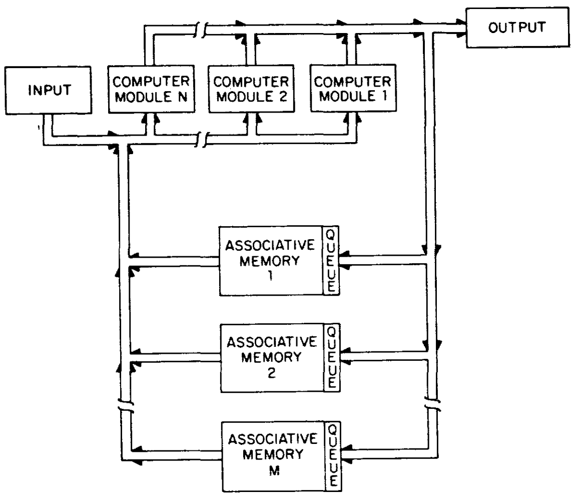


Figure 1. Block Diagram of an Associative Logic Parallel System (ALPS).

problems, with practicable programming. Figure 1 shows the block diagram of an Associative Logic Parallel System (ALPS).

ALPS, like most computing systems, will consist of five functional parts: input; output; memory; control; and computing. However, in ALPS, functional parts such as multiple computer and memory modules may be utilized without any central control. This ability of non-central control and non-distributed control (in the Holland sense), is a novel approach to parallel computing systems. These multiple functions are designed, using associative techniques, to execute one or more problems in parallel. In addition, these functions may be added or subtracted without making essential changes in the programs.

The block diagram of ALPS in Figure 1 shows the general interconnections of the functional parts. The control and computing appear in the diagram as one unit called the computer module. The interconnections will be designed so that the transfer of data among many functional parts may be carried on simultaneously.

The input section of the system will simultaneously take data from any number of input sources and store it in the storage units. Such data may come from punched cards, magnetic tape, disks, manually operated keys, sensors, radar, etc.

The output section simultaneously takes information from the storage units and puts it at the operator's disposal. The output may be in many forms such as punched cards, magnetic tape, printed forms, indicator lights, or displays.

The memory section accepts, holds, and distributes information. For this section, the block diagram shows several associative memory modules, each with an input queue. Each associative memory has the first-in first-out (FIFO) and last-in first-out (LIFO) functions, as well as many of the more usual associative features such as ordered retrieval, selective write, write into the first vacant register, etc.[5, 6] The associative memories will be designed to operate independently.

Each computer module can decode instructions it receives; send requests for data to the associative memories; operate on the data with the usual arithmetical and logical operations; and compute Tags for retrieving successor instructions and operands.

BINARY TREE PROGRAM

One programming method employs a binary tree structure. This program structure is based on earlier work for symbol manipulation.[7, 8] Several processors can be simultaneously working down the branches of the tree with the traffic controlled and the instructions and data retrieved by associative Tags and Status marks. Multidirectional, simultaneous, conditional and nonconditional branching is provided for maximum effectiveness of the parallel mode. Memory efficiency is maintained by multiple memories and memory queues for minor operations directly on words in memory.

The memory word as used in the tree program structure is set up in four fields:

Status	Tag & subscripts	Code	Item & subscripts

The Tag with its subscripts is the principal field for associative retrieval, although the associative operation will at times be applied to one or more of the other fields. The principal part of the Tag gives the name of the tree, while the subscripts show where that particular word falls in topological relationship to the rest of the tree. The Status characters are used to indicate such things as "Ready" for computation, "Branching" structures, "Constants" and "Program" constants, and "Marked" words. The Code field essentially describes the contents of the Item field; that is, whether the item is an "Operation" (+, —, etc.), a "Number", a reference to another "List" or to "Data", or is an "If" statement. If the item is a reference to another list (i.e., tree) or sublist, it will contain the Tag of that list with any necessary subscripts.

Tree Example

The organization and application of the tree structure described will be made clear by the following simple example:

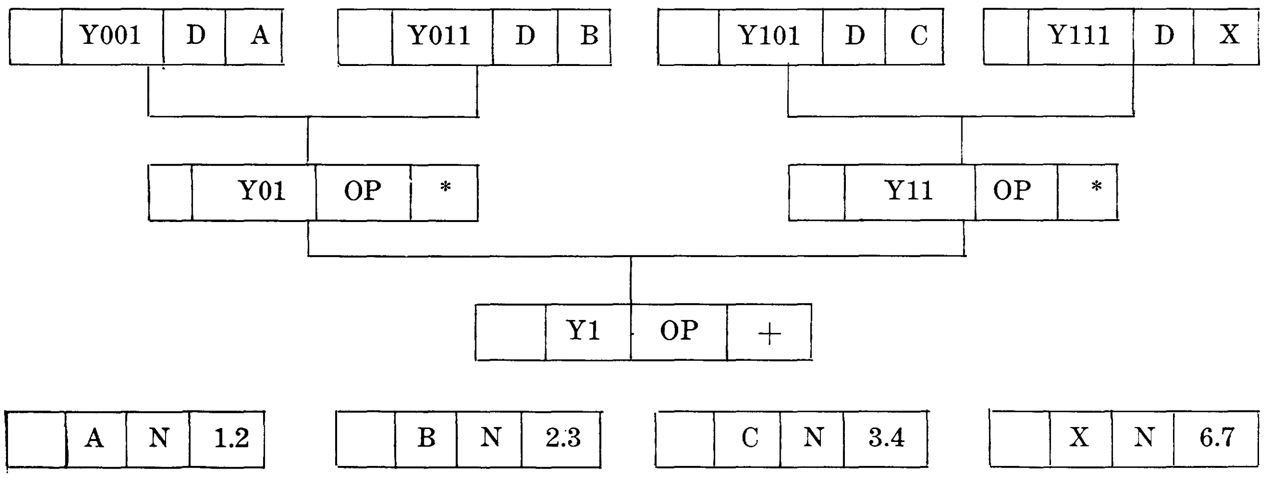
$$Y = AB + CX$$

In conventional programming, this would be represented by a serial list such as:

Load Multiplier	L (A)
Multiply	L (B)
Store	L (E)
Load Multiplier	L (C)
Multiply	L (X)
Add	L (E)
Store	L (Y)

There would also be four words in memory giving the numerical input values for A, B, C, and X. The programmer would decide which multiplication to perform first, although the order is immaterial. Note that these two multiplications can be done simultaneously if two arithmetic units with appropriate control are available.

To perform the same operations in our parallel system, the program, in tree form, would be as follows:



* indicates multiplication

The name of the tree, i.e., the Tag, is Y. The subscripts show the topological arrangement within the tree. For each set of subscripts the blanks (binary 0's), reading from the right of the subscript, are nonsignificant until we come to the first 1, which is also nonsignificant. The rest of the subscripts designate the branchings going up the tree: a 0 for a left branching and a 1 for a right branching. For example, assume that there are four subscript bits with each Tag. Consider the Tag shown as Y011. The fourth subscript bit is a nonsignificant 0 which is shown only by a blank. The third subscript bit which in this case is a 1 is also nonsignificant. The second subscript, 1, and first subscript, 0, are significant representing the right branch of a left branch. Hence, given the Tag for any word, it is easy to compute the Tag of (and thus retrieve associatively) the succeeding, preceding, or complementary word. This is a salient feature of the ALPS system. The four words with Tags A, B, C, X are the input data words. They are not shown as part of the tree.

The parallelism of the two multiplications is obvious: with two computers, we can start at the top of the tree and work our way down along the two branches. When the value for A is loaded, an interrogation on Item A will retrieve program word Y001 and generate the new

Ready word,

R	Y001	N	1.2

The Ready symbol R will indicate that Y001 is ready for computation. Similarly the words Y011, Y101 and Y111 will be generated with the Ready symbol R indicating that they are Ready for computation with the numbers 2.3, 3.4, and 6.7, substituted for B, C, and X, respectively. At this point a free computer interrogating the memory for Ready symbols can destructively retrieve word Y001. Three registers in each computer module are provided to deal with triples of words, in this case Y01 and Y011 in addition to Y001. From the Tag Y001 the Tags Y01 and Y011 are easily computed. The Tag Y01 enables the retrieval of the instruction

word

Y01	OP	*

. With Tag Y011 and Code N, an interrogation is made to determine the availability of the appropriate word. If the loading of the B word has been completed, the retrieval is successful, the multiplication will be completed, and the result 2.8 retained temporarily in one of the three registers, now with the Tag Y01. While this multiplication was taking place, another computer may have found the Ready marks on words Y101 and Y111 and will be working on the multiplication called for by Y11. The first computer will now retrieve instruction word Y1 and, with Tag Y11 and Code N, will make an interrogation to see if the appropriate word is available for its other operand. With this interrogation unsuccessful, the first computer will dump (FIFO) its partial result in memory with the Ready status marked and with Tag Y01. The first computer is then free to look for other words with Ready marks, while the second computer will find the necessary partial result available when it makes its interrogation. Thus the second computer can continue to the final result.

Iterative Loop

Another example of a tree-structured program involves a square root loop. The square root loop is an iterative loop where the number of iterations is based on a maximum predetermined absolute error. This program uses the novel feature of simultaneous multiple branching, which is available in ALPS.

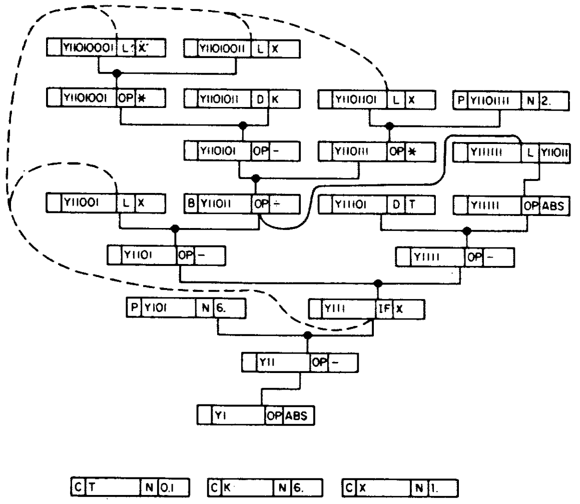
Figure 2 shows a tree representation of a program to solve the following equation.

$Y = | 6 - \sqrt{K} |$ where

$\sqrt{K} = X_{j+1} = X_j - \Delta_j$ if $T - | \Delta_j | > 0)$

T = predetermined absolute error

$$\Delta_j = \frac{X_j^2 - K}{2X_j}$$



$\sqrt{K} = X_{j+1} = X_j - \Delta_j$ IF $T - |\Delta_j| \geq 0$

$\Delta_j = \frac{X_j^2 - K}{2X_j}$ GIVEN : T, K, AND X_0

ABS	Absolute Value Operation	L	List Tag
B	Unconditional Branch	N	Number
D	Data Tag	OP	Operation
IF	Conditional Branch	P	Program Constant
C	Constant		

Figure 2. Tree Representation of $Y = | 6 - \sqrt{K} |$

The basic operations for this program are the same as those described earlier for the simple example. Operations will be initiated by the loading of T, K, and X. However, several important additional functions come into the operation.

Generated data words are normally read destructively as they are retrieved. However, if they are derived from input constants (Status C), they are held in memory for reuse and are deleted only when a new value is brought in from the input. For example, the word

C	Y11101	N	0.1

is generated from the input T. Program constants and program instruction words are deleted by program control only when all use of them has been completed.

The Unconditional Branch (note in Figure 2 the word with Status B and Tag Y11011) has a meaning different from that used in conventional programming. Here we have a two-way branch and we want in each case to go in both directions in parallel, employing the number generated at the branch point. One branching takes us down the tree in the usual fashion; the other follows the curved line to the word with Tag Y111111. This connection is found because Item Y11011 of this word agrees with the Tag Y11011 of the word initiating the branch. The word is found by associative interrogation on the Item field. The Code L indicates that the Item is a reference to another list.

The Conditional Branch appears in the word with Tag Y111, Code IF. This is a one-way versus four-way conditional branch. The conditioning is determined by the sign of the operand generated at the right branch. If the sign is positive, our iteration is complete, and we proceed down the tree employing the value given by the left branch. But if the sign of the right operand is negative, the iteration must be repeated and branching occurs four ways in parallel as indicated by the dotted curved lines. The value given by the left branch is used in all four cases. For this IF instruction, the Item X serves to locate, associatively, the four words with Items X, thus closing the loop.

The loading of a new value for K can serve to reinitiate the computation; or all three parameters, T, K, and X, may be reloaded to start a new block of computations.

SUMMARY

We have presented a brief description of the organization of an associative logic parallel system. This organization provides for the concurrent solutions of one or many problems using more than one computer. One approach to the programming of such a system was described, and two examples were presented.

The main feature of the system is its autonomous control implemented through associative logic. It is anticipated that the programming will not be too difficult, nor will there be substantial changes in programming because of modular expansion.

REFERENCES

1. H. HELLERMAN, "On the Organization of a Multiprogramming-Multiprocessing System," IBM Research Report RC-522.
2. S. H. UNGER, "A Computer Oriented Toward Spatial Problems," Proc. IRE, October 1958.
3. D. L. SLOTNICK et al., "The Solomon Computer," Proc. FJCC, 1962.
4. J. H. HOLLAND, "A Universal Computer Capable of Executing an Arbitrary Number of Sub-Programs Simultaneously," Proc. EJCC, 1959.
5. R. R. SEEBER and A. B. LINDQUIST, "Associative Memory with Ordered Retrieval," IBM Journal, January 1962.
6. G. ESTRIN and R. FULLER, "Algorithms for Content-Addressable Memories," Pacific Coast Computer Conference, 1963.
7. R. R. SEEBER, "Symbol Manipulation with an Associative Memory," National Conference of the Association for Computing Machinery, September 1961.
8. L. HELLERMAN, "A Computer Analytic Method for Solving Differential Equations," EJCC, 1959.

SOME APPLICATIONS FOR CONTENT-ADDRESSABLE MEMORIES*

G. Estrin† *and R. H. Fuller*‡

1. INTRODUCTION

In this paper we investigate several uses for a content-addressable memory (CAM) as a component of the fixed-plus-variable computing system proposed by G. Esterin.[1] Objectives of the paper are to:

1. Delineate some areas of application for a CAM.
2. Describe efficient CAM algorithms for solution of problems in these areas.
3. Compare efficiencies of CAM algorithms to those for conventional computers and for related cellular computers.

The fixed-plus-variable computing system consists of an IBM 7090 as the fixed part, a set of restructurable digital modules as the variable part and a supervisory control which sequences operations within the fixed and variable parts and controls data transfer between parts. The CAM memory would be a component of the variable part.

The CAM memory consists of a memory matrix, a detector plane, circuitry for resolution of multiple matches, word read-write drivers, bit sense amplifiers and drivers, a mask register, a data register, and a control unit. Organization of this circuitry was described previously[2] together with a set of "basic" commands directly executable in the memory and algorithms for "compound" commands executable as sequences of basic commands. The basic commands and a set of compound commands are defined in Appendix A.

The basic command set differs from sets for CAM memories described by others[3, 4, 5, 6] in that it allows writing into a program-selected bit of many words simultaneously. The set of written words may include all words in memory (CLC command), words matching a sequence of search criteria (WSB command) or, if the set of words forms a two-dimensional array, any or all of the four nearest neighbors to words matching a sequence of search criteria (WSB command). The CLC command clears "control bits" defined within the memory matrix.

Compound commands allow simultaneous comparison of an arbitrary set of memory words to an external key in any of the senses, $>$, $\geq$, $<$, $\leq$. Binary arithmetic and logical operations (two operands) may also be executed simultaneously for many operands paired in the following senses.

1. One operand of a pair is an externally derived word common to all pairs. The

* The research work for this paper was carried out within the Digital Technology group at UCLA and was sponsored in part by the Department of the Navy, Office of Naval Research, and the Atomic Energy Commission, Division of Research, under Contract Number Nonr-233(52). Additional funding was supplied by a grant from Telemeter Magnetics Incorporated.

† Professor of Engineering, University of California, Los Angeles.

‡ Staff Engineer, Research and Systems Center, Information Systems Group, General Precision, Inc., Glendale 1, California (formerly Department of Engineering, University of California, Los Angeles).

second operand is stored in each CAM cell of a designated set.

2. Operands of a pair are each stored in a CAM cell of a designated set.
3. One operand is stored in each cell of a designated set. The second operand is stored in cells having a specified location relative to members of the designated set.

When a CAM is included within the fixed-plus-variable structure, a programmer may use the sequential arithmetic and logic capability of the IBM 7090 as well as the parallel processing capability of a CAM. Also, since compound commands are all synthesized from a few basic commands, the set of compound commands may be optimally structured for each problem of interest.

To find applications for CAM we note the following of its properties.

1. Stored data are addressed by means independent of their physical location.
2. Multimembered sets of data may be addressed.
3. Arithmetic and logical operations are executed over many sets of arguments simultaneously.

The ability to address data by means independent of location is also found in such list processing languages as IPL-V,[7] LISP[8] and COMIT.[9] The ability to address multimembered sets of data and simultaneously execute arithmetic or logical operations on these is also found in such cellular computers as Unger's Spatial Computer,[10] the Solomon Computer,[11] McCormick's Pattern Recognition Computer,[17] and the Holland machine.[12] In this paper we illustrate uses for the parallel processing capability of CAM and, in particular, for communication between neighboring word cells. The uses of location independent addressing will be discussed elsewhere.

As an example of a problem which may use the parallel logic capability of CAM, we consider a task of function extremization arising as a repeated subtask in system optimization by dynamic programming techniques. The MAXA (or MINA) command for CAM is used to examine the global structure of the function as an aid to extremization.

The parallel logic capability of CAM is also of use in visual pattern recognition. Local as well as global structure of a pattern is of importance. Operands for logic operations include elements of discretized patterns and their nearest neighbors.

To evaluate the arithmetic capability of CAM we chose to solve elliptic difference equations using point relaxation methods. The CAM relaxation algorithm requires operands to be paired in all senses mentioned above.

2. FUNCTION OPTIMIZATION

Dynamic programming techniques were introduced by R. Bellman[13] and offer the hope for a powerful tool in solution of optimization problems arising in operations research and in analysis and synthesis of complex automatic control systems. However, for many problems of practical interest, computing algorithms based on these techniques have execution times and memory requirements which make them impractical of solution on current digital computers. In this section an algorithm is presented which employs CAM commands for solution of recurrence relations commonly arising in dynamic programming problems. The performance of this algorithm in optimization of a simple final-value nonlinear control system is evaluated by simulation. Techniques used in optimization of this system are extendable to a variety of systems and optimization criteria.

The system equations for the control system to be studied are a set of nonlinear difference equations derived from the Van der Pol equation,

$$X^{+} = X + \triangle Y + G$$
$$Y^{+} = Y + \triangle [- (X^2 - 1) Y - X] + H \quad (1)$$

where: X and Y are state variables (perhaps position and velocity) at time, t; X^{+} and Y^{+} are state variables at time $t + \triangle$; $\triangle$ is a time step; and G and H are control variables.

The state variables (X, Y) and the control variables (G, H) assume discrete values on grids:

$$X = - RS\ (\triangle_s)\ RS$$
$$Y = - RS\ (\triangle_s)\ RS$$
$$G = - RC\ (\triangle_c)\ RC$$
$$H = - RC\ (\triangle_c)\ RC$$

which for convenience we assume to be uniform. The quantities RS and $\triangle_s$ are, respectively, the range and grid spacing for discretized state variables (X, Y): The quantities RC and $\triangle_c$ define a similar set of grid points for discretized control variables (G, H). The state variables (X^+, Y^+) assume continuous values over the range:

$$-RS \leq X^+ \leq RS$$
$$-RS \leq Y^+ \leq RS$$

Starting at time, t, from an arbitrary discretized state point X(t), Y(t) the system moves to the state [X (T), Y (T)] at time T, where $T-t = n\triangle$. It is desired to maximize some object function:

$$f_0 [X(T), Y(T)]$$

by choice of optimal allowable control variables G and H at times $t, t + \triangle, \ldots, t + (n-1)\triangle$. The optimal object function, f_0, and the optimal sequence of control variables (G, H) is to be determined for each allowable starting point [X(t), Y(t)].

The dynamic programming formulation of this problem treats the multistage decision process by considering a series of related single-stage decision processes. An optimal control policy is determined from the "principle of optimality" which states, "An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision."[13]

If the system is at a discretized state point (X, Y) at time $t + (n-1)\triangle$ with one decision remaining, the object function f_1 [X(T), Y(T)] may be explicitly determined as

$$f_1 (X, Y) = \underset{G,\, H}{\text{Max}} \left[f_0 (X^+, Y^+) \right] \qquad (2)$$

where

$$G = -RC\ (\triangle_c)\ RC$$
$$H = -RC\ (\triangle_c)\ RC$$

The quantities X^+ and Y^+ are functions of the control variables G and H as defined by Equation 1.*

* We henceforth treat only single-stage transitions from time t to time $t + \triangle$ and again suppress the time dependence of state variables as in Equation 1. It is also convenient to suppress the dependence of quantities $X+$ and $Y+$ on the control variables.

If the system is at state point (X, Y) with "k" decisions remaining, the optimal object function and control policy may be determined recursively, by application of the principle of optimality as

$$f_k (X, Y) = \underset{G,\, H}{\text{Max}} \left[f_{k-1} (X^+, Y^+) \right] \qquad (3)$$

where

$$G = -RC\ (\triangle_c)\ RC$$
$$H = -RC\ (\triangle_c)\ RC$$
$$k = 2, 3, \ldots, n$$

The desired object function is f_n (X, Y).

Intermediate object functions, f_1 through f_{n-1}, are initially evaluated only at a limited number of discretized state points (X, Y) in order to conserve computation time and storage. If values for these functions are subsequently required at points (X^+, Y^+), which do not coincide with discretized points, they are obtained by interpolation.

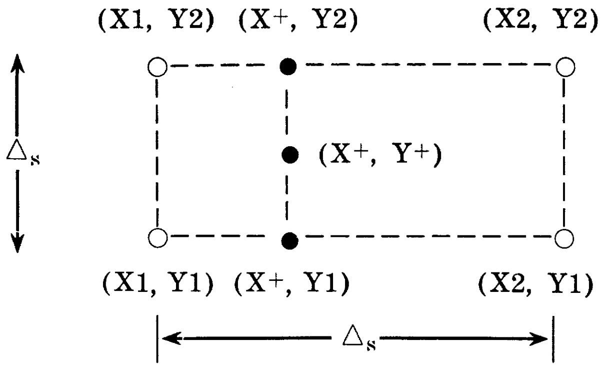
Let there be N_s points in the discretized state space and N_c points in the discretized control space. The function maximization described by Equation 3 is repeated $(n-1)\ N_s$ times, each time over N_c points.

The time to maximize a single object function over $N_c = 81$ points in discretized control space is approximately 18 milliseconds in the IBM 7090. Approximately 80% of this time is spent in interpolation. If we further assume a number of discretized state points $N_s = 10{,}000$ and a number of recursions $n = 20$, the total optimization task for this simple problem requires approximately one hour. Since interpolation requires the largest part of execution time, it is desirable to consider rejection criteria which may eliminate the need to interpolate for all functions $f_{k-1} (X^+, Y^+)$.

2.1 *Use of CAM in Solution of a Control Problem*

We shall now discuss the use of a small CAM memory in implementing some suitable rejection criteria for linear interpolation.† One rejection criterion may be stated as follows:

† For higher-than-linear interpolation, cases arise for which this rejection criterion is not valid.



$$f_{k-1}(X^+, Y1) = f_{k-1}(X1, Y1) + \frac{[f_{k-1}(X2, Y1) - f_{k-1}(X1, Y1)]}{\triangle_S}(X^+ - X1)$$

$$f_{k-1}(X^+, Y2) = f_{k-1}(X1, Y2) + \frac{[f_{k-1}(X2, Y2) - f_{k-1}(X1, Y2)]}{\triangle_S} X^+ - X2$$

$$f_{k-1}(X^+, Y^+) = f_{k-1}(X^+, Y1) + \frac{[f_{k-1}(X^+, Y2) - f_{k-1}(X^+, Y1)]}{\triangle_S}(Y^+ - Y1)$$

Figure 1. Interpolation Neighborhood for $f_{k-1}(X+, Y+)$

Criterion I

1. Consider that f_{k-1} (X+, Y+) has been evaluated for some fraction of the allowable control variables (G, H). Let f_m be the largest of these values.
2. Let a state point (X+, Y+) be determined for some new control variable.
3. If $f_{k-1}(X, Y)$ for each point (X, Y) in the interpolation neighborhood (see figure 1) of (X+, Y+) is less than f_m, f_{k-1}(X+, Y+) must be less than f_m and need not be evaluated. The function f_m remains unchanged.

Interpolations are performed in an order determined by function structure starting at that neighborhood having the largest function value, proceeding through neighborhoods having progressively lower maximum function values at discretized state points until all stored function values are less than a previously calculated maximum. When Criterion I is met, the maximum function value has been determined and further interpolations are unnecessary. If the required bivariate interpolations are performed as successive univariate interpolations, a second rejection criterion (Criterion II) can be applied. When Criterion II is met, a given sequence of three univariate interpolations is terminated and the search for a maximum is resumed. Criterion II may be stated as follows:

Criterion II

Given some f_{k-1} (X′, Y′) > f_m, the interpolation sequence required to evaluate some f_{k-1}(X+, Y+) and executed over a neighborhood containing the discretized state point (X′, Y′) may be terminated if a univariate interpolation using the value f_{k-1}(X′ Y′) yields a function value less than f_m.

The discarded value f_{k-1}(X+, Y+) can only be greater than f_m if the function f_{k-1}(X, Y) at some neighbor point other than (X′, Y′) is greater than f_m. In this instance, either the second point will be identified in a subsequent search for a maximum and the interpolation sequence repeated; or a new maximum f_{k-1}(X+, Y+) will have been determined in an intervening interpolation sequence such that no function value in the interpolation neighborhood of the original point (X′, Y′) is now greater than f_m.

Execution time for the CAM algorithm is strongly dependent on the character of the object function. Times were evaluated by simulation using synthetic object functions generated as follows:

Method 1. Function values, f_k, were chosen from a rectangular distribution of random numbers, $0 \leq f_k < 1$.

Method 2. Function values were computed from a bivariate cubic polynomial of the form

$$f_k = a_1 X^3 + a_2 Y^3 + a_3 X^2 Y + a_4 X Y^2 + a_5 X^2 + a_6 Y^2 + a_7 XY + a_8 X + a_9 Y + a_{10}$$

Coefficients were chosen, for each state processed, from a rectangular distribution of random numbers, $0 \leq a_i < 1, i = 1, 2, \ldots, 10$.

Method 3. Function values were in floating-point notation. Exponents and fractions were independently chosen from rectangular distributions of random numbers.

Results from trial optimizations using each object function are summarized in Table 1 which shows the average number of univariate interpolations performed to maximize each object function for some state point. Averages are also expressed as percentages of the number of interpolations which would be required without rejection criteria (243 interpolations would be required for the chosen problem constants). Data are presented to indicate the efficiency of Criterion I alone and of Criteria I and II jointly.

Table 1
EFFICIENCY OF REJECTION CRITERIA

	Criterion I		Criteria I and II	
Function	Average No.	Average %	Average No.	Average %
1	94.0	38.6	50.7	21
2	4.45	1.8	3.96	1.6
3	10.1	4.15	9.14	3.76

Each criterion significantly reduces required interpolations for all object functions considered. Rejection criteria are most efficient if the covering object function (i.e., a function defined almost everywhere in continuous state space which equals the discretized function at grid points) is analytic as in Function 2. Comparing efficiencies for object functions 1 and 3, it is seen that rejection criteria are also efficient if the covering function has sharp discontinuities with a few dominant maxima as has Function 3.

The CAM memory required for this example would contain 324 words, each approximately 72 bits in length.

3. VISUAL PATTERN RECOGNITION

We now consider use of the CAM system in the task of visual pattern recognition. Patterns may be alphanumeric characters,[14, 15, 16] photographs of events occurring in spark chambers or bubble chambers used for instrumentation in high energy physics,[17] microphotographs of biological cell structures,[18] etc. It might be desired to classify an input pattern into one of several predetermined groupings (pattern recognition) or to select patterns from an arbitrary set which fall into a particular class (pattern detection).

Pattern recognition entails selection of a set of pattern properties suitable for pattern classification, provision of means for determining the properties of a given pattern and provision of a mechanism for classification based on pattern properties. In the following, we consider uses of CAM in determining properties which have been shown to be of use in recognition and detection of patterns.

We desire pattern properties to be invariant to such equivalence transformations as:[15, 16]

1. Over-all size, between wide limits.
2. Position.
3. Orientation, between limits.‡
4. Proportions of component parts (e.g., line width), between limits.

For ideal pattern recognition, each identification property should partition the alphabet of allowable characters into two groups roughly equal in size and no two properties should yield similar partitions. The degree to which one can define and implement detection of a strong set of characteristics which are not sensitive to expected variation determines the power of the pattern recognition process. The properties proposed and used by Unger[14] are representa-

‡ Orientation may serve to distinguish two classes (e.g., 6 and 9 in some renderings). These classes must not be equated.

tive of features which many authors[15, 16] have used. Unger's properties include:

1. Sequence of edges.
2. Possession of holes with a given multiplicity and orientation.
3. Possession of cavities which open in a given direction.

In Table 2 we present a group of pattern transformations executable in CAM which permit programmed tests of properties suggested by Unger. These transformations closely resemble the command set proposed for Unger's spatial computer (SPAC). The reader may consult Unger's papers[10, 14] for discussion of methods for implementing property tests by use of these transformations.

A machine somewhat similar to SPAC is the Pattern Articulation Unit proposed by McCormick[17] for processing of bubble chamber photographs. McCormick's commands are also executable in CAM but are not discussed for lack of space.

The SPAC computer contains a set of one-bit principal registers (PR's) organized on a two-dimensional grid. Each PR has an associated memory register. Logic operations are defined for operands taken from PR's and their memory registers and from nearest neighbor PR's and their memory registers. We equate the PR's of SPAC to the detector plane of CAM and memory registers of SPAC to the control field within the CAM memory matrix.

The SPAC computer includes a link flip-flop between each pair of PR's. These flip-flops are used in execution of "Link" and "Expand" commands within Unger's set. For a Link command, link flip-flops are set to "1" if both linked PR's contain a "1" and to "0" otherwise. For an Expand command, all PR's linked to PR's containing "1's" through true link flip-flops are set to "1". Expansions are directed and may proceed in any combination of horizontal, vertical or diagonal directions. It is thus possible to execute the link command for a set of stored figures, perform transformations which erase undesired figures but leave some part of desired figures, then expand to recover the whole of desired figures.

There are several differences between the original SPAC commands and the CAM version

Table 2

PATTERN RECOGNITION COMMANDS

1. *WRT (TAG):* Copy contents of a single control bit C_i specified by the tag into the detector plane.
2. *STR (TAG):* Store contents of the detector plane in the control bit C_i.
3. *INV:* Complement contents of the detectector plane.
4. *ADM (TAG):* Add contents of the detector plane to contents of a control bit C_i. The addition is modulo 2. The detector plane is unchanged.
5. *MPM (TAG):* Multiply contents of each detector element by contents of control bit, C_i. The detector plane is unchanged.
6. *ADD (L. TAG):* Add (modulo 2) contents of bit C_i in cells at relative address L to contents of detector elements. A tag of "0" specifies the neighboring PR. The result appears in the detector plane and control bits are unchanged.
7. *MPY (L. TAG):* Multiply contents of bit C_i in cells at relative address L by contents of detector elements. A tag of "0" specifies the neighboring PR. The result appears in the detector plane and control bits are unchanged.
8. *SHF (L):* Shift contents of each detector element to its neighbor at address L.
9. *EDG (L):* Multiply contents of each detector element by the complement of element contents at address L. This command retains "1's" in the discretized pattern which have "0's" in the element at address 1, hence are edges of the pattern. The result appears in the detector.
10. *EXP (L. TAG):* Set a detector element true if its control bits match the tag and the element is at location L relative to some true detector element. In contrast to other commands of this set, the tag may have an arbitrary number of "1's" and the parameter L may specify any combination of the eight nearest neighbors to a cell. Repeat the command until a step is reached at which no detector element is set true.
11. *ADR:* Set the four "corner" detector elements to "1".

of these commands (Table 3). The edge command EDG is added to the CAM set to improve the efficiency of a common operation. Unger programs this command using other commands in the set. Also, the expand command EXP is a more economic though slower alternative to "Link" and "Expand" commands provided by Unger.

In the CAM set, the equivalent of linking is storage of the original pattern in some control bit (STR). Upon completion of transformations, the CAM Expand command is issued to recover figures from the original not erased by the transformations. The expansion is directed by use of the parameter L which specifies any combination of the eight nearest neighbors to a true PR as potential members of the output set. Members of the output set must also match the tag (i.e., be members of the original pattern). The tag may consist of any combination of control bits. Expansion is performed by sequentially finding neighbors of true PR's, then neighbors of these, etc., until a step is reached at which no members are added to the output set. At times it is desired to expand about known reference elements. These are set by use of the ADR command. It would, of course, be possible to incorporate link flip-flops into CAM and execute Link and Expand commands as proposed by Unger. The relative efficiency of CAM (without link flip-flops) is considered in the conclusion of this section.

We now present algorithms for the CAM commands EDG and EXP as examples of the way in which basic commands of Appendix A are used to implement pattern recognition commands of table 2.[§]

We define cells having a true detector element as members of a set S_0 and cells at address L relative to these as members of a set S_1.

EDG Algorithm

1. Clear control bits C1 and C2 (CLC).
2. Write C1 = "1" for cells in S_0 (WSB).
3. Write C2 = "1" for cells in S_1 (WSB).
4. Search cells having C1 = "1" and C2 = "0".

§ An exception is the ADR command which is itself a basic command.

EXP Algorithm

1. Clear control bits C1 and C2.
2. Write C1 = "1" for cells in S_0.
3. Write C2 = "1" for cells in S_1.
4. Search cells having C1 = "0" and C2 = "1". If no such cells exist, search cells having C1 = "1" and exit. Otherwise, return to step (2).

Unger estimates that, if SPAC were fabricated from advanced componentry, each command could be executed in approximately one microsecond. Execution times for the related CAM commands in four 1000-word memories of Appendix B are listed in Table 3. For a square array of N^2 elements, the expand command is assumed to be iterated N/2 times.

Table 3

EXECUTION TIMES* FOR PATTERN RECOGNITION COMMANDS

Command	TDCAM	FCCAM	PFCAM	CRCAM
WRT	0.530	0.720	0.560	0.130
STR	0.150	3.100	0.340	0.410
INV	0.680	3.800	0.900	0.540
ADM	0.075	1.800	0.170	0.350
MPM	1.300	6.300	1.700	1.000
ADD	1.300	6.300	1.700	1.000
MPY	1.300	6.300	1.700	0.580
SHF	0.680	3.800	0.900	0.540
EDG	0.790	5.600	1.100	0.900
EXP	30.0	290.0	45.0	35.00
ADR	0.070	0.070	0.070	0.070

* Expressed in microseconds.

3.1 *Summary of Pattern Recognition*

Pattern recognition commands other than EXP are executed in CAM systems other than FCCAM in approximately the same time as in SPAC. If average execution time for the EXP command is 50 microseconds and approximately one-tenth of all commands in a pattern recognition program are EXP commands, the pattern recognition rate of CAM is approximately one-sixth that of SPAC. These estimates seem justified for programs discussed by Unger. If link circuitry were mechanized in CAM, recognition rates in CAM would be comparable to those for SPAC in systems other than FCCAM.

Unger estimates that SPAC would require approximately 170 gate inputs and 11 memory elements per word cell if six-bit registers were provided at each word cell. Link circuitry requires approximately 60 gate inputs and four memory elements of this total. A CAM system requires one detector element, a word driver and approximately 25 gate inputs per word cell (assuming communication to the eight nearest neighbors) in addition to the memory matrix, registers, sense amplifiers and drivers shared by all word cells. A CAM system thus shows promise as a means for implementing the SPAC command set at lower cost though appreciable loss in speed relative to the mechanization proposed by Unger.

Pattern recognition has been experimentally programmed on conventional digital computers by Dineen,[15] Kirsch[16] and Unger,[14] among others. It is obvious that, due to the large number of local bit-wise logic operations required on a grid of practical size, pattern recognition is not efficient on a conventional machine. Unger's commands simulated on the IBM 704 have an average execution time of approximately 10 milliseconds. Dineen[15] programmed slightly more complex pattern transformations for execution on the MIT Memory Test Computer in several seconds.

4. SOLUTION OF ELLIPTIC PARTIAL DIFFERENTIAL EQUATIONS

The solution of partial differential equations is a frequent and time-consuming computational task. Equations of elliptic type arise in studies of fluid flow,[19] stress analysis,[20] atomic physics,[21] and recently in numerical weather forecasting.[21, 22] Two particularly important elliptic equations are Laplace's equation

$$\nabla^2 V(r) = 0$$

and Poisson's equation

$$\nabla^2 V(r) = F(r)$$

where V is a potential, r is a vector and ∇^2 is the familiar Laplacian operator. In this section we discuss methods for solving Laplace's equation in CAM. We choose to solve Laplace's equation with Dirichlet boundary conditions (i.e., known boundary potential) because of its importance and because many methods are known for its digital solution. The chosen solution method is simply modified for solution of Poisson's equation. As a computational example, we solve the Dirichlet problem for Laplace's equation in two dimensions with rectangular boundaries.

We are to find a potential which satisfied Laplace's equation on an open region R bounded by a simple closed curve C upon which we know the potential. The problem is discretized by covering the xy plane with a net of nodes which are intersections of the lines $x = ih$, $y= jh$ $(i, j = 0, \pm 1, \pm 2, \ldots)$. We denote the set of nodes (x, y) within R as the set $\overline{R}$ and the intersection of grid lines with the boundary as the set $\overline{C}$. We assume for simplicity that the boundary C follows grid lines such that points in $\overline{C}$ are nodes. We denote an arbitrary node (x, y) within $\overline{R}$ as P. The four nearest neighbors to P within the union of sets $\overline{R}$ and $\overline{C}$ are numbered P_1, P_2, P_3, P_4, starting at the node $(x + h, y)$. For potentials within a linear uniform media and for uniform net spacing replace each (V2) partial derivative ‖ in Laplace's equation

$$\nabla^2 V(P) = V_{xx} + V_{yy}$$

by its second central difference to obtain the familiar five-point approximation to Laplace's equation

$$\nabla^2 V(P) = V(P_1) + V(P_2) + V(P_3) + V(P_4) - 4V(P) = 0 \qquad (4)$$

We now have a finite set of simultaneous linear equations to be solved for unknown node potentials.

For a discussion of errors which arise from discretization, treatment of boundaries which do not coincide with grid lines, and the use of nonuniform nets, see Forsythe and Wasou,[22] Sections 23, 20.9 and 25.6, respectively.

Turning now to methods for solving the set of simultaneous equations resulting from discretization, we note that the coefficient matrix is sparse. Solution by direct methods (e.g., Gaussian elimination) is not efficient.

Several popular methods of solution involve iterative procedures which, with sufficient repetitions, should converge to a satisfactory approximation of the desired solution. One

‖ The symbol V_{xx} denotes the second partial derivative of V with respect to x.

such method has been termed "Simultaneous displacement." Node potentials are specified for all nodes in $\overline{C}$ and an estimate (perhaps 0) is made for potentials at nodes in $\overline{R}$. Potentials for all nodes in $\overline{R}$ are then recomputed by Equation 5 using known potentials in $\overline{C}$ and estimated potentials in $\overline{R}$:

$$V_k(P) = 1/4 \sum_{i=1}^{4} V_{k-1}(P_i) \quad (5)$$

where $V_k(P)$ is the potential computed for node P at the k th iteration. Each succeeding iteration is performed using potentials determined at the previous iteration.

The convergence rate may be improved by using new data as soon as it is computed. In this instance, the rate of convergence depends critically on the order in which node potentials are computed. A common ordering, and one which is particularly efficient for computation in CAM, is achieved by partitioning nodes ($x = ih$, $y = jh$) in $\overline{R}$ into sets S_1 and S_2 based on the parity of the quantity, $i + j$. For $i + j$ even, nodes (x, y) are members of the set S_1. For $i = i$ odd, nodes (x, y) are members of the set S_2. The resulting sets form a "checker-board" pattern with members of S_1 corresponding (for example) to red squares and members of S_2 corresponding to black squares. The algorithm proceeds by first computing potentials for nodes in S_1 using the most recent value for potentials of nodes in S_2 and $\overline{C}$ then similarly computing potentials for nodes in S_2. Node potentials in $\overline{C}$, of course, are not changed. The faster convergence of this algorithm is heuristically justified by the fact that "fresher" data is used during the latter half of each iteration. The method is known as "successive displacement" or Gauss-Seidel relaxation.

The convergence rate for successive displacement (though not for simultaneous displacement) may be significantly improved by the use of an "overrelaxation factor," ω. Equation 5 is modified to read:

$$V_k(P) = V_{k-1}(P) + \frac{\omega}{4}\left[\sum_{i=i}^{4} V_{k-1}(P_i) - 4V_{k-1}(P)\right] \quad (6)$$

and the algorithm proceeds otherwise as for successive displacement. This iterative procedure, termed "successive overrelaxation," was proposed by Young[23, 24] and by Frankel[25] who showed that fastest convergence is obtained for some optimum overrelaxation factor $\omega_{opt}(1 < \omega_{opt} < 2)$. This factor is generally unknown and must be determined by computational experiments. When we treat successive overrelaxation below, we assume that experiments have been performed such that ω_{opt} is known.

The simplest iterative method for use in CAM would be simultaneous displacement. However, in view of its greater efficiency, we prefer successive overrelaxation. For a conventional computer using the successive overrelaxation algorithm, each interior node must be relaxed sequentially by use of Equation 6. Ball et al.[26] estimate that a node may be relaxed in approximately 80 microseconds using the IBM 7090.

In the following we devise a CAM algorithm for solution of the Dirichlet problem by successive overrelaxation and compare execution time for this to time in a conventional computer and to time in the Solomon parallel processor.

4.1 *Solution of the Dirichlet Problem in CAM*

Word cells in CAM are organized into a two-dimension M by N array forming a spatial analog for the set of nodes. Each cell is partitioned into an A field which stores node potentials and a B field used for working storage. From a rectangular grid, the node $P(X_0 + ih, Y_0 + jh)$ with $i = 1\ (1)\ M$, $j = 1\ (1)\ N$ is assigned to the cell at location $L = (i - 1) + M\ (j - 1)$. Cells storing boundary potentials are members of set S_0. Cells storing interior potentials are member of set S_1 if $i + j$ is even or of set S_2 if $i + j$ is odd. Potentials for cells in S_1 are revised as follows:

1. Clear all B fields.
2. Multiply C (A) for cells in S_1 by $-\omega$. The result appears in B fields of cells in S_1.
3. Multiply C (A) for cells in S_0 or S_2 by $\omega/4$. The result appears in B fields of cells in S_0 or S_2.
4. Add to C (B) for each cell in S_1, C (B) for each of its four nearest neighbors in S_0 or S_2.

5. The quantity $V_k - V_{k-1}$ is now stored in B fields for cells in S_1. Test if $| V_k - V_{k-1} | < \epsilon$
6. Add C (B) to C (A) for cells in S_1.

Potentials for cells in S_2 are then revised in a similar manner.

If $| V_i - V_{k-1} | < \epsilon$ for all cells, the iterative procedure has converged satisfactorily and is ended.

For convenience in clearing, we assume that B fields are stored in control bits. The time to perform one iteration over all nodes is approximately

$$T_{CAM} = 2T_{CLC} + 4T_{MPYC} + 8T_{ADDF} + 2T_{GEK}$$

Ball et al.[26] have studied use of the Solomon computer[11] in this problem and state that iteration over 1024 nodes for fields of length p in Solomon requires a time

$$I_S = (17.5p + 3.6) \text{ microseconds}$$

In Table 4 we compare average iteration times per node in solution of a problem by optimum successive overrelaxation in the four CAMs of Appendix B with times for Solomon and for the IBM 7090. Comparisons are made for nets having 1024 nodes with potentials expressed to accuracies of 18 and 36 bits.

4.2 *Summary of Elliptic Difference Equations*

It has been shown that arithmetic commands of CAM are efficient for solution of Laplace's equation by successive overrelaxation. This algorithm has also been suggested for use in the Solomon computer where, by use of more complex cellular processing elements, its execution is significantly faster than in any CAM. Solution times for CAM are far less than for the IBM 7090.

5. CONCLUSIONS

We have demonstrated uses for parallel processing capabilities of computers equipped with CAM memories in several problems of interest. For these problems the availability of CAM increases system efficiency by factors ranging from two to several hundred.

We have also compared the efficiency of CAM memory systems to that of more complex cellularly organized machines and have shown that CAM systems offer attractive performance relative to these with significantly less complexity.

It is evident that parallel processing capabilities of cellular machines such as CAM, SPAC, or Solomon require novel algorithms for efficient problem solution. We and others have attempted to derive efficient algorithms for problems reported in this paper. However, much work remains to be done in this area as well as in further definition of useful properties for cellular machines and in efficient implementation of these machines.

6. ACKNOWLEDGEMENT

The authors gratefully acknowledge the programming efforts of Mr. Phillip Schaefer in support of this research.

BIBLIOGRAPHY

1. ESTRIN, G., "Organization of Computer Systems—The Fixed Plus Variable Structure Computer," *Proceedings of the Western Joint Computer Conference*, San Francisco, Calif., May 3-5, 1960.

Table 4

SUMMARY OF COMPUTER PERFORMANCE* IN THE DIRICHLET PROBLEM

p	TDCAM	FCCAM	PFCAM	CRCAM	Solomon	IBM 7090
18	1.3	5.8	1.7	1.0	0.31	80
36	2.6	11.6	3.4	2.0	0.60	80

* Iteration time per node in microseconds.

2. ESTRIN, G., and R. FULLER, "Algorithms for Content Addressable Memory Organizations," *Proceedings of the Pacific Computer Conference*, Pasadena, Calif., March 15-16, 1963.
3. CORBELL, R. C., "A Tunnel Diode Associative Memory," M. S. Thesis, University of California, Los Angeles, Calif., June, 1962.
4. KISEDA, J. R., H. E. PETERSEN, W. C. SEELBACH, and M. TEIG, "A Magnetic Associative Memory," *IBM Journal of Research and Development*, vol. 5, pp. 106-121.
5. DAVIES, P. M., "A Superconductive Associative Memory," *Proceedings of the Spring Joint Computer Conference*, May 1-3, 1962.
6. NEWHOUSE, V. L., and R. E. FRUIN, "A Cryogenic Data Addressed Memory," *Proceedings of the Spring Joint Computer Conference*, May 1-3, 1962.
7. NEWELL, A., *Information Processing Language—V Manual*, Prentice-Hall, Inc., Englewood Heights, N. J., 1961.
8. Computation Center and Research Laboratory of Electronics, Massachusetts Institute of Technology, *Lisp 1.5 Programmer's Manual*, Cambridge, Massachusetts, July, 1961.
9. The Research Laboratory of Electronics and The Computation Center, Massachusetts Institute of Technology, *An Introduction to COMIT Programming*, Cambridge, Massachuetts, November, 1961.
10. UNGER, S. H., "A Computer Oriented Toward Spatial Problems," *Proceedings of the IRE*, vol. 46, pp. 1744-1750, October, 1958.
11. SLOTNICK, D. L., W. C. BORCK, and R. C. MCREYNOLDS, "The Solomon Computer," *Proceedings of the Fall Joint Computer Conference*, Philadelphia, December, 1962.
12. HOLLAND, J. R., "Iterative Circuit Computers," *Proceedings of the Western Joint Computer Conference*, San Francisco, Calif., May 3-5, 1960.
13. BELLMAN, R., *Dynamic Programming*, Princeton University Press, 1957.
14. UNGER, S. H., "Pattern Recognition and Detection," *Proceedings of the IRE*, vol. 47, pp. 1737-1752, October, 1959.
15. DINEEN, G. T., "Programmed Pattern Recognition," *Proceedings of the Western Joint Computer Conference*, Los Angeles, Calif., March 1-3, 1955.
16. KIRSCH, R. A., L. CAHN, C. RAY, and G. H. URBAN, "Experiments in Processing Pictorial Information with a Digital Computer," *Proceedings of the Eastern Joint Computer Conference*, Washington, D. C., December 9-13, 1957.
17. MCCORMICK, B. H., and J. L. DIVILBISS, "Tentative Logical Realization of a Pattern Recognition Computer," Report No. 403, Digital Computer Laboratory, University of Illinois, Champaign Urbana, Illinois, 1961.
18. MANSBERG, H. P., "Automatic Particle and Bacterial Colony Counter," *Science*, pp. 823-827, October 25, 1957.
19. KARPLUS, W. J., "Water-coning before Breakthrough—An Electronic Analog Treatment," *Petroleum Transactions of the American Society of Mechanical Engineers*, vol. 27, pp. 240-255.
20. CLYMER, B. A., "Operational Analog Simulation of the Vibration and Flutter of a Rectangular Multicellular Structure," *IRE Transactions on Electronic Computers*, vol. EC-8, pp. 381-390, September, 1959.
21. ALT, F. L., *Advances in Computers*, vol. 1, pp. 43-91, Academic Press, New York, 1960.
22. FORSYTHE, G. E., and W. R. WASOU, *Finite Difference Methods for Partial Differential Equations*, John Wiley and Sons, New York, 1960.
23. YOUNG, D. M., "ORDVAC Solutions of the Dirichlet Problem," *Journal of the ACM*, vol. 2, pp. 137-161, 1955.
24. YOUNG, D., "Iterative Methods for Solving Partial Difference Equations of Elliptic Type," *Transactions of the American Mathematics Society*, vol. 76, pp. 92-111, 1954.
25. FRANKEL, S. P., "Convergence Rates of Iterative Treatments of Partial Differential Equations," *Mathematical Tables and Aids to Computation*, vol. 4, pp. 65-75, 1950.
26. BALL, J. R., R. C. BOLLINGER, T. A. JEEVES, R. C. MCREYNOLDS, D. H. SHAFFER, and

D. I. CAPLAN, "On the Use of the Solomon Parallel Processing Computer," *Proceedings of the Fall Joint Computer Conference*, Philadelphia, December, 1962.

27. LONG, T. R., "Electrodeposited Memory Elements for a Non-destructive Memory," *Jour. Appl. Phys.*, supp. to vol. 31, no. 5, pp. 123S-124S, May, 1960.

APPENDIX A

CAM Command Set

In this appendix we list a set of basic CAM commands and a set of compound commands synthesized from these and having relevance to applications discussed in this paper. Additional description of the CAM command set may be found in a previous paper.[2]

Table A.1
BASIC MEMORY COMMANDS

Mnemonic Code	Command
1. SCH (Tag) (Search)	The detector plane is cleared to the matched state then reset to mismatched state for CAM cells having data contents which do not match the masked key or having control bits which do not match the masked tag. Cells remaining in the matched state are designated as an "input set" for other commands which use such a set.
2. WNW (Tag) (Write Next Word)	Unmasked bits of data register are written into the next sequential cell (according to a defined ordering) of the selected input set. Masked bits for this cell are unaltered. The masked tag is written into the control field for this cell. The word-select ladder then advances to the next matching cell.
3. WTL (Y, Tag) (Write Location)	Unmasked bits of the data register are written into the cell at location Y. Masked bits for this cell are unaltered. The masked tag is written into the control field for this cell.
4. RNW (Read Next Word)	Contents of the next sequential cell of the selected input set are read nondestructively into the data register. The word-select ladder then advances to the next matching cell.
5. RDL (Y) (Read Location)	Contents of the cell at Location Y are nondestructively read into the data register.
6. WSB (L, Tag) (Write Selected Bit)	A single defined bit is written into a defined bit position for all cells at location L relative to locations of cells in the selected input set. The parameter L may specify the input set or it may specify one of the four nearest neighbors of cells in the input set.
7. CLC (Tagm) (Clear Control Bits)	The tag consists only of the mask, Tagm. The control field is cleared to "0" in all bit positions where the mask is "1".

Table A-2
TYPICAL COMPOUND COMMANDS

Mnemonic Code	Command
1. MAXA (Absolute Maximum)	Members of some input set which have the maximum absolute value for some field are located.
2. GEK (Greater than or Equal to Key)	Members of some input set which have the numeric value for some field greater than or equal to a key are located.
3. LEK (Less than or Equal to Key)	Members of some input set which have the numeric value of some field less than or equal to a key are located.
4. ADDC (Add Constant)	Contents of some field in the D register are added to all cells in some input set.
5. ADDF (Add Fields)	Contents of two fields are added for each cell in some input set.
6. ADDP (L) (Add Pairs)	Contents of a field in words at location L relative to members of the input set are added to the the corresponding field for members of the input set.
7. MPYC (Multiply by Constant)	The product of a field in the D register by a field in each cell of an input set is formed in the cell.

APPENDIX B

Summary of CAM Realizations

In this appendix estimates are given for achievable execution times of basic commands of Table B.1 in four mechanizations of the memory matrix. Times were determined for each command by summing transmission delays in drive and sense lines and associated active circuits, element switching times, delay in the word-select ladder and match type indicator and times for required logic operations. Transmission delays were determined by analysis of equivalent lumped parameter lines. Element switching speeds and response times for active circuits were estimated, using published data when available. Times are determined for a memory having 1024 words and 128 bits per word.

The four chosen memory elements are:

1. Tunnel diodes in a circuit proposed by Corbell[3] (TDCAM).
2. Ferrite ring cores of 30 mil O.D. in a circuit proposed by Petersen[4] (FCCAM).
3. Thin Permalloy film plated on copper wire as proposed by Long[27] (PFCAM).
4. Cryotrons in a circuit proposed by Davies[5] (CRCAM).

The tunnel diode, ferrite core and plated wire memory each use tunnel diode detector elements organized as a two-dimensional array. Associated wtih both dimensions of this array are "edge registers" each position of which may be set to "1" for any "1" in a row or column of the detector place. Word-select ladders are implemented, using a tunnel diode circuit, to detect the first "1" in each edge register and transmit its address to the address selector. Operating time for this circuitry was experimentally verified. In each of these memories searches are executed serial-by-bit.

The cryotron memory uses a word-select ladder proposed by Newhouse.[6] To decrease the long propagation time reported by Newhouse, the ladder is segmented into "primary segments" of 128 words and a second ladder selects the first primary segment containing a match. We assume a plate size and packing density such that all words serviced by a primary ladder are contained on a single plate. This minimizes inductance of interconnecting leads. Repeater amplifiers are inserted in each ladder periodically to raise impedance levels and therefore decrease delay. In this memory searches are executed parallel-by-bit.

Approximate execution times for basic commands in these memories are listed in Table B.1. All times are expressed in nanoseconds. Logic operations are executed in Time T_L.

Table B.1
COMMAND EXECUTION TIMES

Time	TDCAM	FCCAM	PFCAM	CRCAM
T_{SCH}	500 + 33k*	700 + 50k*	500 + 50k*	130
T_{WNW}	330	900	340	270
T_{WTL}	150	700	150	380
T_{WSB}	75	1800	170	350
T_{RNW}	310	400	280	310
T_{RDL}	130	220	100	310
T_{CLC}	75	1300	170	60
T_L	70	70	70	70

* "k" is the number of bits searched.

A COMPUTER AID FOR SYMBOLIC MATHEMATICS

*Lewis C. Clapp and Richard Y. Kain**
Bolt Beranek and Newman, Inc.
Cambridge, Massachusetts, 02138
Telephone: Area Code 617, 491-1850

INTRODUCTION

Long before it is solved, many a problem has been tossed aside by creative scientists simply because of the sheer tedium in carrying out repeated mathematical operations. In addition to being time consuming, this "clerical" activity is a frequent source of trivial errors which can invalidate the entire analysis. One approach toward solving this problem is the creation of computer aids that will assist the scientist as he develops a theory or tries to solve a mathematical problem. However, when a scientist is trying to develop a theory he does not only want numbers or graphs that satisfy a particular case. He is most happy when he can derive a general formula that not only describes a large number of cases, but also shows how the various cases are interrelated. The theoretician often starts with a set of postulates or some initial equations and manipulates them symbolically to get resulting equations and conditions. Occasionally the scientist has to develop his own mathematics as he works out a problem by creating new symbols, operators, and functions. A computer system that will aid the scientist in the creative process must be flexible enough to accept these new rules with a minimum of effort on the scientist's part. Our work in developing one experimental system of this nature (Magic Paper 1)[1] will be described and several possible extensions to this program will be suggested.

When will the scientist prefer computer aid to working alone at his desk? He will probably wish to use such a system when he has a lengthy and complicated analysis to perform. He may prefer to work at his desk first and then repeat the calculation on the computer as a check against errors. He will need a computer system if he has a great deal of experimental numerical analysis to carry out. (Culler and Fried[2] discuss this application in detail.) But when the numerical anaylsis is straightforward, the problem is best handled by a conventional algebraic compiler with little need for the scientist's presence as the computation is being performed. On-line computing may also be useful when the scientist wants to get a feel for the various mathematical functions by seeing them plotted on the face of a display scope. Finally, the computer may be useful simply because it gives the scientist a chance to get away from his desk, to push a few buttons and still work toward the solution of his problem.

The computer and the scientist can complement each other in a remarkable way. The computer is able to carry out lengthy processes repeatedly without making an error or suffering from boredom. On the other hand, the scientist, who quickly becomes inefficient under the stress of clerical activity, has an insight and creative ability which is necessary for the successful development of a theory or body of mathematics. In developing Magic Paper 1,

* Consultant; Assistant Professor, Department of Electrical Engineering, Massachusetts Institute of Technology, Cambridge, Massachusetts.

we have incorporated many operations into the system which scientists find annoying or tedious, operations which are readily implemented on a digital computer. However, we have omitted from the preliminary system many other operations which, although programmable, are at present more efficiently handled by the scientist himself. The program is open ended in the sense that as better algorithms are developed for handling the more sophisticated processes, they can be incorporated and in fact can call upon the simpler operations already included.

We would like to point out some of the differences between Magic Paper 1 and other programs which aid the scientist as he works on various mathematical problems. Algebraic compilers (such as Fortran and Algol) are used daily by scientists and engineers who wish to obtain numerical evaluation for specific and well defined equations. These programs are most useful when the user knows in advance exactly what he wants. If he is uncertain about the best way to proceed and needs to try a number of different approaches, on-line computer aid offers a number of advantages. Culler and Fried[2] have developed a system for on-line evaluation of functions. In their system, the user is able to specify highly structured manipulations on a symbolic level. However, the data itself is numerical so that output is either a series of numbers or a plot of the curve. It should be possible for them to extend their system to handle the symbolic manipulation of equations. There is still a fundamental difference in approach between Culler's system and the one we have developed. Culler has stated at the numerical evaluation level and plans to work up to the symbolic level. Our approach is to give less emphasis to the numerical evaluation aspects by starting directly at the symbol manipulation stage.

Other groups have pointed out the importance of symbol manipulation for aiding mathematical research. Engelbart[3] sketched out a broad program of research leading to such computational aids in 1962 and pointed out that Vannevar Bush[4] suggested similar ideas in 1945. Minsky[5] at M.I.T. has emphasized the importance of this problem and is encouraging his graduate students to work in this field. This group is developing a system for mathematical manipulation of equations which includes an elaborate scope display and data input facility. We understand that they have also developed an algorithm for polynomial factorization which finds all real or complex roots. Under Fano, Project MAC at M.I.T. is also exploring this problem as part of a broad program seeking to aid the scientific community by placing a large time-shared computer at their disposal with adequate software backup. At the same time, many scientists, such as Feynman at the California Institute of Technology, have repeatedly pointed up the need for such computer aid.

Unlike many of the research programs just mentioned, Magic Paper 1 was developed for a small timed-shared (PDP-1) computer[7] without a larger computer as a backup. The smaller machines have the advantage of being inexpensive and are more readily acquired by universities and research organizations. They also seem to be adequate for the task of aiding symbolic mathematical manipulation. Furthermore, when such machines are time-shared, a relatively large number of scientists can have access to them. Perhaps the medium sized computer will eventually be supplanted by the large computing centers of the Project MAC type. But this will not happen for another few years—and the scientist must face the task of mathematical analysis even now.

OPERATIONAL DESCRIPTION

Let us first describe the operation of the system from the scientist's point of view. The scientist places himself at the console consisting of a typewriter, a display scope and a light pen. A paper tape reader and punch and several magnetic tape units are also available. The scientist initiates symbolic manipulation activity by typing one or several of the control characters listed in Figure 1. He can combine the action of a group of existing control functions and assign this action to a new control symbol. Previously developed equations may be retrieved from a magnetic tape file or the scientist may type a new equation on the typewriter and see it displayed on the scope. Using the light pen, he can underline various terms to specify which part of an expression is to be

SYMBOL	CONTROL SIGNIFICANCE
I	ENTER INPUT MODE
,	LEAVE INPUT MODE
P	DISPLAY POINTER ON SCOPE
L	LABEL EQUATION
$\rightarrow$	SUBSTITUTE
U	UNDERLINE EQUATION
R	REPLACE UNDERLINED SYMBOLS
F	FACTOR OUT UNDERLINED TERMS
G	GET EQUATION FROM STORAGE AND ADD TO BOTTOM OF DISPLAY
S	ENTER SKETCH INPUT MODE (to construct a figure)
D	DISPLAY FIGURE
T	TRANSPOSE UNDERLINED STRINGS
N	NAME UNDERLINED STRING
K	EXPUNGE EQUATION
Wn	CHANGE WINDOW SIZE TO n
M	MOVE WINDOW (u -up, d -down, b-page back, f-page front)
EVAL	EVALUATE FUNCTION
GEN	GENERATE GRAPH
c<···>i, j, ···, n	CREATE NEW CONTROL FUNCTION

Figure 1. Typical Executive Control Characters.

manipulated. Typical manipulations are insertion, substitution, multiplication of an equation by a term, transposition, and the addition of two equations. Many other operations normally required in the course of analytic computation are included. The result of processing an equation by these operations is a new equation which is also displayed on the scope. The scientist can retain the new equation or reject it and repeat the process in a different manner. He can also examine his equations graphically to get a better feeling for their behavior. For instance, he can ask that parametric curves of the equation be plotted as a function of one of the arguments, keeping all other variables constant at some specified values. The computer will generate and display the requested curves. He can also manipulate the graph, enlarging various sections, to examine it in greater detail.

The scientist is able to continue in this way as long as he likes. He can label equations, store the results, pursue a line of reasoning to its conclusion or, if the line turns out to be fruitless, abort it and pick up the thread at an earlier point in the development. At the end of the session, the scientist has several options. He can simply walk away from the machine leaving all of his data on magnetic tape in a form that is immediately usable at a subsequent session, or he can request hard copy typeouts of all or part of his results. Finally, he can request paper tape punchouts of the results. The paper tape can either be listed off-line or can be used as in the input during his next session on the computer.

SYSTEM ORGANIZATION

We turn now to the internal data and program organization of the Magic Paper System. The sketch in Figure 2 shows, in rough terms, the overall program structure. The scientist communicates directly with the executive control program which in turn interprets his commands and data and calls other portions of the system into play as needed. We will take a closer look at the program structure, after a brief description of the data organization.

One of the most important decisions that had to be made during the design of Magic Paper 1 was the choice of internal data format. The concept of list structures was quite appealing. However, nested list languages such as LISP or IPL V appeared to have much more generality than was needed for the system that we were trying to achieve. One pays for this generality in memory space and computation time. Furthermore, such languages are not presently available on medium-sized computers, and would be difficult to time-share even if they

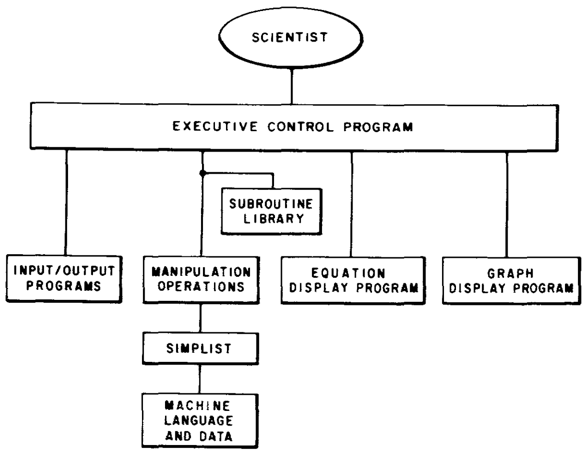


Figure 2. Program Organization.

were. Accordingly we have developed a simplified linear list language which has proved more than adequate for the magic paper type of application. This language, called SIMPLIST, will now be described.

SIMPLIST is a program which handles linear list structures through push-down and pull-up operations. Data can be added to or retrieved from either end of a list. To keep the language as simple as possible, branching within a list is not permitted; however, a list can reference other lists to any desired depth. One reason why we chose SIMPLIST for the implementation of Magic Paper 1 is that ordinary mathematical expressions are somewhat like linear lists. One normally reads and processes equations from left to right. Because of its inherent simplicity SIMPLIST itself requires a minimum of memory space (approximately 700 registers on the PDP–1), making it very attractive for medium-sized computers. It is relatively fast, and finally, it was extremely easy to write using an Algol-like compiler, DECAL-BBN.[8]

A list is a consecutive sequence of memory locations. A small set of pointers is used to keep track of the status of each list. The list structure and significance of the pointers is sketched in Figure 3. The three pointers at the left define the memory space assigned to the list. Specifically, PSEUDOBEG and PSEUDOEND point to the limits of the list area and INITBEG points to the location of the first data word added to the list after it is reset.

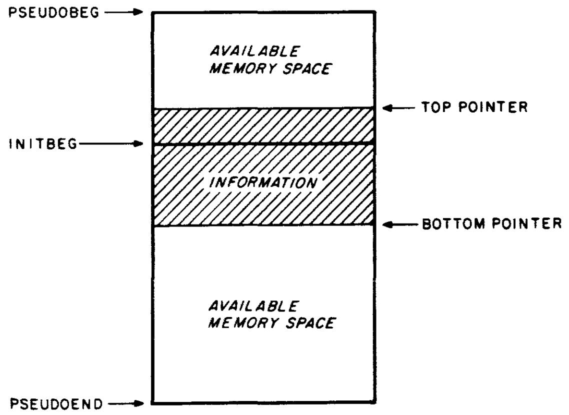


Figure 3. List Structure and Pointers.

Two data pointers for each list, which we call "top pointer" and "bottom pointer", denote the boundary of the information in the list. These pointers are indicated at the right of the figure. When data is added to the bottom of the list, for example, the bottom pointer is incremented by one for each register of data. As data is removed from the bottom of the list, the pointer is decremented by the appropriate amount. The top pointer is handled in the reverse manner.

A memory location, called the C-Register, is associated with each list and acts like an accumulator for that list. Data is generally written into the list through this C-Register. On reading a list, the data also ends up in the C-Register.

We return now to the program structure (Figure 2), and describe how manipulation procedures are developed from the SIMPLIST operations, which include:

Push L—Take the word in the C-Register for list L and add it to the top (front) of the list. This is similar to a push down operation.

Pull L—Read the word at the top of list L into the associated C-Register.

Tuck L—Take the word in the C-Register for List L and add it to the bottom of the list.

Tug L—Get the bottom most character in list L and bring it to the C-Register.

Simple operations to copy a complete list, to add part of one list to a second list, to reset the list pointers (i.e., empty the list), and to compare two lists for equivalent strings, have been programmed using the above SIMPLIST commands. These operations are then combined at the executive level into higher level operations which the scientist can use to generate algorithms and various manipulation techniques. At this level, the system can be taught procedures such as the symbolic solution of pairs of simultaneous linear equations.

Because memory space is at a premium in a medium-sized computer, it is desirable that all of the subroutines share the same work space so that each uses only as much memory as it actually requires. SIMPLIST contains a subroutine called the Automatic List Definer. This subroutine maintains its own list of available memory space and assigns it to other subprograms as needed. For example, the symbolic multiply routine requires several small lists for the temporary storage of partial results. Therefore when the symbolic multiply routine is called into action, it requests list assignments from ALD. At the conclusion of its operation, the multiply subroutine releases the temporary storage lists which are no longer needed. These lists may then be used by another subroutine.

Input-Output Format

New equations or expressions can be entered into the system through the on-line typewriter, a light pen on the scope display or by means of paper tape prepared on an off-line flexowriter. In the present version of the system the typewriter is the main communication device. However a typewriter is not a particularly convenient device for typing mathematical symbolism. The standard typewriter has only 84 distinct characters which is hardly enough for most mathematical problems. Moreover, some of these characters are required for control purposes. Finally, it is most difficult to handle exponents, subscripts and unusual characters such as integral signs. To get around this problem we assigned a special control significance to the 13 characters shown in Figure 4. The scientist types his equation as a string of symbols and intersperses these format control characters as needed. Several examples of these typed strings and their equivalent appearance on the scope display are shown in

SYMBOL	FUNCTION
□ (SPACE)	DELIMITER (except after nonspacing characters)
⇒ (TAB)	DELIMITER
↩ (CARRIAGE RETURN)	DELIMITER (end of line)
· (CENTER DOT)	DISMISS THE SYMBOL WHICH FOLLOWS FROM ITS CONTROL SIGNIFICANCE
↑	DENOTES EXPONENT
^	DENOTES SUPERSCRIPT
v	DENOTES SUBSCRIPT
| (VERTICAL BAR)	RESTORE TO MAIN LINE
?	DROP EXPONENT LEVEL ONE LINE
→	EQUATION LABEL FOLLOWS
⇐ (BACKSPACE)	IGNORE LAST CHARACTER
⊃	TREAT THE STRING JUST TYPED AS A DEFINED SYMBOL

Figure 4. Format Control Characters.

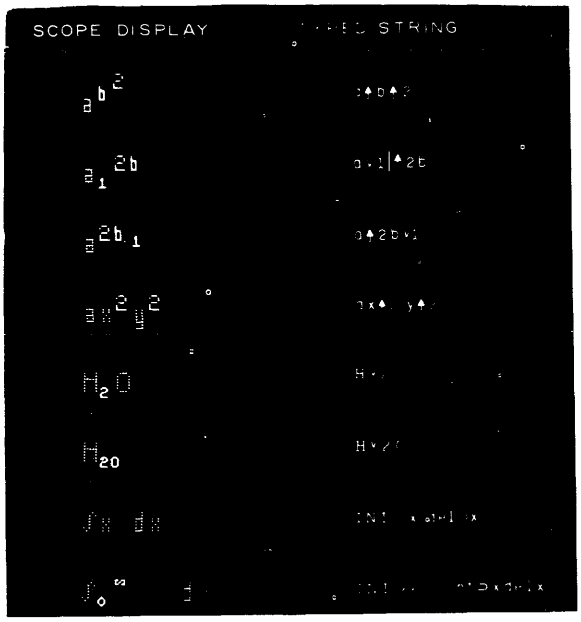


Figure 5. Example of Typed Strings.

Figure 5. When the scientist needs a symbol which is not in the normal type set, he types an abbreviation for the symbol followed by the implication sign ⊃ . The input program then looks the abbreviation up in a table and puts a special code into the string. The scope display, of course, will show the standard mathematical character. The last example in Figure 5 shows how the integral sign was *inputed.*

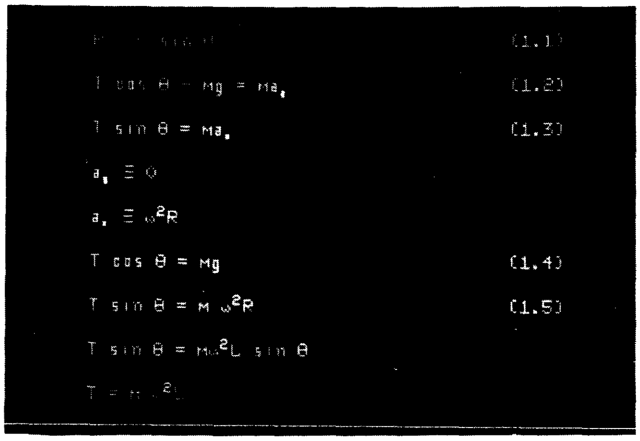
After a brief period of experimentation, the scientist can learn to type equations fairly efficiently despite the cumbersome notation. After initial equations are in the machine, the subsequent equations are usually created by the computer itself, so a very small portion of the scientist's time on the machine is spent in typing. Nevertheless, this method of getting data into the system is far from optimal.

An alternative method of getting commands and data into the system with a panel of buttons may be easier for the scientist. Culler uses this technique quite successfully in his on-line computing system. We are now experimenting with a similar button panel arrangement.

A display scope with light pen is a very convenient method for getting data into the computer. In recent years character recognition programs have been developed which permit the user to simply write his mathematical expression on the face of the tube. The character recognition program then translates the expression into the appropriate coded representation. Teitelman[9] has created a particularly sophisticated program of this type called ARGUS. During a brief learning phase, the user draws all of the characters that he expects to use in his work and identifies them. ARGUS is then switched to its normal mode and adds the symbols to a string as they are drawn by the user.

As in most man-machine systems, the data input problem is of fundamental concern. This is particularly true of on-line computing, where the user is not apt to be a programmer or even sympathetic with computers. The scientist wants to devote his full attention to the problem at hand and will only use the machine if it can conveniently help him attain that end. He does not see the internal programs directly and regardless of how good they are, judges the whole system on the basis of input and output—the weakest link in any computation system. None of the techniques just suggested are completely satisfactory, and we must therefore use some combination of two or even all three methods. But then we have a new problem—the scientist finds himself switching back and forth between the typewriter and the light pen. We still have a long way to go before we can make the information input process as easy as writing on a sheet of paper.

We will now discuss output operations. The principal on-line output device is the display



[illegible] (1.1)

$T \cos\theta - Mg = Ma_1$ (1.2)

$T \sin\theta = Ma_2$ (1.3)

$a_1 \equiv 0$

$a_2 \equiv \omega^2 R$

$T \cos\theta = Mg$ (1.4)

$T \sin\theta = M\omega^2 R$ (1.5)

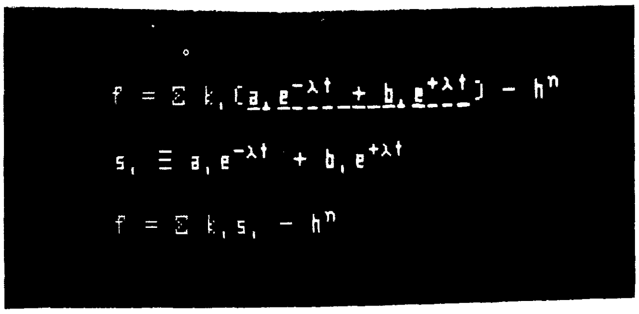
$T \sin\theta = M\omega^2 L \sin\theta$

$T = M\omega^2 L$

Figure 6. Computation Shown on Scope Display.

scope. Equations and other data are displayed directly in standard mathematical notation (see Figure 6). Special notations which may be more convenient to the user can also be introduced as desired. New equations resulting from manipulations on earlier equations are tentatively added to the bottom of the scope display. After examining this new equation, the scientist may delete it if it is not the result desired, or he can move it to another location.

Equations can also be labeled and underlined. For instance, Figure 7 shows how the scientist uses the underline feature to assign a new symbol to a portion of an equation. There are two modes for displaying equations. The normal mode displays the equation exactly as it would appear on the printed page. A second mode shows the non-printing characters (such as space, tab and carriage return) as well as the other format control characters in the equation. This mode is used primarily for editing purposes.



$f = \Sigma k_1 [\underline{a_1 e^{-\lambda t} + b_1 e^{+\lambda t}}] - h^n$

$s_1 \equiv a_1 e^{-\lambda t} + b_1 e^{+\lambda t}$

$f = \Sigma k_1 s_1 - h^n$

Figure 7. Definition and Substitution of New Symbol.

A user may save the results of his computing by photographing the oscilloscope display, by having the equations punched out onto paper tape (which can be listed off-line) or by writing the symbolic expressions onto magnetic tape so that they can readily be used in a later computer session. A photograph of the oscilloscope display is useful for recording graphs and sketches. It also produces a copy of the equations in standard mathematical rotation rather than the typewriter equivalent notation that is used during input. Punched paper tape provides a convenient storage medium if the results are to be used as a starting point for a later calculation, or if the user wishes to share his results with others who might not be using the same computer. Symbols which were defined as special characters on the oscilloscope are replaced by their typewriter equivalents, so that the user has paper tape output compatible with the input. Magnetic tape is most useful for storing and obtaining rapid access to large amounts of information. The user of the system can elect any one or all of these forms of output. He can, for example, keep his total record on magnetic tape and prepare a summary printout of his most important results on the typewriter for later desk use.

Numerical Evaluation

In the present experimental system, we have concentrated on symbol manipulation procedures and not emphasized numerical evaluation techniques. However, numbers are of great importance—even to the theoretician. He may wish to evaluate his formulas and compare them with experimental results, or he may want to plot a function simply to get some intuitive insight into its general behavior. This phase in the development of a mathematical theory is just as important as symbolic manipulation; its implementation on a computer can save the scientist a great deal of time.

The approach we have taken to this problem may be of some general interest because of its simplicity. Once the function to be evaluated is specified, the Magic Paper system modifies it for an Algol-like compiler (DECAL-BBN). A small evaluation program is produced and then operated under Magic Paper executive control. The numbers generated by the evaluation program are then displayed on a graph[10] (see Figure 8) or printed on the typewriter. The compiled program can also be run at a later time independent of the Magic Paper system itself. Thus, Magic Paper is a buffer between the scientist and the compiler for this particular application, allowing him to speak to the compiler in his own mathematical language without concerning himself with the trivial details of preparing a problem for numerical solution.

New numerical evaluation operators not included in the basic list (Figure 9) can be defined and added to the system. In normal usage, the definitions which are needed for certain types of applications would be collected on a punched tape, so that they could be entered into the computer at the beginning of a user's time on the machine.

Almost all operations of numerical analysis can be expressed in terms of the operators in the system. But infinite limits are always difficult to handle. However, if the sequence converges, an operational definition of infinity can be made through the comparison of succeeding values in the sequence. Note that infinity must be handled in a different manner than the other numbers, because testing for completion must involve comparing values of the function

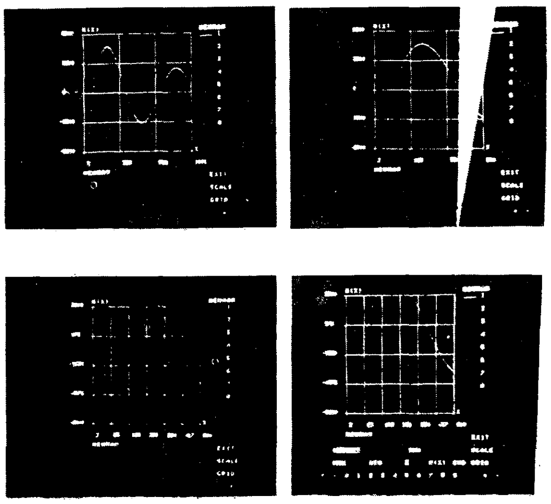


Figure 8. Graph Display.

SYMBOL	FUNCTION
+	ADDITION
-	SUBTRACTION
×	MULTPLICATION
/	DIVISION
Σ	SUM OVER AN INDEX BETWEEN LIMITS
π	PRODUCT OVER AN INDEX BETWEEN LIMITS
↑	EXPONENTIATION
( ) [ ]	PARENTHESISATION
if then < > ≤ ≥ when	CONDITIONAL EXPRESSIONS

Figure 9. Evaluation Operators.

rather than values of the argument. Thus, infinity must be treated as an operator, rather than a number.

Flexibility

System flexibility should be a major goal in any computer aid for symbolic mathematics. Each scientist who approaches the machine with a problem will have his own language and particular techniques for arriving at a solution, and the system should be adaptable enough to handle these different users with their diverse problems. Insofar as possible, we must avoid standardizing the scientist by making him conform to "the system," for surely the scientist will work far more efficiently if he can rely on his customary habits and his normal symbolism. There is, of course, a limit to the degree of freedom that can be incorporated into any system. With too much flexibility an exorbitant amount of time might be spent simply specifying what the system should do. On the other hand, a well-engineered system will have enough options so that the user has the feeling that it has been personalized to meet his individual needs.

Two types of flexibility will be discussed here. The first we shall term flexibility of notation and the second, flexibility of function. Simple matters of notation have often been key factors in the progress of science. It would have been impossible to attain the present state of our civilization if we were always dependent on Roman numerals for our scientific calculations. That notation, while adequate for expressing all formulas, was just too cumbersome to permit extensive calculations of any sort. Notational considerations are still important in contemporary mathematics. In 1916, while developing General Relativity, Einstein noticed that whenever tensors were to be summed, the summation index always appeared twice. For convenience, he introduced the now standard convention that repeated indices always imply summation without the necessity for repeatedly writing a summation sign with limits. Other physicists have also found it convenient to introduce their own non-standard notations from time to time. Similar notational innovations, while not absolutely necessary for performing the calculation, should be permitted in a mathematical computer aid. They give the scientist a freedom to communicate with the computer in precisely the same language that he would use with his other colleagues. We have already described how this notational flexibility is built into the Magic Paper system by permitting the user to define new control and data interpretation operations as he proceeds with his problem.

Functional flexibility which allows the scientist to define new operators and rules for processing symbols as he proceeds with his work is more than just a matter of convenience to the user. It is necessary for the creation of new mathematical techniques. Suppose a scientist comes to the computer for aid with a quantum mechanics problem. At the very outset, he would want to inform the machine that there will be certain pairs of symbols, such as x and p, which should be non-commutative under multiplication. Once the program is given these rules it must be clever enough to realize that $xp - px \neq 0$. Otherwise, erroneous cancellation of these terms might occur in a calculation. This example from quantum mechanics can be handled quite easily today for we know in advance that this problem is likely to come up. Likewise, we can anticipate that certain scientists will need vector analysis, matrix algebra and perhaps even symbolic logic, for all of these are well known mathematical tools. We can make allowances for their inclusion into the system. But some scientists, working in a new area, may have to develop new mathematical tools before they can solve their problem. They will want to introduce some novel rules and try out the calculation with their newly created mathematics. They may even have to go back and modify the rules several times before they attain exactly what they need.

Although there are techniques that can be incorporated into mathematical systems to handle certain types of rules that the scientist might create, there is still (to the best of our knowledge) no completely general technique to handle all rules that the scientist might want to introduce. This is an intriguing area of research on which to center attention as more advanced Magic Paper systems are developed.

REFERENCES

1. The term "Magic Paper" was first suggested by Professor Richard P. Feynman of the California Institute of Technology. The authors have adopted the name because it is an excellent characterization of the computer's role. The scientist uses the machine in much the same manner as a sheet of paper. As he thinks out various steps in the process, the paper magically responds by performing the indicated operation.
2. GLEN J. CULLER and BURTON D. FRIED, "An On-Line Computing Center for Scientific Problems," Report M19–3U3, Thompson Ramo Wooldridge, Inc., Canoga Park, California, 11 January, 1963.
3. D. C. ENGELBART, "Augmenting Human Intellect: A Conceptual Framework," Summary Report AFOSR-3223, Stanford Research Institute, Menlo Park, California, October, 1962.
4. VANNEVAR BUSH, "As We May Think," *The Atlantic Monthly,* July, 1945.
5. Personal Communication.
6. R. FANO, Report of Project MAC 1963 Summer Study at MIT in preparation.
7. The PDP–1 computer is manufactured by the Digital Equipment Corporation of Maynard, Massachusetts.
8. RICHARD MCQUILLIN, "Decal-BBN (Intermediate Operating Manual)," Report Decus T–39, Digital Equipment Users Society, Maynard, Massachusetts.
9. WARREN TEITELMAN, "ARGUS—A Handwritten-Character-Recognition Program," (paper presented at the ACM National Conference, Denver, Colorado, August, 1963).
10. Development of the BBN Graph Display was supported by the Council on Library Resources, Washington, D. C.

STOCK MAINTENANCE BY TELEPHONE—ONE STEP TOWARDS INTEGRATED MANUFACTURING CONTROL IN A MULTI-SHOP MANUFACTURING COMPLEX

George P. Lewett
Formerly of Western Electric Company, Inc.
Presently with Technical Operations, Inc., Fort Belvoir, Virginia
and
Stephen Choolfaian
Western Electric Company, Inc., Kearny, New Jersey

INTRODUCTION

The Kearny Works of the Western Electric Company employs some 14,600 people, who are involved in the manufacture of a wide variety of electrical equipment, and their electrical and mechanical components. Thirty production shops operate more or less autonomously to manufacture some $200,000,000 of product each year. These shops obtain parts, components and raw material from outside suppliers, and from other shops and Works in the Company. They sell to other shops and Works, Bell System Companies, and the Government. Stock investment in independent storerooms associated with the various shops is considerable, and its effective control is always of strong concern to management. It was recognized early that introduction of Operations Research and Computer methods would be of considerable benefit.

The manual tabulating system of stock control generally in use was developed over many years of operation and is based on product-shop concepts. Each store stocks items required by its associated shop to make its product. Where items are common to more than one shop the heaviest user becomes the stocking store. This policy gives assurance that the most qualified judgment is applied to forecasting future requirements. However, it makes system analysis difficult, since store operating policies and procedures differ significantly from one ordering and production-servicing organization to another. In addition, a wide variety of types of stock items, differing in demand patterns, procurement intervals, value, suppliers, and type of usage, are usually present in a store or group of associated stores.

This paper describes the completed first stage of a generalized Electronic Data Processing system designed for application to stock maintenance in the various Works storerooms. The installed system incorporates DATA-PHONE service for transmission of store transactions to a Data Center and processing of these transactions by a 7080 computer for stock status evaluation. Discussion centers on design of the system from two major viewpoints—

(1) as the initial stage in constructing and implementing an effective Operations Research model for inventory-production control;

(2) as one phase in the construction of a wholly integrated manufacturing control system.

OPERATIONS RESEARCH AND SYSTEMS ENGINEERING

The practical difficulties touched upon above had much to do with how an Operations Research Group was led to develop the system described in this paper. Such development is more properly characterized as manufacturing systems engineering. We want to make it clear from the outset that we are aware of this. In fact the question of how it came about is a significant aspect of this paper.

An Operations Research Group has been operating at the Headquarters location of Western Electric since 1954. However, during the past 2–3 years, groups have been organized independently at various manufacturing locations to exploit cost reduction possibilities of the new field in local in-plant operations. When the Kearny group was formed early in 1960, the objective of tangible cost reduction and the assignment of a functional responsibility for implementation of OR techniques enabled us to reassess commonly accepted concepts in the field of OR operation: (1) that OR organizations have little real need for a computer, and in fact, for most effective operation should operate independently of a computer, (2) that OR is primarily a consulting function in a strictly mathematical area. These two beliefs probably derive from the usual method of operation of a staff organization or an independent consulting firm, where OR is brought in on request from a local organization, which still bears the basic responsibility for methods improvement and cost reduction. The two concepts can be accepted to some extent as general truisms on this basis. However, they put OR as a purely local organization in a rather unenviable position. Consider first the ramifications if the first belief was accepted as our operating policy for the local OR group here at Kearny.

It was clear that the logical place to put OR to work was in the production and inventory control area. Theory was well developed, management was aware of the potentialities, and the processes concerned were of the large scale, continuous nature which could be expected to generate significant cost reduction. However, with largely manual systems operating in the decision-making areas of inventory and production control, certain significant phases of an OR project could not satisfactorily be carried out. The systems analysis and data collection required to define a problem and justify the expenditures needed for a solution in any worthwhile area was a massive project in itself. Even if the problem could be satisfactorily defined, a model constructed, a solution derived and adequately tested, a system to implement the solution with effective control would be required. The initial conclusion drawn was that organization of the OR function according to accepted practice would be too limited to be effective. A group working strictly within the defined OR function would necessarily be restricted to one-shot projects and consultant work until the Computer Methods Development organization had constructed a workable framework for initiating an OR project in inventory and production control. This type of operation was not likely to meet management objectives in any reasonable period of time.

The second belief, which would require us to concern ourselves primarily with mathematical formulation, would have been even more impractical. The major difficulty in applying OR locally lay in the definition of a problem and the implementation of a solution, not in the mathematical formulation necessary to construct a model and derive a solution. Much basic research has been done in the area of production and inventory control. The literature is replete with mathematical models and theory. The challenge was to put the theory into practice. Restricting the OR function to mathematical formulation would serve merely to increase by another inch, the mile-long gap between theory and practice.

What we did of course is clear. We ignored the second preconception, and redefined the OR project to render the first invalid. The first step of an OR project is frequently described as "define the problem." Selecting general inventory and production control as our project, we rationalized this first step to include construction of a data collection and processing system. This would enable us to analyze the current methods and policies of operation, and secondly, to collect data for model construction and test. At the same time, conversion from manual to electronic data processing would generate tangible savings in system operation for the short term. Before describing the operation of the system, some general remarks on what we consider to be its most significant feature—flexibility.

EVOLUTION VS. REVOLUTION

As indicated earlier, the Kearny Works comprises some 30 different manufacturing shops. Each operates almost autonomously, making different products, under differing conditions, with different machines and processes. The functions of supporting organizations (accounting, engineering, wage incentives, purchasing, etc.) generally overlap two or more of these different shops. The shops themselves usually are linked to each other in one way or another.

In the introduction of EDP to Kearny inventory and production control, an "evolutionary" approach has been taken. The Works manufacturing process can be described very generally as composed of three stages: (1) equipment ordering and analysis, (2) stock control and (3) shop production. Orders placed by customers for equipment, when exploded into required assemblies and components, determine the loads to be placed on each shop. Component orders placed on the Works manufacturing shops are either filled from stock or made to order by that shop. Computer Methods Development was directing its effort to completing the ordering stage, covering all shops and products before proceeding to the next phase. It should be noted that decision-making processes were not involved in this area. Computer systems were complex, but did not require incorporation of decision-making controls or criteria. The EDP system we engaged to develop similarly was designed to cover one stage, stock control, in breadth, before extending in depth to the third stage, that of production control.

This approach is in contrast to that frequently taken, which covers one shop in depth for all three stages at one time, thus revolutionizing the operation of the shop in a relatively short period. The evolutionary approach was required at Kearny because of the number and variety of shops to be covered. However, it also offered significant advantages in the cushioned impact it had on the shops and in the facility of system analysis and design. With this approach it was essential that in designing a data collection and processing system we should be able to:

- — easily adjust to operational differences between stores and shops;
- — install the system quickly in each store and with minimum change in normal operations;
- — provide for easy linkage to the ordering stage under development by the Computer System group;
- — provide for extension into the production control stage;
- — provide for expansion to the subsidiary functions of accounting, engineering, purchasing, warehousing, etc.

From an OR standpoint, it had to be borne in mind that the system we were designing was concerned primarily with the definition of existing decision-making processes and was to provide the data for model construction and test. It then had to provide facilities for control and implementation of the solution. In short, the system had to be extremely flexible.

This need for great flexibility was our primary reason for selection of DATA-PHONE service and for organization of a central data receiving area for the Works.

STOCK CONTROL BY DATA-PHONE SERVICE AND COMPUTER

The Kearny Works includes five satellite manufacturing locations in addition to the tract at Kearny, as indicated in figure 1. In all there are some thirty separate storerooms, each associated with certain manufacturing shops. Basic bookkeeping methods are substantially the same in all stores. Records are kept, manually or by Electric Accounting Machinery, of material movement in and out, replenishment orders and receivals, stock transfers balancing entries, and so on. However, the factors which must be investigated in order to effectively improve control of investment differ considerably from store to store. The new DATA-

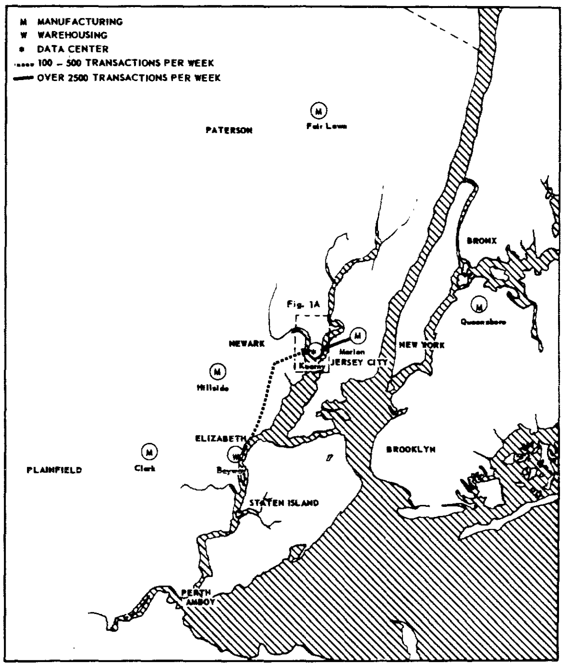


Figure 1. Kearny Works and Satellite Locations—(Showing Present Inter-Plant Network).

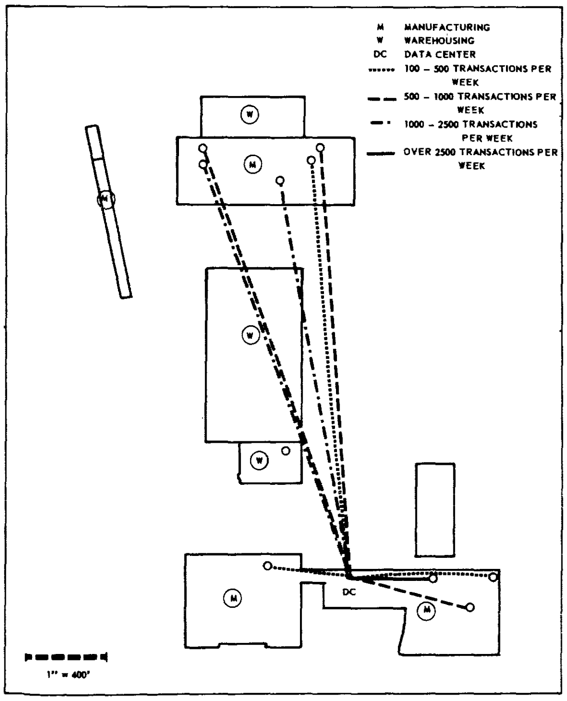

M MANUFACTURING
W WAREHOUSING
DC DATA CENTER
100 – 500 TRANSACTIONS PER WEEK
500 – 1000 TRANSACTIONS PER WEEK
1000 – 2500 TRANSACTIONS PER WEEK
OVER 2500 TRANSACTIONS PER WEEK
DC
1" = 400'

Figure 1A. Kearny Works—Main Plant Internal DATA-PHONE Service Network.

PHONE-Computer system is designed to provide more effective bookkeeping while facilitating comprehensive study of storeroom operation. Although the major benefits of the system are to be realized over a period of several years as a result of such study, considerable savings are being achieved in the initial stages simply by providing management with more effective control information, using EDP methods and some fundamental OR techniques. The system has two component phases:

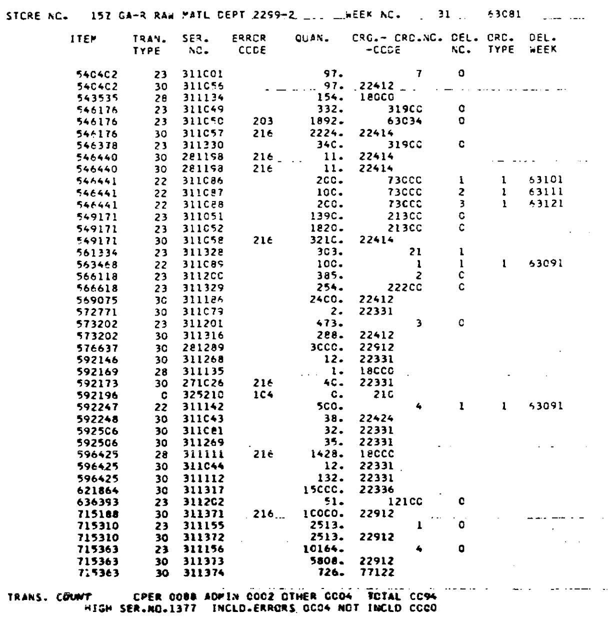
Data Collection

A Data Center is located in the main building of the Kearny tract proper, convenient to the computer. The Center currently consists of three receiving stations (DATA-PHONE receiver, translator and specially equipped keypunch), a sorter and an interpreter.

Transmitting stations (DATA-PHONE Data sets and card reader or transmitting keypunch) are installed in the ordering-stock maintenance unit of each storeroom organization in the system (figure 1A). Storeroom transactions are transmitted periodically to the Data Center in batches determined by store activity. A normal transmission consists of fixed information (item number) from a master card kept on file and variable information (quantity) keyed in by the operator. Valid transactions include all types currently used in the store recordkeeping function, plus "administrative" transactions, to be discussed in a following section.

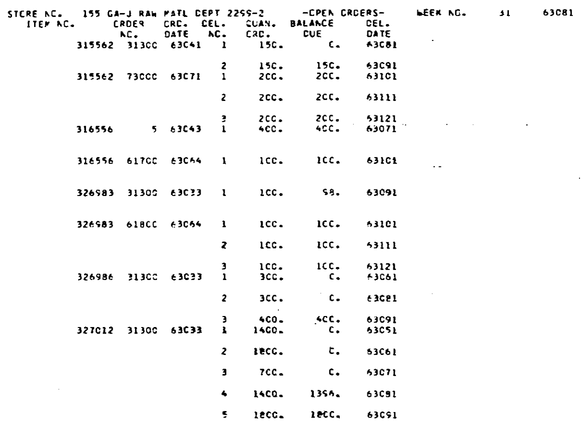
Data Processing

Transaction cards received in the Data Center are processed on the IBM 7080 Computer. Processing is usually performed on a weekly basis but may be done daily. This "control" portion of the program verifies that transactions are valid, updates a master tape and provides the storeroom with a list of all transactions received (figure 2). Where transactions are found to be invalid (for example, a receival against an order which has been previously recorded as filled), the transaction is listed with an error code for correction by the store. Once a week the updated master is run through the "Report" portion of the program, which reviews the status of each item and provides:



```
STCRE NC.   157 GA-R RAW MATL CEPT 2259-2         WEEK NC.    31     63C81

      ITEM     TRAN.  SER.    ERRCR    QUAN.    CRG.- CRC.NC. CEL.  CRC.  CEL.
               TYPE   NC.     CCCE               -CCCE         NC.   TYPE  WEEK

      54C4C2    23   311C01              97.              7     0
      54C4C2    30   311C56              97.   22412
      543535    28   311134             154.   18CC0
      546176    23   311C49             332.          319CC     0
      546176    23   311C5C    203     1892.          63C34     0
      546176    30   311C57    216     2224.   22414
      546378    23   311230             34C.          319CC     C
      546440    30   281158    216       11.   22414
      546440    30   281158    216       11.   22414
      546441    22   311C86             2CC.          73CCC     1     1   63101
      546441    22   311C87             10C.          73CCC     2     1   63111
      546441    22   311C88             2CC.          73CCC     3     1   63121
      549171    23   311051            139C.          213CC     C
      549171    23   311C52            1820.          213CC     C
      549171    30   311C58    216     321C.   22414
      561334    23   311328             3C3.             21     1
      563468    22   311C89             10C.              1     1     1   63091
      566118    23   3112CC             385.              2     C
      566618    23   311329             254.          222CC     C
      569075    3C   311126            24C0.   22412
      572771    30   311C79               2.   22331
      573202    23   311201             473.              3     C
      573202    30   311316             288.   22412
      576637    3C   281289            3CCC.   22912
      592146    30   311268              12.   22331
      592169    28   311135               1.   18CCC
      592173    30   271C26    216       4C.   22331
      592196     C   325210    1C4        C.     21C
      592247    22   311142             5C0.              4     1     1   63091
      592248    30   311C43              38.   22424
      5925C6    30   311C81              32.   22331
      592506    30   311269              35.   22331
      596425    28   311111    216     1428.   18CCC
      596425    30   311C44              12.   22331
      596425    30   311112             132.   22331
      621864    30   311317           15CCC.   22336
      636393    23   3112C2              51.          121CC     C
      715188    30   311371    216    1COC0.   22912
      715310    23   311155            2513.              1     0
      715310    30   311372            2513.   22912
      715363    23   311156           10164.              4     0
      715363    30   311373            5808.   22912
      715363    30   311374             726.   77122

TRANS. COUNT     CPER 0088 ADMIN C002 OTHER CC04   TCTAL CC94
      HIGH SER.NO.1377   INCLD.ERRORS 0C04 NOT INCLD CCC0
```

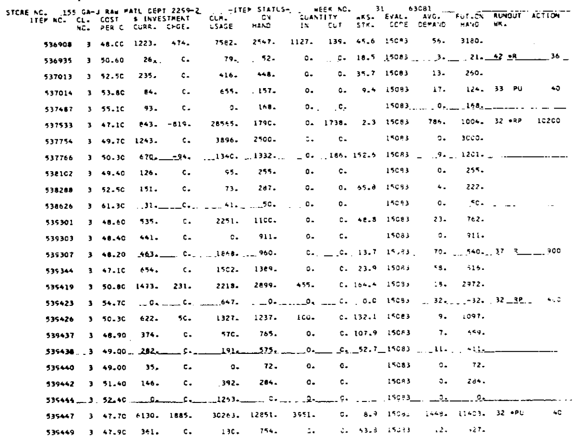
Figure 2. Transaction Listing—Showing Transactions Processed in Week.



```
STCRE NC.   155 GA-J RAW MATL CEPT 2259-2      -CPEN CRCERS-     WEEK NC.    31     63C81
  ITEM NC.       CRDER  CRC.  CEL.   CUAN.  BALANCE     CEL.
                  NC.   DATE   NC.   CRC.    CUE        DATE
     315562  313CC  63C41   1    15C.       C.      63C81

                            2    15C.     15C.      63C91
     315562  73CCC  63C71   1    2CC.     2CC.      631C1

                            2    2CC.     2CC.      63111

                            3    2CC.     2CC.      63121
     316556      5  63C43   1    4CC.     4CC.      63071


     316556  617CC  63C64   1    1CC.     1CC.      631C1


     326983  3130C  63C33   1    1CC.      58.      63091


     326983  618CC  63C64   1    1CC.     1CC.      631C1

                            2    1CC.     1CC.      63111

                            3    1CC.     1CC.      63121
     326986  313CC  63C33   1    3CC.       C.      63C61

                            2    3CC.       C.      63C81

                            3    4C0.     4CC.      63C91
     327C12  3130C  63C33   1   14C0.       C.      63C51

                            2   18CC.       C.      63C61

                            3    7CC.       C.      63C71

                            4   14C0.    1356.      63C91

                            5   18CC.    18CC.      63C91
```

Figure 3. Open Order Listing—Showing Current Status of Orders Placed on Suppliers and Feeder Shops.

(1) an *Item Status* report (figure 4)—listing stock on hand, investment, weeks of stock, cumulative usage, average demand and status during a future interval. Future status is determined by comparing the quantity (ON HAND + DUE IN — EXPECTED DEMAND) against a preset low limit over an interval of weeks determined by the storeroom.



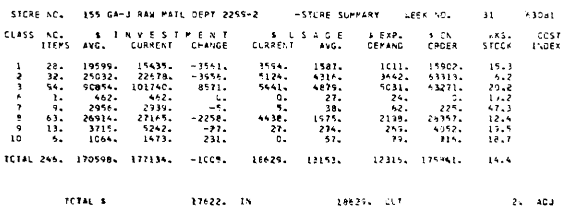
```
STCRE NC.  155 GA-J RAW MATL CEPT 2259-2     -ITEM STATUS-     WEEK NC.   31    63C81
  ITEM NC. CL.  CCST   $ INVESTMENT   CUM.    CN      QUANTITY    WKS.  EVAL.  AVG.   FUT.CN  RUNOUT  ACTION
           NC.  PER C  CURR.  CHGE.   USAGE   HAND    IN    CUT   STK.  CCCE  DEMAND  HAND    WK.

  536908   3  48.CC  1223.   474.    7582.   2547.  1127.   139.  45.6  15C83    56.   3180.
  536935   3  50.60    26.     C.      79.     52.     0.     C.  18.5  15083     3.     21.  42 *R      36
  537013   3  52.5C   235.     C.     416.    448.     0.     0.  35.7  15C83    13.    260.
  537014   3  53.8C    84.     C.     655.    157.     0.     0.   9.4  15083    17.    124.  33  PU     40
  537487   3  55.1C    93.     C.       0.    168.     0.     C.        15083     0.    168.
  537533   3  47.1C   843.  -819.   28565.   179C.     0.  1738.   2.3  15C83   786.   1004.  32 *RP  1C2C0
  537754   3  49.7C  1243.     C.    3896.   2500.     0.     C.        15083     0.   3CC0.
  537766   3  50.3C   670.   -94.    134C.   1332.     0.   186. 152.6  15C83     9.   1201.
  538102   3  49.40   126.     C.      95.    255.     0.     C.        15C93     0.    255.
  538288   3  52.5C   151.     C.      73.    287.     0.     0.  65.8  15C93     4.    222.
  538626   3  61.3C    31.     C.      41.     5C.     0.     0.        15C93     0.     5C.
  539301   3  48.60   535.     C.    2251.   11CC.     0.     C.  48.8  15C83    23.    762.
  539303   3  48.40   441.     C.       C.    911.     0.     C.        15083     0.    911.
  539307   3  48.20   463.     C.    1848.    960.     C.     C.  13.7  15C83    70.    540.  37  R      900
  539344   3  47.1C   654.     C.    15C2.   1389.     0.     C.  23.9  15083    58.    516.
  539419   3  50.8C  1473.   231.    2218.   2899.   455.     C. 164.4  15C83    18.   2972.
  539423   3  54.7C     0.     C.     647.      0.     0.     C.   0.0  15C83    32.    -32.  32  RP     4C0
  539426   3  50.3C   622.    5C.    1327.   1237.   1C0.     C. 132.1  15C83     9.   1097.
  539437   3  48.90   374.     C.     57C.    765.     0.     C. 107.9  15C83     7.    559.
  539438   3  49.00   282.     C.     191.    575.     0.     C.  52.7  15083    11.    411.
  539440   3  49.00    35.     C.       0.     72.     0.     0.        15C83     0.     72.
  539442   3  51.40   146.     C.     392.    284.     0.     C.        15C83     0.    284.
  539444   3  52.40     0.     C.    1253.      C.     0.     C.        15C83     0.      0.
  539447   3  47.7C  6130.  1885.   30263.  12851.  3951.     C.   8.9  15C83  1448.  11403.  32 *PU     4C
  539449   3  47.9C   341.     C.     13C.    754.     0.     C.  53.8  15C83    12.    427.
```

Figure 4. Item Status Report—Showing Week End Stock Condition and Suggested Action.

This interval is usually procurement time plus a safety factor. This report can be issued, as requested by the store, either on all items or on an "exception" basis (only on items with questionable status). When status is questionable either an order quantity or the pulling up of a delivery date of an existing order is recommended.

(2) an *Open-Order* report (figure 3)—listing each order for replenishment of stock, and showing order number, date order was placed, order quantity, balance left to be delivered and scheduled delivery dates. This can also be on an exception basis, tying in with the Item Status Report.

(3) a *Store Summary* (figure 5)—listing for each class of items or material and for the store in total, the average and current investment and change since previous weeks, cumulative current and



```
STCRE NC.   155 GA-J RAW MATL DEPT 2259-2      -STCRE SUMMARY    WEEK NO.    31     63C81

CLASS  NC.       $  I N V E S T M E N T         $  U S A G E      $ EXP.     $ CN     WKS.    CCST
      ITEMS    AVG.     CURRENT    CHANGE     CURRENT    AVG.     DEMAND    CRDER    STCCK   INDEX

  1     22.    19599.    15435.    -3541.      3594.    1587.     1C11.    15902.    15.3
  2     32.    25C32.    22578.    -3556.      5124.    4316.     3442.    63313.     6.2
  3     54.    9C854.   10174C.     8571.      5441.    4879.     5C31.    63271.    20.2
  6      1.      462.      462.        C.         0.      27.       24.        C.    17.2
  7      9.     2956.     2939.       -5.         5.      38.       62.      225.    47.3
  8     63.    26914.    27165.    -2258.      4438.    1575.     2139.    26357.    12.4
  9     13.     3715.     5242.      -27.        27.     234.      253.     4052.    17.5
 10      5.     1C64.     1473.      231.         C.      57.       79.      315.    18.7

TCTAL  245.   170598.   177134.    -1CC9.     18629.   13152.    12315.   175961.    14.4


        TCTAL $              17622.  IN              18629.  CUT                  2.  ADJ
```

Figure 5. Store Summary—Showing Week End Store Condition and Investment Position.

average dollar usage, weeks of stock, and dollar total on order. A "cost index," defined as the dollar cost in ordering and investment changes per $100 of annual usage, can also be printed.

SYSTEM OPERATION

A prime requirement in designing the system was ease of installation. The impact of the new system on the people actually operating the stores had to be considered. A change in mechanical methods such as the use of DATA-PHONE, and reliance on transmission and computer accuracy, was a sufficient load to absorb at one time. No attempt was made to sell or install automatic control concepts such as exponentially smoothed forecasting, mathematically derived reorder points or quantities, changes in ordering intervals or policies, and so on. Control of the system and consequently of investment had been left wholly in the hands of the store. Every effort was made to adjust the system to fit standard operating procedures, particularly where they were governed by accounting or auditing practices.

Options are provided in the system such that the store can designate:

— use of an Economic Order Quantity (Camp Formula) or a specific number of weeks stock on reordering;

— use of one of several forecasting procedures based on past usage, or the use of firm short-term advance requirements when available;

— use of "exception" or complete reporting;

— preposting of advance "selects" or actual in-out posting.

The store is able to change control criteria on any item at any time. "Administrative" transaction formats are provided so that the store may change the forecasting or ordering options (above), the safety stock level, procurement or ordering interval, or may directly adjust forecasted demand on an arbitrary basis.

The system thus provides, in effect, for both manual and automatic decision-making. The item status evaluation procedure will recommend action to be taken on an item, dependent on criteria furnished by the store. On review the store may or may not find the recommendation accurate. Criteria are adjusted until the recommended and the actual action taken correspond most of the time. At this point adjustment is discontinued and item evaluation is left on an automatic basis.

Administrative transaction codes which will transfer items between stores, cancel an item from a stock list, change item costs or descriptions, make corrections to erroneous prior transactions or correct the balance directly, are also provided. An inventory verification transaction is provided to permit periodic audits (physical count) to be entered. This procedure conforms to standard accounting control practice and automatically reports on system accuracy each month.

All transactions are transmitted to the Data Center where they are batched for processing by the computer. No editing is required in the center, which acts as a collection agency, enters new items, and changes store parameters or operating policies. The Data Center also provides transmission master cards, disseminates the computer-generated reports and maintains a central file of reports.

The previous method of operation was based on continual perusal by stock maintenance clerks of updated individual item ledger cards. The computer-generated reports replace the ledger cards and permit one clerk to handle twice as many items as before. Under the new system, although the clerk's job remains substantially the same, the supervisor is able to more effectively monitor the clerk's judgment and maintain a much firmer control on investment.

DATA TRANSMISSION

Flexibility was the main consideration in the design of the transmission system. The telephone network servicing Kearny compares in magnitude with that of a small city. Development of DATA-PHONE Data Sets by the Distribution Data Processing organization made it feasible to utilize the existing telephone system as a made-to-order data transmission network. Our end objective was Works-wide integrated manufacturing control, therefore every location and organization had to be considered as a potential future transmitting station. With DATA-PHONE service a new station could be added or an old one moved with minimum cost and effort. It should be borne in mind that our entire approach has been an evolutionary one, with self-supporting gradual change based on developments tailored for general application. The design of a separate cabling system, with the capital investment required and the design accuracy necessary to avoid costly future revisions, would have been crippling from the standpoint of time and economics.

The basic transmitting station consists of a card reader* designed and manufactured by Western Electric with a DATA-PHONE subset. In one store with extremely high activity, a transmitting IBM 026 Keypunch (Encoder), adapted by our Distribution Data Processing organization for compatibility with 401 series DATA-PHONE set, was installed.

Receiving stations consist of an IBM 026 Keypunch equipped with self-checking, with the Western Electric translator and Bell Telephone receiving equipment. A paper tape receiving terminal for increased flexibility is expected to be in trial operation in the near future.

With 14 stores operating in the system at present, activity is in the order of 10,000 to 12,000 transmissions per week. Approximately 4,800 of these are made by Encoder from the Jersey City plant, using cards generated in a prior computer system (equipment order explosion).

COMPUTER SYSTEM

The two major programs discussed above were programmed in FORTRAN for the IBM 705. During the development phase, considerable adjustment was made in machine language. The system now operates through INTERPRET 580 on the IBM 7080, with 1401 periphery support on input and output. A modified 1401 card-to-tape program provides an initial screen for invalid and rejected transmissions. The first phase of the 7080 system provides a more thorough check for improper or invalid store transactions.

Figure 6 diagrams system operation. The master balance tapes indicated contain a 3180-character record for each stock item, providing considerable unassigned space for potential future additions to the system. Much of the information on the master is maintained for research purposes. For example, exponentially-smoothed forecasts are developed on each item, although the procedure has not as yet been released for application by the stores. The output transaction tape is used for research output (in 80-column card images) as well as for discrepancies.

Main processing time on the 7080 is approximately 2 hours per week for the 10,000 items currently in the system. In addition, a supplementary transaction analysis is run on the 7080 once a quarter, summarizing and consoli-

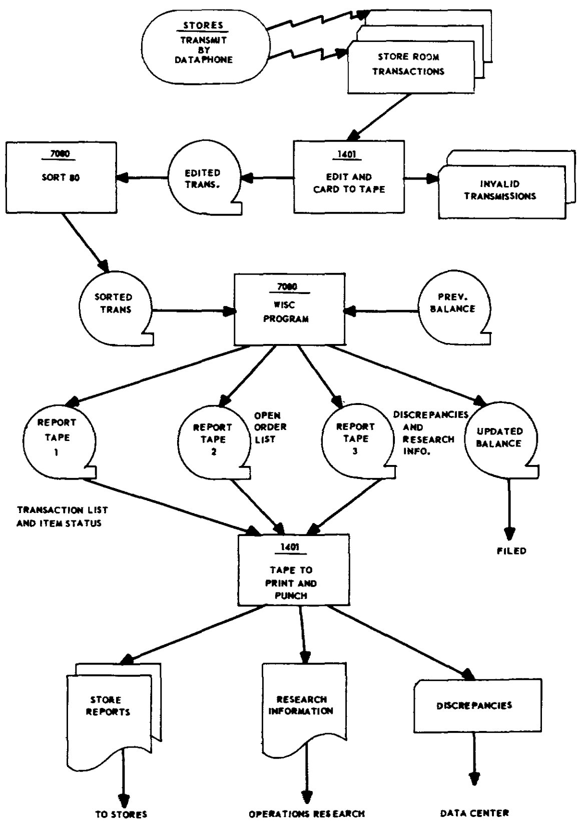


Figure 6. Flow Diagram of Stock Control Data Processing System.

*The IBM 1001 transmission terminal is similar to and based on this reader.

dating for storage all transactions by store in the preceding three months.

EXTENSIONS AND REFINEMENTS

As indicated earlier, our objective is a completely integrated manufacturing control system for the Kearny Works. The stock control system described in this paper is only a part of the work being done towards this objective. Our Computer Methods Development organization has made considerable progress toward completion of a generalized Equipment Ordering system. This system is now operating in 7 major equipment shops and has been linked to the stock control system in one of these. Additional links, such as the placing of orders on a stock store for component apparatus, parts or raw material in tape or card form compatible with stock control formats, are to be made in the near future. Improved stock control system accuracy and effectiveness, and reduced manual effort in transmission will result.

Entry into the Production Control area is keyed to output of both the Equipment Ordering and Stock Control systems. Shop output requirements (production control input) are defined by both non-stock orders generated and placed directly on the shop by the Ordering system, and by orders placed by the store for stock replenishment. Before this area is clear for full development, the Stock Control system must be refined to generate effective order quantities. In current operation, order quantities generated by the system and those finally used are still determined by store personnel, using traditional judgments and rule-of-thumb methods.

It was mentioned earlier that a basic reason for development of the system was to facilitate ordering model construction. With the system developed and operating we are now in a position to do this. The computer programs involved are designed so that we can, for example, draw off each week from the operating system, results for a sample of items. These results can be used to characterize demand patterns and determine delivery variability, determine and evaluate forecasting procedures, and establish reorder points and shortage probabilities. The model based upon this analysis can then be evaluated using the actual operating data under simulated alternative operating conditions.

Initial results on stock parts made in a feeder shop will be course reflect a sub-optimum. However, as we get further along in production control, this will be corrected. Most important is that an almost complete record of actual performance will be available, and can be used to accurately evaluate and prove in any changes proposed in store operating policies, when compared to results derived from parallel operations.

In addition to the main line for development, equipment ordering, stock and production control, attention can now be given to extending the system into subsidiary areas and to refinements in the existing system. Future developments of this type include:

— automatic generation of purchase order details and paper work. This should incorporate generation of feedback cards for receivals and the possible inclusion of major suppliers, other Works and the Purchasing organization in the DATA-PHONE service network.

— transmission of required information from the source, thus reducing or eliminating current paper work routines and procedures.

— improvement of existing hardware to eliminate the necessity for batch transmission, thus reducing card handling at the receiving end and providing a linkage to future decentralized shop production control systems.

CONCLUDING REMARKS

There are many facets to the continuing revolution in data processing which has been taking place over the past ten years. Increased speed of processing in itself caused significant changes. The flexibility afforded by the manipulative characteristics of the stored program speeded further change. But the increased use of decision-making mathematics, together with improved input-output links, is primarily responsible for the acceleration in pace now becoming evident.

EDP systems can be roughly divided into two categories: (1) systems designed to speed the flow of information or process data more economically and (2) systems to facilitate or eliminate management decision-making. Development has proceeded to date largely in applications of the first type. Early conversion of EAM applications in payroll, billing and general accounting fall wholly in this category, as do a great number of control systems. Decision-making in this category is primarily a matter of following a short series of known rules which automatically determine, and in control systems execute, a course of action. For complete systems integration of an operating enterprise it is necessary to incorporate systems of the second category. Development in this area in general is just beginning.

It is true that in process industries and military applications, progress in decision-making has been rapid. But these areas are, generally speaking, in one case functionally not complex, in the second as applications to new processes rather than changes to existing structures. Too, in a large number of businesses, information systems are sufficient for almost all needs. Significant decision-making is conducted largely at executive levels and need not be incorporated in day-to-day operating procedures. In companies of any significant size in the manufacturing industries this is not usually the case, and the distinction between the two types of systems becomes pertinent.

Development of non-decision-making systems in manufacturing has been the rule to date, due primarily to the fact that savings in this area are normally measurable beforehand and can be readily realized. However, the practical limit to applications of this type is being reached by users who have been in the field for any appreciable length of time. It must be recognized, for continued effective development, that this limit marks a turning point. A change in outlook is required as development begins in the decision-making areas of production control. Systems people will be working directly in the heart of manufacturing activity, examining decision-making processes now performed by production management and engineering. Input data to this point has frequently been in the form of punched cards prepared for previously existing EAM procedures, or was prepared in a similar manner. In production areas, much of the data required for effective decision-making must be initiated directly at the production machines or transmitted from the production floor. The background language to this point was largely that of billing, wages, taxes, bill of materials and sourcing. In production areas, the programmer-analyst will be concerned with manufacturing layouts, process characteristics and limitations, machine or tool types and functions, production and quality requirements. What all this means is that some reorientation is required to establish clearly the direction for progress towards long-term objectives. Systems have till now usually been independent and isolated from each other. Future development will result in a common interface through which these isolated areas will be linked. Input-output links must be compatible, but existing work on the subsidiary functions should not determine their basic forms. Basic production needs must override if the eventual system is to be sound. The automatic factory will be operated under a completely integrated manufacturing control system. We would hesitate to say that this degree of automation can be achieved in a manufacturing complex such as the Kearny Works. But we would be equally hesitant to say that it could not.

INFORMATION HANDLING IN AN ARMS CONTROL INSPECTION AND VERIFICATION ENVIRONMENT*

Lt. Col. L. F. Mathison
U. S. Arms Control and Disarmament Agency
Project Cloud Gap

On 18 April, 1963, the United States tabled at Geneva an outline of a treaty on general and complete disarmament. It is this document and its subsequent modifications that provide, and will provide, the bases for planning of arms control inspection and verification systems. It must be emphasized, however, that an actual system configuration will be predicated on agreements reached at the conference table and on coordinated international technical design criteria. The operational and system concept discussed in this paper, therefore, may have little, if any resemblance to an actual operating system, whose resultant design would be based on international political and technical accord. Nonetheless, it appears certain that automated information handling will play a major role in any disarmament program.

Apart from any political considerations, the functions and the technical elements of an arms control inspection and verification system would constitute a command and control environment not unlike the National Military Command System (NMCS). The functions of data collection, data absorption, identification of significant information, analysis, decision-making and feedback are basically the same for both systems. Assimilation of large masses of data is a common problem, as are requirements for complex and extensive communications systems. Real time operation may not be as critical in an arms control inspection and verification system as in the NMCS; however, the lack of ability to communicate between nations and individuals participating in arms control agreements may be a greater obstacle than in the NMCS. Whereas detailed requirements for the NMCS are at least defined, specific operational and certain technical requirements for arms control are still to be determined.

Despite the lack of specified requirements, we can, with a degree of certainty, conclude that systems for arms control should have the capability to perform the following functions:

1. Verify the agreed destruction, reduction, declared levels of arms production, force levels, facilities, and stockpiling of military materials, and the incremental changes in these levels as disarmament proceeds.
2. Detect, identify and confirm illegal production stockpiling research and development, and testing or deployment of arms.
3. Detect illegal movements or shipment of arms or forces out of or into specified zones or to other countries.

The functions of command and control, as related to the arms control problem, specifically

*The comments expressed in this paper are those of the author and do not necessarily reflect the position of the Arms Control and Disarmament Agency.

include the gathering of information in and about an area; the storage and subsequent analysis of this information for meaning and intent; identifying changes against an established data base, and finally, to interpret the significance of these indicators and their complex inter-relationships to the environment being inspected. Within this system, the information handling subsystem must accept, filter, store, collate, analyze and display a myriad of data on which command and control decisions can be based.

In examining the functions of this system, it is reasonable to conclude that it is designed to detect illegal and clandestine operations very akin to a gigantic intelligence system. However, we must not lose sight of the fact that ideally many nations will participate in the design, construction, operation, and maintenance of this system; therefore, the security aspects normally associated with intelligence activities will not apply. A national requirement by participating countries for unilateral backup will undoubtedly be necessary during the early phases of disarmament or until a level of mutual confidence is reached. The discussion of such systems is beyond the scope of this paper.

The single major and governing factor in the design of a command and control system for arms control is the degree of access which the host country will tolerate. If unlimited access could be assumed, the need for any command and control system for disarmament might be questionable. Certainly the need would be progressively reduced once the intention of participating countries to abide by the terms of the treaty was established. It is highly unlikely that any country will agree to complete access; therefore, for the purpose of this paper, it has been assumed that disarmament agreements will include a degree of access that will permit effective inspection. It has also been assumed that this access will be for the purposes of ground and airborne inspections.

Let us now examine the rudiments of an arms control inspection and verification system and its methods of operation. The concept which has been developed envisions the use of three subsystems integrated by a command and control communications complex. The subsystems include ground inspection, airborne inspection, and a data handling capability. Essentially, these subsystems will perform as follows:

Airborne Inspection

Airborne inspection data will be collected by aircraft flying at altitudes up to 40,000 feet and above, equipped with a variety of sensors. Low altitude and/or limited area coverage will be accomplished by aircraft also having a multisensor capability or helicopters with selected sensors and observers.

Ground Inspection

Three types of ground inspection teams will be required: (1) teams for inspection of production and deployment of armaments; (2) teams for inspection of transport facilities, such as rail centers, highway terminals, airfields, ports and harbors; and (3) teams for investigation of clandestine activities, spot checking, etc. Inspector capabilities may be augmented with radar, photographic equipment, and infrared sensors, magnetometers, spectrophotometers, Geiger and Beta counters, chemical, physical and biological instruments and communications monitoring devices.

Data Handling

With expected acquisition of data from a combination of ground and airborne inspection teams, there is the complex task of transforming these data into information of a kind that can be used as a basis for effecting the functions of inspection and verification. This is an information handling and analysis problem which calls for a proper mix of machine processing techniques and human capabilities in a system that can perform the prescribed functions efficiently and at a reasonable cost.

Data collected by aerial and ground inspection teams must be accepted, translated if necessary, reduced and processed at the data handling center. Both image and non-image data are received from each mission necessitating reduction, interpretation, classification and insertion into the appropriate files. Data from each mission are correlated and compared with data already stored and confidence factors established. In this manner, files would be continuously updated, enlarged, and confidence factors revised.

To fully appreciate the magnitude of the data handling problem, an examination in a greater detail of the system components would be helpful. Let us first examine the operational aspects, beginning with the sensors and ending with the conversion of inspection data into machine language. This is shown in Figure 1.

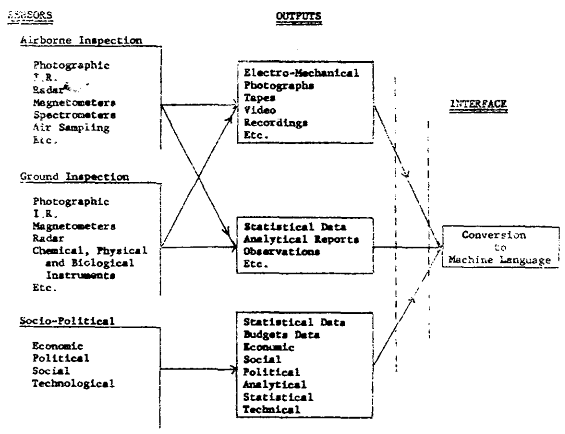


Figure 1.

The sensors listed in Figure 1, under airborne and ground inspection, are only a few of the technical equipments which will possibly be used in an actual arms control situation. If we assume an inspection area the size of the U.S. or the U.S.S.R., the amount of data which these devices can collect, even with limited access, defies processing and assimilation without electronic data processing assistance. Add to this the socio-political data which are available in tremendous quantities and the task of data handling becomes almost insurmountable. One might ask why this latter data is used if it complicates an already complicated situation.

Two reasons should help to answer this question. The first reason is that a vast amount of information has been accumulated over the years and is available for the asking from sources such as the Department of Commerce. The second reason is that electronic data processing provides a technique for analyzing the data and presenting it in a meaningful fashion for arms control and disarmament use. These socio-political considerations such as economic, political, social and statistical data, when interpreted in an arms control and disarmament context, could provide satisfactory substitute methods for physical inspection of a country. It is our plan to determine the validity of this assumption.

We have in our system concept reached the sensor output stage where we have very large quantities of data in many forms. Some of this data is readily converted into machine language; the majority, however, cannot be readily converted. In the case of some ground and airborne sensors, it must first be processed into raw workable data, and the pertinent or desired information extracted for further man-machine processing. Other sensor outputs could be converted to digital form and sent directly to the data processor. In the case of statistical data or reports, a major problem of cataloging and filing is encountered. All of these problems can be summed up in one word, "interface."

Precisely what are the interface problems in a system such as we are considering? Take photographic techniques as an example. The use of light sensitive film as a sensing means will perhaps provide one of the most meaningful inputs to an arms control inspection and verification system. Yet one of the major interface problems with which we are confronted is in this area. How is the film read and analyzed and the data fed into the data handling subsystem? Do we use a man as the interface or can we automate the operation to give us a real or near real-time approach? Side-looking radar presents a similar problem, as do other sensing techniques we have mentioned.

These problems are being studied by the Arms Control and Disarmament Agency. The difficulties are recognized. Continuing research should, however, provide solutions to these problems. They have been a problem area for many years in military systems for which large sums have been expended in searching for a solution. We feel our problem in this area is especially important because we would be hard put to recommend an automated operational

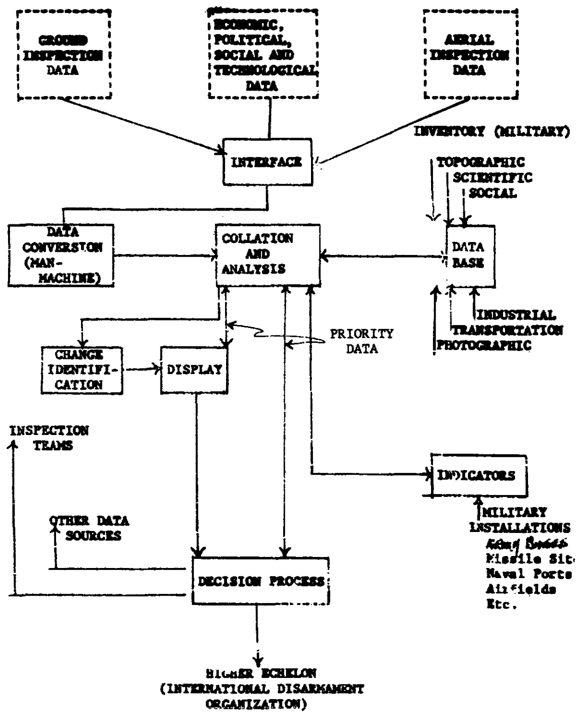


Figure 2.

system should such a system be required in the near future.

Figure 2 is a simplified diagram of a concept for an Arms Control Inspection and Verification System including the command and control function. We have indicated in this diagram the interface and data conversion as separate functions. In an actual system they could or possibly would be combined. For simplicity, many important data processing steps have actually been eliminated from the diagram. These include storage of data before collation and analysis, updating of the data base and indicator files, the means for coordination of activities between the major elements of the system and the current operational file which would be associated with the collation and analysis process.

With the exception of the data base and indicator files, the remainder of the system operates in a dynamic environment changing constantly as data is updated and corrected. The data base and indicator files are changed only after a thorough analysis and study of data to be inserted. The data base consists of information collected over prolonged periods of time and is filed in such a manner that new data collected by the sensors can be compared with information already in the file. Examples of data in the data base might be steel production figures over the past 50 or so years, salient features of the trans-Siberian Railroad and scientific and technical data extracted from technical papers, publications, etc. The data base is also updated with information from the sensor system as these data reach a certain confidence level. There is a continuing updating and inputting to maximize the confidence level.

The indicator file is an up-to-date consolidation of information required for the identification of the various types of military bases and industrial complexes. It differs from the data base in that the information it contains is general in nature, whereas the data contained in the data base concerns a specific site, area or activity. In other words, the indicator file contains the salient descriptive features necessary to identify a missile site, nuclear power plant, Army base or similar facilities.

After the data obtained from the sensor has been converted to machine language, the process of collation and analysis begins. First, all of the data obtained from the sensors in and about a specified area is processed and any priority-type data made available to the decision makers. The next step is to compare current sensor data with information contained in the data base or indicator files. In this process, an attempt is made to determine if a change in a normal or agreed to situation can be detected. Using previous examples, steel production shows an increase of 13 percent, a new railway spur is being constructed on the trans-Siberian Railroad, or a large number of nuclear physicists are being transferred to a new facility being constructed in central Russia. These changes may not in themselves be violations of any treaty or agreement, but they could indicate a clandestine activity in operation or in the early phases of a buildup. It should be emphasized at this point that the purpose of the information handling system is to detect change and not to categorically state that a violation has been committed. That a violation has occurred is determined by the decision makers who could be inherent within the system, or the facts surrounding a changing situation might be sent to an International Disarmament Organization where a decision would be made.

When a change has been identified, the pertinent facts will be appropriately displayed to facilitate the decision-making process. For example, we might display all of the steel production figures for fifty years. These could show a gradual increase during normal times with peaks for World War I, World War II, and Korea. However, now we note a sharp increase in production with no readily available explanation. Our industrial data do not indicate any vast expansions and no major housing or other commercial projects are being undertaken. Display of this data can be made so that a decision concerning the criticality of the situation can be reached. The decision makers may ask for a reassessment of the facts, send out inspection teams in an attempt to verify the facts directly, ask the inspectee country for an explanation or perhaps verify or discredit the information through the unilateral information sources available to members of the IDO.

Other more complicated situations could be cited such as a combination of the construction of a railroad spur to a remote area, increased steel production, development of large housing communities in the vicinity of the railroad spur, and movement of scientists and engineers to the area. This data alone indicates a major change from a normal situation, although not necessarily establishing the fact that a violation has been committed or construction of a clandestine activity is underway. It is interesting to note, however, that the data could have been obtained from a variety of sensors and only through collation and analysis was it possible to determine that a significant "change" in a geographical area was taking place.

While the foregoing are relatively simple examples, there could have been hundreds of other bits of information made available through the system which could serve to describe the activity in relatively minute detail. It should be emphasized that if this system were manual in character, only limited data could be used with consequently reduced confidence placed in the conclusions; however, with electronic data processing the source of data used can be extended many times and thereby add credence to the apparent discovery of a violation. Therefore, the confidence factors assigned to a chance detection is dependent to a large extent on the amount and numbers of sources of information which are used in the collation and analysis function.

It is recognized that the above is a very generalized treatment of the data handling and command and control functions of an arms control inspection and verification system. The approach to an ultimate system, aside from political factors, must evolve from comprehensive studies of the problem and experience gained through the conduct of actual field tests and experiments. It is the field testing area with which Project CLOUD GAP is most concerned. Research, concept development and studies will not attain the confidence level that a well designed, executed and evaluated field test can provide.

It is emphasized that, for the purpose of arms control and disarmament, a system such as that described does not exist nor has its practicability been fully investigated. Consequently, the scope of this paper has been purposely confined to a limited treatment of both requirements and operational objectives. We are, however, certain that access, the requirement for real or near-real time operation, sensor capabilities and interface problems all will have a major effect on the systems design.

AN APPROACH TO MANUFACTURING CONTROL USING INEXPENSIVE SOURCE TO COMPUTER COMMUNICATION

Claude A. R. Kagan and Ronald Tevonian
Western Electric Company, Inc.
Princeton, N. J.

SUMMARY

More efficient use of computers to reduce costs as a result of improved scheduling and control of manufacturing operations is made possible through the availability of timely, accurate and complete information.

This requires more effective data gathering from the lowest and most dispersed level of operations. Such extensive coverage is made economically feasible through the application of a concept of tailor made systems which provides versatility through the use of a variety of modular subassemblies. Low cost is possible through extensive use of components and devices mass produced for other purposes. Some aspects of two systems which employ these concepts are described.

BACKGROUND

An understanding of the reasons which underlie the development program described in the pages that follow forms an important background to the design philosophy being pursued.

The principal business of Western Electric Company is manufacturing telephone apparatus, cable, switchboards, etc., chiefly for the A.T.&T. Company and its telephone subsidiaries, procuring and selling to such companies materials and supplies not of its own manufacture, and installing central office equipment for such companies.

Company operations are spread across the country in 12 factories and 35 Distributing Houses which in total employ approximately 100,000 people. In general, goods of a particular type are produced at only one location. There is, however, extensive flow of material between factories in the course of building up product from basic raw materials to its final form.

From the diversity of products being made and the number of making shops that depend for their input on the output of others, it becomes evident that the Company is at grips with a very substantial control problem. This problem has its roots on the manufacturing shop floor.

The traditional way to avoid missing ship schedules is to provide adequate in-process stocks of materials to keep the pipeline full. In this manner, it is hoped that disturbances due to unpredictable factors will be smoothed out. Low annual demand for a proportion of the product line makes such a strategy costly.

The degree of effectiveness in controlling (sic: provide guidance and instructions) the organization of the Company from the lowest working to the highest policy making level depends on the ability to deliver accurate and timely information that is sufficient *and only sufficient* to the people *and only those people* that need it. Potential reductions in the cost of producing goods is the basic incentive underlying the search for improved effectiveness in manufacturing control.

INTRODUCTION

The variable characteristics of the data which have a bearing on the effectiveness of a control system are:

1. Timeliness.
2. Accuracy.
3. Sufficiency.

The system parameters which have a bearing on the above characteristics are:

A. Closeness of originator of data to point of data entry.
B. Sophistication of communication system.
C. Degree of computer involvement.

Timeliness

Timeliness of the data is beyond the ability of the originator or his supervisor to adjudge. In general, data must be timely enough to be of value and only in retrospect can the need for prompt transmittal of one item of data be evaluated. The timeliness factor is enhanced by bringing the point of data entry as close to the originator of the data as possible. Timeliness is also enhanced by providing increased sophistication in the data communication system. Finally, the timeliness characteristic is most appropriately judged within the computer in competition with all other items of entered data that form the overall problem parameters.

Accuracy

Accuracy is largely controlled by the elimination of manual data handling between the originator and the data processor. Hence, it would seem that the accuracy might be achieved by requiring the originator to introduce his data himself.

Accuracy is enhanced by an increase in data communication sophistication coupled with judicious use of controlled formats, timing devices and prompt error control feedback.

Greater accuracy is yet possible with the application of computers to cross check separate reports which can be cross correlated and errors can thus in fact be corrected before damage to the control model can occur.

Sufficiency

Sufficiency applies both to the amount of data entered and the amount of control (guidance or instructions) disseminated.

The amount of data to be introduced *by a person* decreases as the point of entry gets closer to the source. The use of pseudo-codes and automatic data generation contributes to the reduction of the amount of data to be actually entered. (By-product effect is to increase accuracy and reduce delay).

Sufficiency is primarily enhanced by application of computer technology. Highly coded and abbreviated input data can be readily transformed into full blown expanded form by computers in the course of analysis and processing. Computers can then, on the basis of fundamental information, produce the whole range of reports and directives tailored in each and every case to the specific needs and requirements of the recipient.

Sophistication of the communication system in this case permits the dissemination at more frequent intervals of more abbreviated documents. This has the effect of relieving the recipients of a great deal of analytical effort and handling of superfluous material.

THE CHALLENGE

Current commercially offered designs, in general, contain capabilities far in excess of those required for the systems, comprised of large numbers of data entry devices, deemed desirable for reasons discussed.

The cost of purchasing equipment to satisfy each of four different applications with the equipment offered for sale by six different manufacturers are shown in Figure (1). We find that we can produce systems capable of fulfilling our needs at costs substantially lower than those shown in Figure (1).

This cost reduction is made possible, in part, through extensive use of mass produced ele-

Features Provided	I	II	III	IV
Number of Stations	24	9	49	15
Message Length	Variable	Fixed	Controlled	Controlled
Card Input	Recirculating	Recirculating	No	Recirculating
Clock	5 minute	No	3 minute	1 minute
Display	1 - Numeric 6 - Lamps	Preset Switches	No	4-12 Message at three stations
Signal at Station	Bell Flashing Lamp	Lamp	Lamp	Lamp
Supervisory Keys	6	3	No	No
Error Feedback " "	Use Numeric Display	No	Timeout	Timeout Error Lamp
Output Paper Tape Printer	20 ch/sec Off line Model 33	60 ch/sec No	20 ch/sec Model 29	20 ch/sec Model 33
Associated Computer	1620 (IBM)	Monroe XI	1410 (IBM)	1410 (IBM)
Cost to Provide in $1000				
Manufacturer A	36	16	71	25
B	52	20	102	34
C	51	21	101	33
D	71	29	82	41
E	51	29	92	37
F	79	36	143	53

Figure 1. Comparative Costs of Implementing Certain Systems.

ments made usually for the communications industry. We, of course, because of familiarity and ready availability, prefer to employ Western Electric products. Similar elements to the ones we use are available from a number of other sources thus permitting our concepts to be implemented by others.

THE DEVELOPMENT

Station Requirements

We have imposed on ourselves these requirements:

1. Originator must have immediate access to a station without having to travel (if a bench hand, device must be at arm's reach)—This does not necessarily imply immediate access to the system for purposes of entering data; in fact, this arrangement makes possible the use of slower access and transmission rates.
2. Station must be the smallest possible capable of providing necessary features for its location and application (bench space at premium).
3. Station must be that which is least expensive yet capable of providing necessary features for the required location and application.
4. Volume of variable data ot be entered by an individual must be kept to a minimum.

FEATURES

System Features

Maximum Use of Pseudo-codes

The ability of computers to accomplish such functions as table look-up allows the use of pseudo-codes. Fewer digits may be used for complete identifications of jobs, materials, machines or people.

Maximum Use of Automatic Information Generation

Identity of originating station carries with it information to aid in interpreting data such as identity of originator and location; hence nature of reportable jobs or work.

Provisions to Enhance Accuracy and Reliability of Information Entry and Flow

Controlled format, feedback to data originators, use of card readers to maximum extent possible for fixed types of data and simple procedures to be followed.

Station Modules, One or More of Which Are Required for Entry of Data

Variable Information Input Module

1. Rotary dial
2. Ten Key push button dial
3. Multiposition switch units

Fixed Information Input Module

1. Card reader to handle special design cards
2. Card reader for handling conventional tabulating cards
3. Memory devices (automatic diallers)

Means to Assist or Direct Users . . .

1. In requesting service.
2. That service is available.
3. In using set correctly and entering data accurately.
4. Through an information delivery module of minimum capability to serve their needs.

The Simplest Data Collection Station:

Entry—A rotary telephone dial provides limited but effective means to:

1. Request service
2. Enter transaction codes
3. Enter fixed data
4. Enter variable data

Advantages are:

1. Minimum operator training
2. Proven reliability
3. Low cost

Disadvantages are:

1. Slow in operation
2. Digit by digit operation
3. Range limited to about ten miles unless audio modulation is employed

Visual Signalling—A signal lamp provides indication to user of:

1. Service availability
2. Proper functioning of system
3. Correction of operating procedures

Advantages are:

1. Low cost
2. Creates no disturbance to others

Disadvantages are:

1. Only limited meaning can be applied

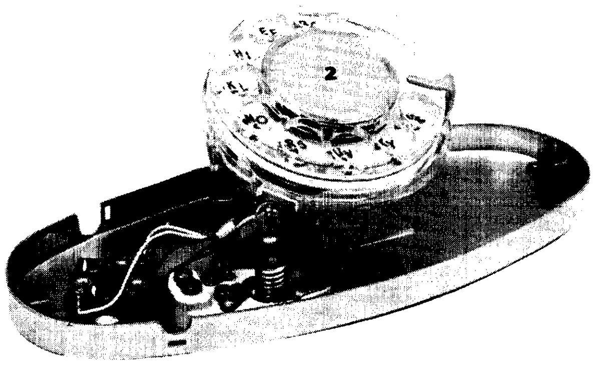


Figure 2. The Simplest Data Collection Station.

Packaging

Assembling a rotary dial and a signal lamp in the least expensive housing available provides a complete data collection station that can be petite, graceful, familiar to factory workers at the lowest level of production—and it lights up.

Comments

It is evident that this type of station lacks all sophistication within itself.

Features to provide error control are incorporated in the control center as required and described further. Effective interpretation to permit use of the data introduced requires the backup of a digital computer.

Add-on Modules

A broader range of station features is possible through the use of larger housings and the incorporation in the station circuitry of several additional modules.

A multiple key assembly can be plugged in the station set. Each of the keys can be set for locking or momentary contact mode of operation. The purpose or function of each key is marked on the adjacent designation strip. Suitable wiring at the control center fulfills the control or data reporting functions called for by each particular key as a function of the specific station identity.

Auxiliary lamps under each designation strip position are available for supervisory signalling to the station user.

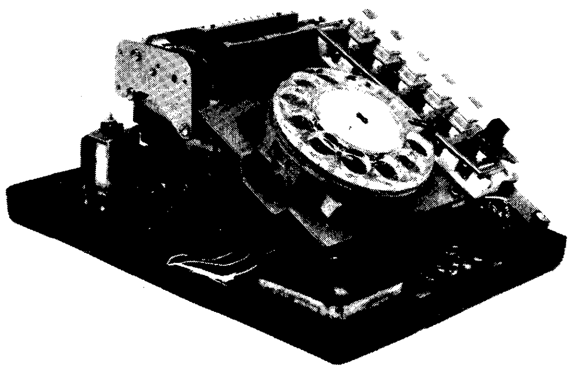


Figure 3. Some Add-on Modules.

A card reader compatible with the rotary dial can be installed in the station set. This particular reader, intended for automatic telephone number dialing, accepts a plastic card which has a capacity of fourteen decimal numbers.

The practice of recirculating cards through the system made possible by pseudo-code assignment enhances the economy of the hardware. Security and reliability is provided as required through use of self-check number arrangements.

Numeric Display—In certain cases the use of a digit-at-a-time display is indicated. This display provides the station user with some check of the accuracy of data entry. In addition, this numeric display can be used for centrally originated dissemination of control information to the station user. Work assignment is an important potential use of this single digit display.

Message Display—More sophisticated displays such as a twelve-message unit are used in stations where data entry sequence is more complex or in special stations used during training periods.

Voice Feedback—The least expensive technique of feedback to data originator is through the use of voice to an earphone or loudspeaker at the station. Voice feedback of entered data on a digit or field at a time basis from the control center is in many cases quite appropriate. With commercially available digital-to-spoken converters this feature is provided economically.

Audio Frequency Signalling Modules

Ten Key Dial With 2/8 Oscillator

The use of audio frequency signalling provides a number of additional features. The operation of one key on a Ten Key dial and oscillator module delivers an audio output consisting of two tones out of eight.

The advantages of this unit are:

1. Faster operation possible
2. Unlimited range of transmission over conventional voice grade circuits
3. Provides potential for increased station versatility

Disadvantages are:

1. Some practice required to use efficiently
2. Cost higher than rotary dial
3. Control center requires additional electronic equipment to receive and decode signals generated at station

Multiple Key Assembly

The 2/8 code provides for sixteen discrete combinations. Some of these may be generated by suitable connection of the multiple Key Assembly to the oscillator subassembly of the dial module. In this manner, special controls and supervisory functions can be added to the system with no change to the cabling.

Card Readers

A card reader similar to that used in DC pulsing systems can be added. This card

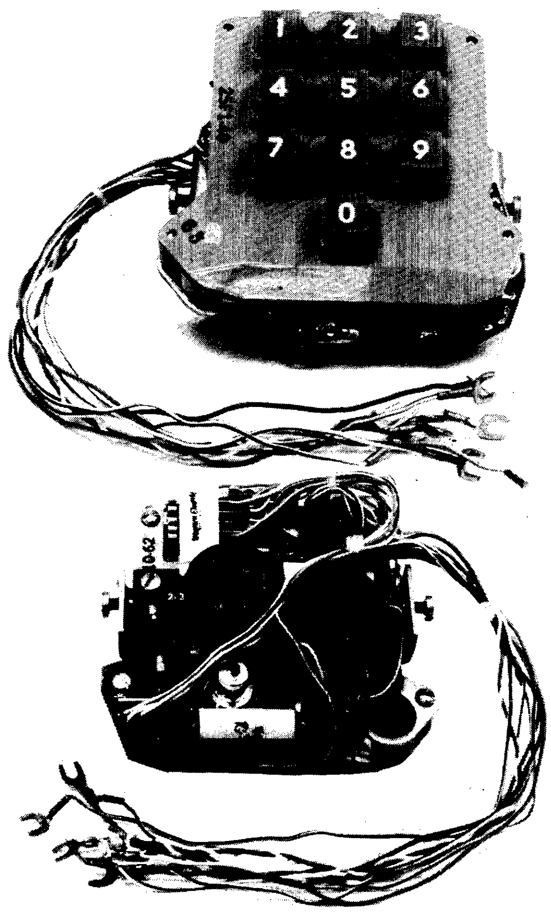


Figure 4. Ten Key Dial with Two Out of Eight Oscillator.

reader connects to the oscillator subassembly and causes the generation of 2/8 coded output. It should be noted that the coding and the card used in both types of card readers is identical, hence mixed systems using the same cards are quite feasible.

A more sophisticated card reader capable of handling tabulating cards can also be readily connected to the oscillator subassembly.

Special Stations
Clock

The entry of time in the flow stream of input data adds a frequently needed parameter. A clock-controlled digital code generator is connected to the system on the same basis as any data input station. The clock station circuit is set to request service at predetermined absolute times. Access to the system is granted in queue with regular stations and true time (the actual time of access) is logged. The difference between logged time and predetermined "service request time" provides a measure of service delay. This allows the actual time of service requests by station users to be determined statistically. Special circuitry prevents successive clock entries if no operator entries have been made in the interim period.

WIRED APPARATUS SHOP

Shop Character

The "Wired Apparatus Shop" is typical of many production organizations both in Western Electric as well as in other companies. This particular shop was chosen to test hardware concepts and techniques for effective utilization of management information.

The physical arrangement which is shown as Figure (5) employs 49 people in each of two full shifts of operation. The group consists of three sections. One of these is concerned with the production of one class of telephone switching frames. The second section is responsible for the production of another class of similar frames. The third section is responsible for inspecting and testing the output of the first two sections. The organization chart, of this shop and of its management superstructure, is shown in Figure (6).

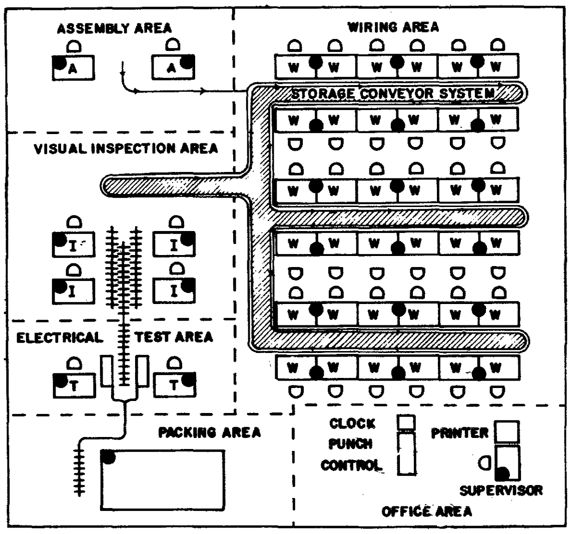


Figure 5. Physical Arrangement of the Wired Apparatus Shop.

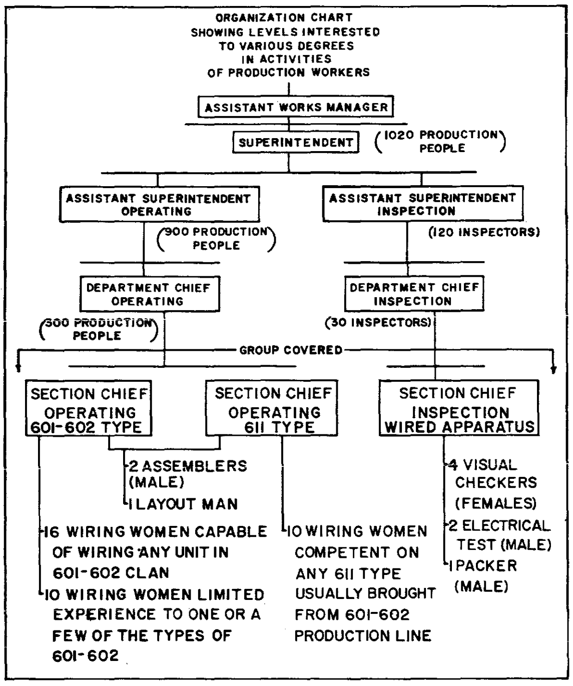

ORGANIZATION CHART
SHOWING LEVELS INTERESTED
TO VARIOUS DEGREES
IN ACTIVITIES
OF PRODUCTION WORKERS
ASSISTANT WORKS MANAGER
SUPERINTENDENT
(1020 PRODUCTION PEOPLE)
ASSISTANT SUPERINTENDENT OPERATING
(900 PRODUCTION PEOPLE)
ASSISTANT SUPERINTENDENT INSPECTION
(120 INSPECTORS)
DEPARTMENT CHIEF OPERATING
(300 PRODUCTION PEOPLE)
DEPARTMENT CHIEF INSPECTION
(30 INSPECTORS)
GROUP COVERED
SECTION CHIEF OPERATING 601-602 TYPE
SECTION CHIEF OPERATING 611 TYPE
SECTION CHIEF INSPECTION WIRED APPARATUS
2 ASSEMBLERS (MALE)
1 LAYOUT MAN
16 WIRING WOMEN CAPABLE OF WIRING ANY UNIT IN 601-602 CLAN
10 WIRING WOMEN LIMITED EXPERIENCE TO ONE OR A FEW OF THE TYPES OF 601-602
10 WIRING WOMEN COMPETENT ON ANY 611 TYPE USUALLY BROUGHT FROM 601-602 PRODUCTION LINE
4 VISUAL CHECKERS (FEMALES)
2 ELECTRICAL TEST (MALE)
1 PACKER (MALE)

Figure 6. Organization of Wired Apparatus Shop.

The total weekly output per shift is approximately 1400 frames. There are in all 40 varieties of frames that can be produced. The required quantity may range from 700 for one type to none for others. Typically, the 601/602 class as a whole (17 types of frames) is produced in weekly one-shift quantities of the order of 1000 frames. The 611 class (21 types) is produced at a rate of 400 frames. Expected time required to complete each of the various operations on each type of frame is established and serves as the basis for computing efficiency and determining group incentive bonus pay. Typical wiring times for several of the types of frames are shown in Figure (7).

CLASS OF FRAME	NUMBER OF TYPES	TIME TO WIRE MINUTES
601-602	4	28
	4	34
	4	42
	5	38
611-	21	32
		42
		48
		88

Figure 7. Typical Wiring Times for Certain Frames.

The production process consists of:

(A) Assembling subassemblies to frames.

(W) Wiring between these subassemblies and cable terminals as prescribed for each type of unit.

(I) Inspecting visually for a variety of defects.

(T) Electrically testing the entire wired assembly.

(P) Packing for shipment.

Other shops outside the group in question provide the variety of materials required to produce the frames. The materials required include wired subassemblies, factory made cables, power units, framework parts, fastening devices, precut and skinned wired and miscellaneous supplies. In all, there are some 1000 different items required. Requirements for each item vary with the mix of types of frames the group is ordered to produce. Fortunately, the maximum variation is generally felt on low cost items.

The value of the product shipped annually is approximately $20 million. Each shift contributes approximately $1 million of direct effort including its portion of overhead costs. The cost of carrying in-process inventories or safety stocks to forestall unpredictable occurrences can be quite high. Slight decreases in efficiency tend to increase this inventory or cause delays in shipping. These delays are costly both to the group in terms of increased inventory and to the customer who frequently has established an installation schedule with committed materials from other shops, and may have manpower waiting on site.

The famed maxim "For the want of a nail the campaign was lost", provides an insight to the reasons that lead to a desire for complete and timely knowledge of elemental facts. Many of the reasons for decreases in efficiency and delayed shipments are found to have assignable causes. These causes can be corrected in most cases with early knowledge of negative and disturbing occurrences. The absence of the only worker skilled in wiring a particular frame is a disturbing but obvious occurrence which can be adjusted by rescheduling.

More elusive, however, is a general but slight decrease of efficiency among many workers.

Frequently, this is detected weeks after the goods were shipped, probably on schedule, but at the cost of bad temper and ulcer producing pressures applied at short notice. Detection of a negative efficiency trend on an operator to operator and frame to frame basis with source data collection on the same basis, alerts the supervisor to seek the trouble. Recognition of the problems followed by prompt corrective action can lead to quicker control of inventory investment and schedules.

A common reason for decreased efficiency results from defects found by the inspectors and testers. Analysis of these on an operator, defect, and type of frame basis provides the supervisor with a daily guide as to where he should expend true supervisory effort in the sense of assistance and training of his group.

Work assignment by type of frame to individual operators can be performed to meet the objectives of the group. At periods of heavy order, load frames are assigned to operators to take advantage of their experience and state of training. During intervals of light load or when ahead of the game, deliberate assignment can be made for training purposes thereby upgrading the potential effectiveness of the group. This requires detailed timely knowledge of operator activity only practical with point of source data collection.

There are numerous other areas which can be developed to improve overall shop effectiveness and minimize the effect of disturbing events through the availability of basic and timely information as to:

WHO	is the operator
WHAT	happened (start of work, found defects)
WHERE	is the report made from
WHEN	was work on the unit started

Hardware Organization
Data Originating Station

The entry device to the system consists of a simple housing to mount a rotary dial, DC pulsing card reader, six button key assembly, bell, and a miniature "Nixie" numerical display tube for visual display of data being entered.

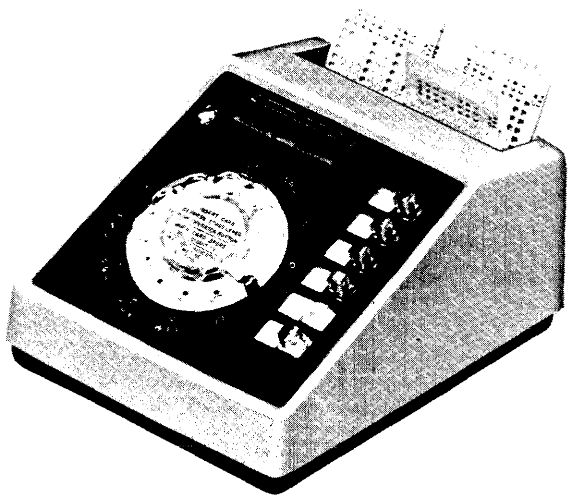


Figure 8. Data Originating Station.

The row of push keys for this set are labeled in this application from top to bottom: "release digit", "cancel entry", "Spare", "right operator", "left operator", and "end entry".

The user can introduce information in the system by inserting a plastic perforated card in the reader slot. As the card rises, its coded content is read into the system. This can be such data as job number, part number, special instructions or user identification.

Next, if required, the user can dial quantity or other appropriate modifiers. Each dialed digit appears on the visual display tube to provide assurance that the entry has been recorded at the control center. The push keys are used for special functions such as terminating the transmission, canceling a dialed digit or a whole message, and special meanings assigned to each station. Predetermined codes are logged whenever one of these keys are operated.

A queuing feature of the system assures the users that they will be given access in the order of request. In this system, we have true point-of-source data collection eliminating the need for an operator to leave the work position to enter data.

Time Generating Station

Connected to the control center on the same basis as any operator station is a digital time generator of the type described earlier.

Selective Feedback of Quality Information to Bench Workers

Engineers functional for the operation of the shop added to the basic data collection system an interesting feature which does not interfere with normal data collection operations. Transmissions made by inspectors are monitored by a pulsing relay inserted in series with the input to the control center counter module. This relay operates switching equipment to recognize the entry of four of some twenty types of inspection items. This equipment then directs signals to the bench position of the operator responsible for these four types of defects where for each a daily cumulative count of occurrences is displayed.

Information Flow and Usage

The layout of the shop shows a total of 28 data collection stations located in such a manner that it is not necessary to leave the work position to enter a report.

Although all the stations are substantially identical in appearance and mode of operation they provide a wide range of reporting functions. The identification of each station is automatically entered at the end of each transaction. Operator assignment to particular positions are reported only whenever a change is made. In this manner, the WHO and WHERE and some indication of WHAT is provided with no user effort.

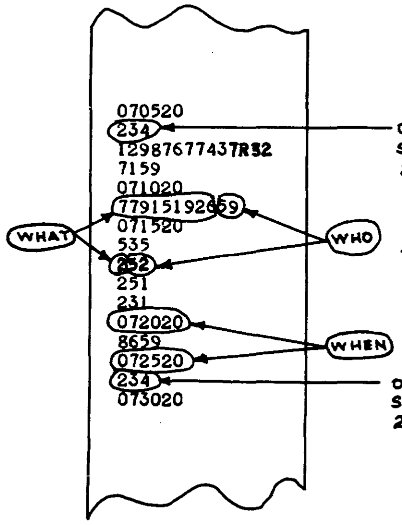


Figure 9. Typical Segment of Raw Data Stream.

Type of Station	Number of Positions	Total Number of Messages	Number of Characters per Message Auto	Card	Manual	Total Number of Characters per Message	Day
Assembly	2	280	3	5	3	11	3080
Wire	36	280	3	5	0	8	2240
Inspect	4	280	3	5	6*	14	3920
Test	2	280	3	5	3*	11	3080
Pack	1	280	3	5	0	8	2240
Clock	1	100	7	0	0	7	700
Supervisors	1	30	3	5	5	13	390
Total per Day		1530					15650
* Typical							

Figure 10. Daily Expected Information Flow Per Shift.

Number of Characters Entered (Per Day)		Rate of Entry (Ch/sec)	Time Required (Seconds)
Automatically	4990	20	250
Through Card Reader	7150	10	715
Manual	3510	2	1755
Total Daily	15650		2720 or 45 minutes

Figure 11. Expected Per Shift Busy Time.

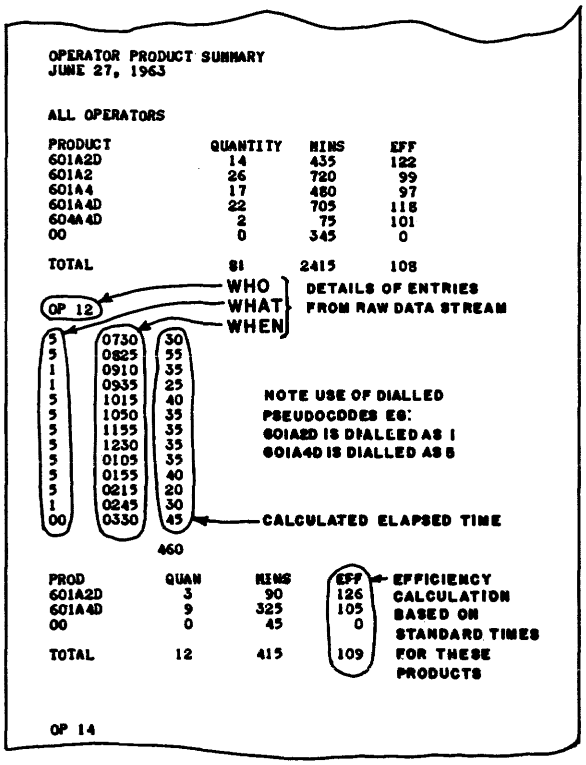
WHEN is provided automatically, as described earlier, by the Digital Time Generator.

The WHAT for each station is then the only information required to be entered by the using operator. The bulk of the WHAT messages become an error free semi-automatic operation through the use of a coded prepunched card.

The assembly operator originates one of the more important transactions: He enters a precoded card which will be associated with a frame until it is packed. He must also dial a code which identifies the type of frame he hangs the card on. Subsequent entry of the card by a wiring operator associates that particular unit with that operator. Entry at the time of starting work on the unit, by inference, signals termination of work on the unit previously reported.

Inspectors and Testers have more complex entries to make in addition to the card; they must dial in defect codes and quantities of each type found.

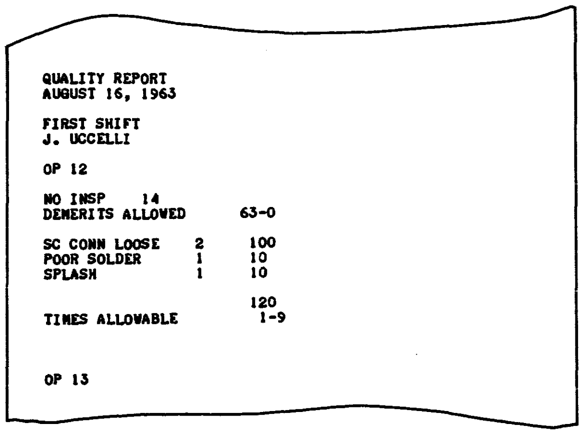
The computer can readily assimilate the above facts to produce a very broad spectrum of control information. This spectrum is limited only by the amount and accuracy of background information also available to the computer for reference. Figure (12) shows the



```
OPERATOR PRODUCT SUMMARY
JUNE 27, 1963

ALL OPERATORS

PRODUCT            QUANTITY   MINS    EFF
601A2D                14       435    122
601A2                 26       720     99
601A4                 17       480     97
601A4D                22       705    118
604A4D                 2        75    101
00                     0       345      0

TOTAL                 81      2415    108
                    WHO  ]  DETAILS OF ENTRIES
OP 12               WHAT }  FROM RAW DATA STREAM
                    WHEN ]
5    0730   30
5    0825   55
1    0910   35
1    0935   25
5    1015   40         NOTE USE OF DIALLED
5    1050   35         PSEUDOCODES EG:
5    1155   35         601A2D IS DIALLED AS 1
5    1230   35         601A4D IS DIALLED AS 5
5    0105   35
5    0155   40
5    0215   20
1    0245   30
00   0330   45    CALCULATED ELAPSED TIME
            460

PROD        QUAN      MINS     EFF   EFFICIENCY
601A2D        3         90     126   CALCULATION
601A4D        9        325     105   BASED ON
00            0         45       0   STANDARD TIMES
                                     FOR THESE
TOTAL        12        415     109   PRODUCTS


OP 14
```

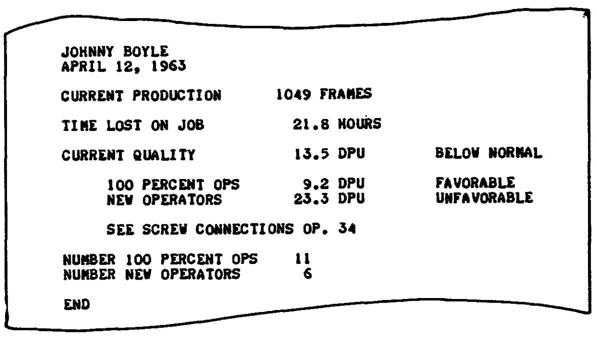
Figure 12. Daily Production Analysis (By Operator).

type of control information developed as a function of facts collected and background information available. Figures (13) – (15) are indicative of the types of reports which it is technically feasible to provide, either automatically or on demand, to the various levels of management through low cost page printers located near the recipients.



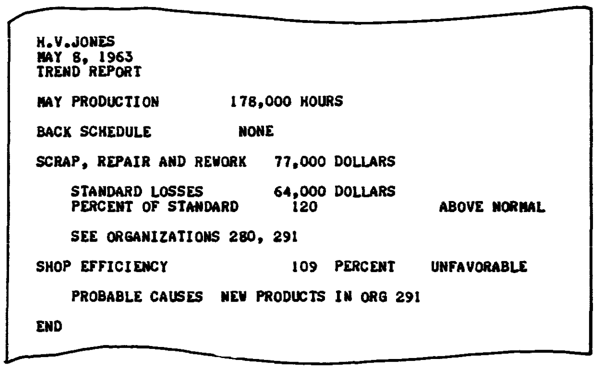
```
QUALITY REPORT
AUGUST 16, 1963

FIRST SHIFT
J. UCCELLI

OP 12

NO INSP      14
DEMERITS ALLOWED      63-0

SC CONN LOOSE   2     100
POOR SOLDER     1      10
SPLASH          1      10

                      120
TIMES ALLOWABLE        1-9


OP 13
```

Figure 13. Daily Quality Report (By Operator).



```
JOHNNY BOYLE
APRIL 12, 1963

CURRENT PRODUCTION          1049 FRAMES

TIME LOST ON JOB            21.8 HOURS

CURRENT QUALITY             13.5 DPU        BELOW NORMAL

     100 PERCENT OPS         9.2 DPU        FAVORABLE
     NEW OPERATORS          23.3 DPU        UNFAVORABLE

     SEE SCREW CONNECTIONS OP. 34

NUMBER 100 PERCENT OPS      11
NUMBER NEW OPERATORS         6

END
```

Figure 14. Weekly On-Request Report to Section Chief Operating.



```
H.V.JONES
MAY 8, 1963
TREND REPORT

MAY PRODUCTION          178,000 HOURS

BACK SCHEDULE             NONE

SCRAP, REPAIR AND REWORK    77,000 DOLLARS

     STANDARD LOSSES        64,000 DOLLARS
     PERCENT OF STANDARD     120             ABOVE NORMAL

     SEE ORGANIZATIONS 280, 291

SHOP EFFICIENCY              109 PERCENT   UNFAVORABLE

     PROBABLE CAUSES   NEW PRODUCTS IN ORG 291

END
```

Figure 15. Shop Trend on Request Report to Superintendent.

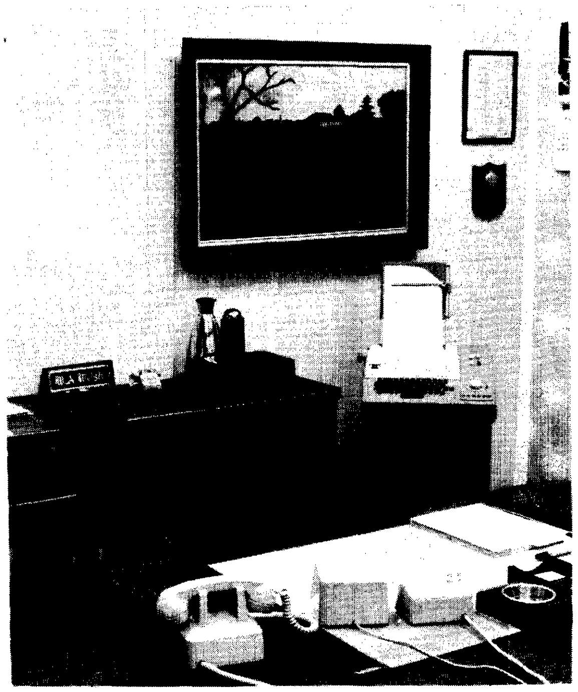


Figure 16. A Low Cost TELETYPE Page Printer.

THE PIECE PARTS SHOP

Another embodiment of the concepts put forth so far is a newly installed Data Collection System in a metal piece parts shop. This is a shop where about 125 people on each of two shifts produce small metal parts using a variety of manual, semiautomatic, or automatic machine tools such as punching and stamping machines, shears and power brakes, screw machines, lathes and milling machines. The work travels in lots of from less than ten units to tens of millions of units per lot. The number of operations to be performed on a lot varies from one to more than twenty with an average of about five. A computer provides the shop with schedules for control of production. By having access to current "setup" and "make times" rather than theoretical values, changes in efficiency or capacity can be taken into account by the computer to yield more realistic schedules.

Standard production messages which are transmitted to the computer from the shop indicate the start and finish of each operation on each lot together with the identity of the machine on which it is being performed and the time of the day at which each occurrence takes place. In addition, there must be inspection reports and a number of other "special situation" reports to handle the variety of occurrences in the shop. A total of 18 different types of transmissions are used to cover the variety of situations (Figure 17).

A given message may be formed by appropriate combination of the following components:

1. Transaction Code: One or two digits which define the nature of the message.
2. The identification of the particular job or lot travelling through the shop.
3. An operation number to identify the one of several operations.
4. Machine group to identify the particular machine used for running this job.

Various combinations of the four items account for the bulk of normal production messages. In addition to these there are several special message types such as lot splits, quantity changes, inspection results, error corrections, etc., which may also involve quantity.

TAPE FORMATS

Format: / Digit #	5	6	7	8	10	11	12	12A	15	Format: / Digit #
1	ST	ST	ST	ST	ST	ST	ST	ST	ST	1
2	ST	ST	ST	ST	ST	ST	ST	ST	ST	2
3	SC	SC	SC	SC	SC	SC	SC	SC	SC	3
4	T	T	T	T	T	T	T	T	T	4
5	C	T	C	T	C	T	T	C	C	5
6	C	C	C	C	C	C	C	C	C	6
7	C	C	C	C	C	C	C	C	C	7
8	C	C	C	C	C	C	C	C	C	8
9		C	P	C	P	C	C	P	P	9
10			P	P	P	P	P	P	P	10
11				P	M	P	P	Q	Q	11
12					M	M	C	Q	Q	12
13					M	M	C	Q	Q	13
14						M	C	Q	Q	14
15							C	A	C	15
16									C	16
17									C	17
18									C	18

PRINTED FORMAT

```
for one-digit transaction codes:  ST ST SC  D DDDD DD DDDD DDDD
for two-digit transaction codes:  ST ST SC DD DDDD DD DDDD
```

Legend:

ST	Station Number digit	P	Operation Number digit
SC	Scanner identification (includes both zone bits for aborted messages	M	Machine Grp. Card digit
		Q	Quantity digit
		A	"Accept" digit (1)
T	Transaction Code digit	D	Printed digit from Memory
C	Lot Card digit		

INPUT MESSAGE FORMATS

Standard	Format	Trans. Code
Interrupt: Resume Work	5	2
Set-Up Start	10	3
Set-Up Complete	10	4
Run Start	10	5
Run Complete	10	6
Rework Start	5	7
Interrupt: Stop Work	6	91
Rework Complete	6	97
Shut Down	11	01
Special		
Lot Split	15	2
Quantity Change	15	3
Accept, Void Transaction	7	4
Reject	5	5
Accept Complete Lot	12A	6
Rework Start	5	7
Run Complete, No Time	8	96
Run Complete, Void Transaction	8	94
Lot Renumber	12	03
Job Merge	12	02

Figure 17. Tape and Message Formats.

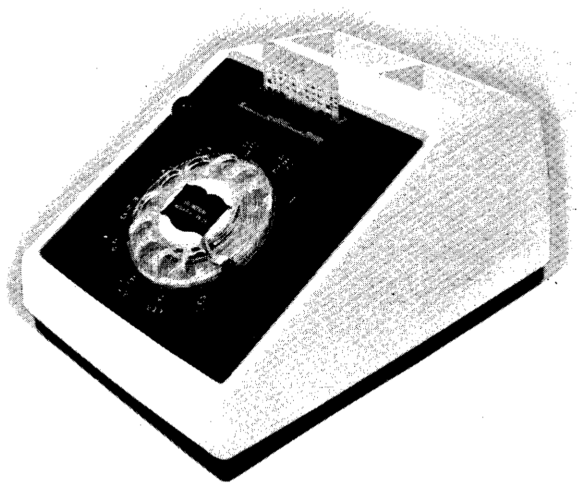


Figure 18. The Basic Station.

Such special messages, however, are restricted and are permitted only from a limited number of special stations set aside for that purpose.

The GATES II Data Collection System

The basic station is very simple in nature, consisting of a rotary telephone dial for variable information entered by the human operator, an automatic card dialer for entry of fixed information and a signal lamp for control purposes.

A limited number of special stations also are provided. These stations are logically similar to the basic station described above with the addition of four Burroughs sphericular displays. Their purpose is to allow verification of such information as quantity which is entered only from these stations. The right-most display unit is used to guide user by identifying the fields within a message. Such expressions as: CODE, CARD, OP #, and QUANT are displayed. Preservation of the fixed length of individual fields is desirable for purpose of error detection. In the case of quantities, which normally require one to eight digits, it was decided to represent all quantities with four digits; the first three significant digits (including leading zeros when necessary) plus a magnitude indicator. Reported quantities will be in error by no more than 1% which is satisfactory for this application. The magnitude indicator is equal to the number of digits in the original number. Thus: 12,425 would be entered as 1245 and displayed as 12400; 231,604 entered as 2316 and displayed as 231,000, etc. The fourth display device is arranged to present the proper number of low-order zeros to give a rounded number in proper form.

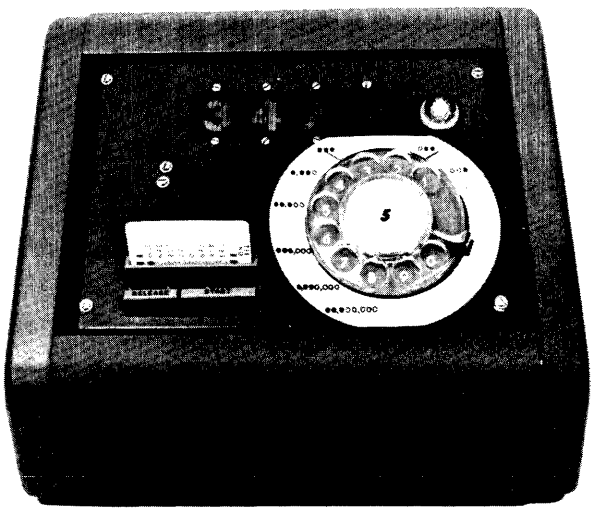


Figure 19. The Special Station.

A digital clock is provided for logging of time. Every two minutes the clock enters the time of day if there has been activity since the last two minutes interval.

Basic Station Operation

Let us consider a situation where a worker wishes to make a standard production report indicating, for example, that work has been finished on the run of a particular job. This person takes a card on which is coded the job or lot identification number and also a card which identifies the machine which was used for this operation, to a basic station. The lot identification card is inserted into the slot of the card reader. Now the user requests service from the system by simply dialing a "one". The central control equipment recognizes this operation of the dial, remembers the request for service, and when available, connection is made to the station and the readiness to receive data is indicated to the user by the lighting of its signal lamp—this lamp is referred to as the service light. The user has ten seconds to notice that the service light is lit and begin entering the first item of information which is a transaction code. The system is set up to allow transaction codes to be either one or two digits in length thereby allowing subclassifications of major transaction codes. Entry of the first digit of the transaction code immediately determines the length of the message, its format, and field delineation. The first item after the transaction code in a job completed message is the job-identifying card. To cause this card to be read, the user merely depresses the start bar. As the card is read, the service light flashes. When the card stops moving and the service light ceases to flash, the user realizes that the reading of the card is complete and enters the next item of information which is a two-digit operation number taken from a layout or instruction sheet traveling with the job. Three digits from the machine card are read and, in the case of a complete message, the transmission is terminated. The control equipment having received the proper number of digits previously determined by the transaction code, disconnects the station and extinguishes the service light.

Human Engineering Features

Several aspects of this operation are interesting from the human point of view. The first is the use of a timer to measure the elapsed time between successive digits. Three different timing intervals are provided. Ten seconds are provided from the moment the service light is first lit until data begins and also at those times when a user may be removing or inserting a card from the card reader. Five seconds are permitted between successive digits within a field of variable information being entered from the dial and approximately half a second is permitted between successive card-reader-originated digits for the purpose of catching attempts to dial fixed information from the rotary dial. The result of any of these timing intervals being exceeded is the termination of service for that particular input device and the extinguishing of the service light. The control equipment, realizing that a message has been terminated prematurely, discards all data entered so far and considers the message aborted. When the message is entered properly the service light will go out immediately after the entry of the last digit of the message, no sooner, no later. The user is instructed that in the event the light should go out before the message has been completed, or in the event that it remains lit after the message is completed, then an error condition is indicated and this particular message will be discarded. The operator must reinsert the message from the very beginning. If the operator senses that he has made an error in mid-stream, he simply stops what he is doing, allowing the current time interval to be exceeded and causing the message to be aborted automatically. The ability to deliberately abort a message in process gives the user a degree of freedom in catching his own mistakes.

A second aspect from the human standpoint is the organization of the data sequence for entry. First, two fields of variable data are separated by a field of fixed data from a card. This separation is amplified in the mind of the user and confusion between adjacent fields is reduced. Secondly, the last field of information is fixed. This means that the user has the option to abort even the last digit of variable information and should not enter the last card unless he is reasonably sure that all data to that point has been entered correctly. If the last field of data were variable then the control equipment would receive the last dial digit and take away service assuming that it had a correct message. The user then would not have an opportunity to abort the message if he suddenly sensed that the last entered digit was an error. In the one instance where the data could not be organized to assure that a card would be last, a special pseudo field is required consisting of the single digit "one". This is the last digit entered and serves as an indication of acceptance of the message.

System Organization

A block diagram of the control system of GATES II is shown in Figure (20). Each station is connected with its own three-wire cable to the central control unit where an appearance is connected to several station scanners, each consisting of a rotary stepping selector which is capable of connection to any of the stations. Each station scanner is the input to one of three service channels. Their outputs converge in the punch and printer control which controls the outputs to a printer located in the shop and a paper tape punch located remotely in the computer room. The printer provides a listing of certain selected types of messages as explained

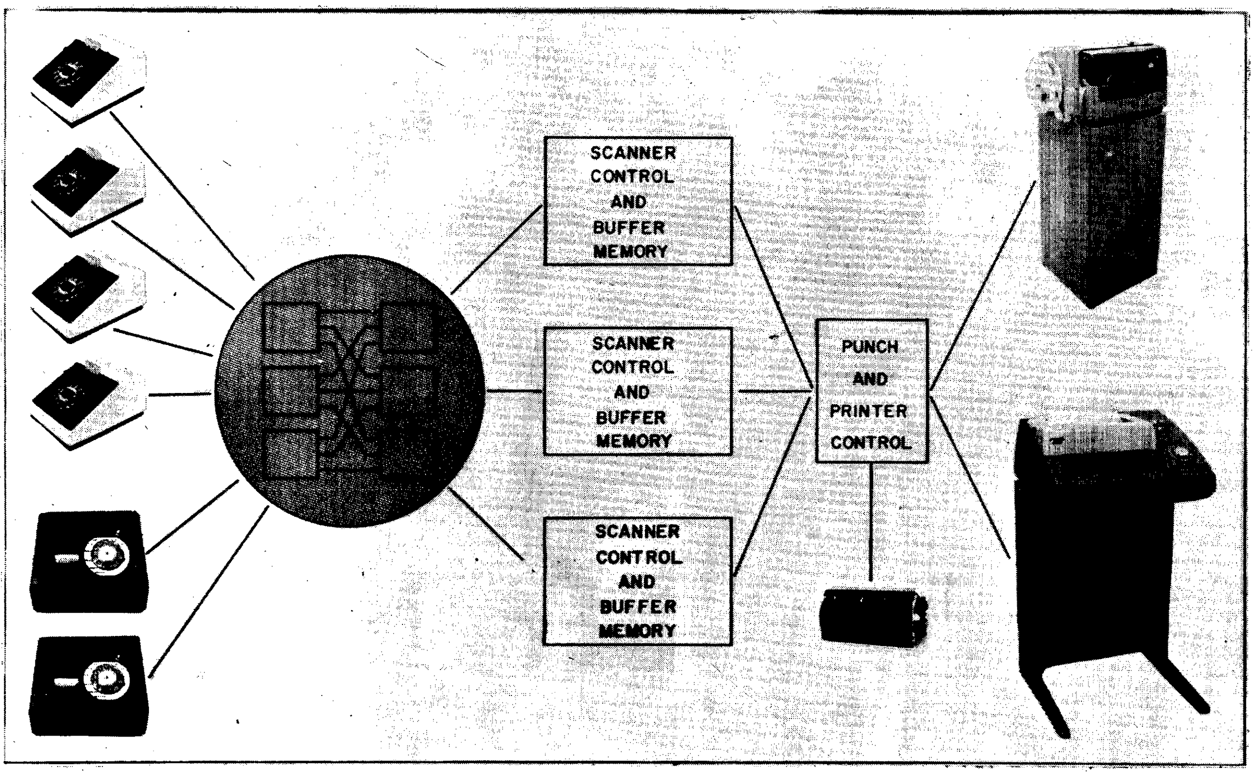

SCANNER CONTROL AND BUFFER MEMORY
SCANNER CONTROL AND BUFFER MEMORY
SCANNER CONTROL AND BUFFER MEMORY
PUNCH AND PRINTER CONTROL

Figure 20. Organization of the GATES II System.

further. Any failure of a unit within a service channel or the disconnection for maintenance or repair of any unit causes the associated service channel to be made automatically busy and traffic is routed around to other available channels.

When a station requests service, a line relay associated with it operates and locks to remember the request. The first available service channel is then connected, the light in the station is turned on, and the first timing interval is begun. The first digit received which is a transaction code is stored in the first digit position of the buffer memory and interpreted to determine the number of digits which should be received in this particular message and the interdigital timing between each successive digit. In addition, the combination of the transaction code and the identity of the calling station is used to determine whether or not this particular message should also be printed. In the event that it is a printing message, the separation of fields on the printer by spaces is also determined at this time. As each successive digit is entered, a digit counter advances and stores each received digit in the appropriate position of a 15-digit buffer memory. When the proper number of digits are received, or in the event of timeout, the service light is extinguished. The scanner directs the punch and printer control to connect to the punch or the punch and printer and proceeds to output characters previously stored. Characters are outputted at approximately nine characters per second when punching and printing or at 18 characters per second when punching only. The tape receives information in BCD "Flexowriter" code while the printer receives ASA X3.4 code. The first information to come from the scanner control is the identification of the calling station and the identification of the particular service channel which is being used. Immediately upon emission of the station number, the station scanner is freed and made available to hunt and connect to any other station requesting service. The illumination of the service light, however,, is delayed until the scanner control has completed outputting the previous message and has properly reinitialized itself. When the last digit is cleared from the buffer memory and a blank character is attempted on the next cycle, the punch and printer control disconnects the scanner and executes the termination sequence which consists of a carriage return signal to the punch and a carriage return line feed signal to the printer. The scanner control then clears its memory and resets to its proper initialized condition.

CLOSING COMMENTS

We have a general approach to the handling of special purpose information flow systems allowing us to provide at short notice economical source data collection.

This is done through a versatility of technique rather than through an all purpose station. The concerted effort to use mass produced low cost elements is an important factor in keeping equipment costs down.

By providing, as described, inexpensive and easy to use data collection systems, we are able to justify the use of computers in our manufacturing planning, control and supervisory functions. Many of these functions could not have otherwise been performed short of being part of much more complex projects, which, because of their complexity, may never have been tackled.

The authors wish to express appreciation to their colleagues who helped make these systems possible—in particular, to Jim Gorman in the areas of circuit design and Miss Bonnie Small who was responsible for the design of the "management information as required" concepts.

REFERENCES

1. The Globotype Telegraph. DAVID MCCALLUM (London, 1856).
2. Telegraphy in the Bell System. P. S. HOWE (AIEE—TP44-129, 1944).
3. Random Access and the Manufacturing Control Problem. C. A. R. KAGAN (AIEE —CP-57-1063, 1957).
4. Push Button "Dialing". H. F. HOPKINS (Bell Lab Record, Vol. 38, No. 3, March 1960).
5. A Production Control System Using a Small Computer. R. TEVONIAN (AIEE—Winter General Meeting, 1961).
6. On the Control of Production in Small Job Shops. S. E. ELMAGHRABY and R. T. COLE (12th Annual AIIE Conference, May 1961).
7. Central Office Receiver for Touch Tone Calling. C. G. MORRISON (Bell Lab Record, Vol. 39, No. 6, June 1961).
8. The Versatility of Touch Tone Calling. H. E. NOWECK (Bell Lab Record, Vol. 39, No. 9, September 1961).
9. The Automatic Card Dialler. W. PFERD and H. J. HERSHEY (Bell Lab Record, Vol. 39, No. 10, October 1961).
10. Central Office Modification for Touch-Tone Calling. N. LAZO and A. S. MARTINS (Bell Lab Record, Vol. 39, No. 12, December 1961).
11. Notes on the Philosophy of Automatic Data Processing for Business Applications. C. A. R. KAGAN (Proceedings of IFIP Congress '62—North Holland Publishing Company, 1963).
12. A Single Server Queue with Feedback. L. TAKACS (Bell System Technical Journal, Vol. 42, No. 2, March 1963).
13. A New Line of Low-Cost Light Duty Teletypewriter Equipment. N. A. JACOBS (AIEE—CP63-462, 1963).
14. Touch-Tone Card Dialer Set. J. H. HAM and J. F. RITCHEY (Bell Lab Record, Vol. 41, No. 7, July-Aug., 1963).

ENGINEERING CHARACTERISTICS OF CYLINDRICAL THIN-FILM PARAMETRONS FOR USE IN DIGITAL SYSTEMS

B. A. Kaufman, W. G. Pfieffer, V. K. Randery, and A. J. Kolk
The National Cash Register Company
Electronics Division
Hawthorne, California

I. INTRODUCTION

The parametron, a phase-locked subharmonic oscillator, is one of the newer devices available to the system designer for use as a logical element in digital systems. The basic concept of phase-locked oscillators for logical operations has been discussed in the literature.(1) (2) Parametrons are inherently reliable, being composed of passive elements only, are potentially inexpensive, and possess a unique universality since one basic type of parametron may be interconnected to perform all logical functions (gates, flip-flops, comparators, etc.).

Compared to more conventional ferrite parametron devices, thin-film parametrons(3) have several advantages. From a systems viewpoint, the most promising one is increased operating speed. The rotational properties of thin magnetic films permit an increase in speed over conventional ferrite units by an order of magnitude or better.

A parametron utilizing an electrodeposited thin-film on a small cylindrical wire substrate offers many attractive features. The continuous plating technique used for preparation of the thin magnetic film results in improved uniformity at reduced costs.

Use of the substrate wire to carry the pump current provides a low inductance pump circuit, allowing a large number of parametrons to be driven from a single pump source, as well as providing close coupling between pump field and magnetic film.

II. BASIC PARAMETRON CONCEPTS

For a detailed review of parametron logical circuit operations, the reader is referred to Goto.(1) However, a brief review of the basic functions performed by a parametron will be helpful.

The parametron is capable of implementing a complete logic system. As such, it has inherent bistability which exists in the phase domain, since the subharmonic oscillation can exist only in two distinct phase states which are 180° apart.

Directionality, the ability to propagate signals in one direction only, is complicated by the fact that the input and output for a parametron exists at the same two terminals. However, by means of a "three-beat" excitation system, this problem can be overcome.

The device possesses circuit gain, typically in the order of 60 db, and "live" storage may be provided by a closed loop of three parametrons circulating information in a ring. A unique feature of the parametron is its ability to provide logical inversion without delay. This may be obtained by reversing the winding sense of the input term to the input transformer.

Parametrons are basically threshold elements; as such, they are more efficiently used with logic written in majority notation. This operation involves the algebraic summation of all input signals to the parametron. The net signal provides the "seed" signal to establish the phase state of the parametron as the pump is turned on.

Since phase is a relative concept, a standard for comparison must be established in a system. This is normally done by connecting three parametrons in a ring and defining the state of the ring as a "one" for the entire system. All references to "one" or "zero" are made with respect to this ring. Aside from this restriction, normal concepts of majority logic will apply.

It may be seen that it is possible to mechanize, in a primitive fashion, "and", "or", "not" functions—the normal Boolean set. However, with a threshold device, the fan-in capability is rapidly dissipated in this inefficient mechanization. For example, the parametron to be described here has a fan-in capability of 7. This allows only four-input Boolean gates. However, it is known that, in many cases, a given problem can be implemented more efficiently using majority logic. This is especially true of comparators and carry circuits for adders. A unified analytic theory of majority logic design is slowly evolving(4) (5) but has not yet reached the refinement of present conventional Boolean logic synthesis.

III. DEVICE CONFIGURATION

Figure 1 is a circuit of a single cylindrical thin-film parametron element. A 20-mc pump current, approximately one ampere peak to peak, superimposed on 500 ma dc-bias, is passed down the substrate wire to provide pumping action. Inputs are transformer-coupled by one-turn loops through the ferrite toroid. Thus, a completely floating system with good common mode noise rejection is obtained. The use of a lossy core provides proper phase shift (theoretically 90°) of the seed current to achieve phase matching between seed current and circulating (tank) current of the parametron to be controlled.

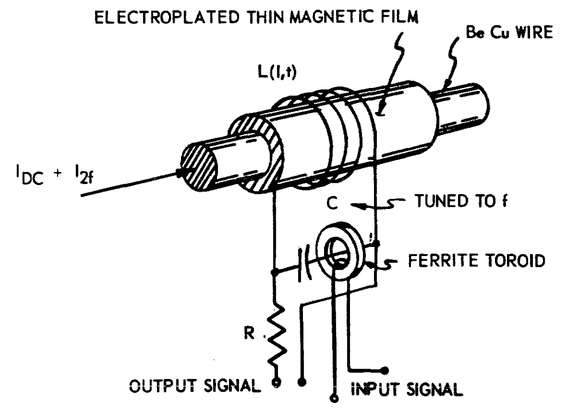


Figure 1. Cylindrical Thin Film Parametron.

For ease of application in experimental systems, parametrons are assembled in modules of 12 units as shown in Figure 2, each with appropriate input and output connectors. The plated wire is positioned in a slot milled in the board. An etched circuit strip on the back forms a return path reducing the inductance of the pump line. The 40-turn tank solenoids are preformed on an automatic winder, slipped over the plated wire, and assembled on the board, one piece of plated wire serving twelve parametrons. Resistors and capacitors are connected in normal fashion. The outputs are brought through sets of pins at the top for connection to the logical circuitry. The aperture through the input core is used for the logical input.

IV. WIRE PLATING PROCESS

The electroplated wire is prepared in a continuous plating process as shown in Figure 3.

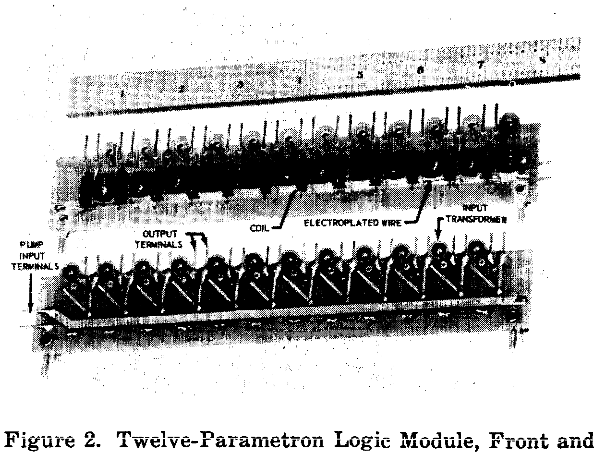
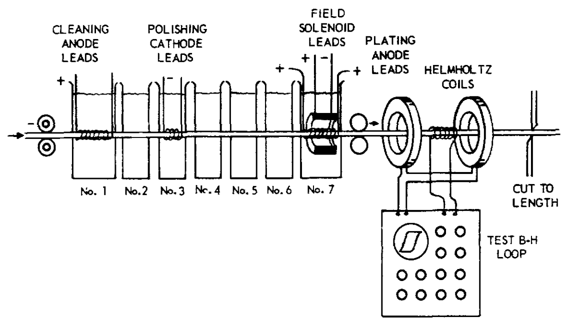


Figure 2. Twelve-Parametron Logic Module, Front and Rear View.



No. 1. - CATHODIC ALKALINE CLEANER
No. 2. - DISTILLED WATER RINSE
No. 3 - ELECTROCHEMICAL POLISHING
No. 4. - DISTILLED WATER RINSE
No. 5. - ACID ETCH
No. 6. - DISTILLED WATER RINSE
No. 7. - MAGNETIC FILM PLATING SOLUTION

Figure 3. Flow Diagram of Electrodeposited Thin Film Parametron Fabrication Process.

Prestraightened 10 mil diameter beryllium-copper wire is first given a thorough cathodic cleaning, then electrochemically polished and chemically etched, and finally plated with 81Ni-19Fe alloy (permalloy) using a variation of Wolf's plating bath.[6] A uniform magnetic field of 120 oersteds is applied parallel to the axis of the wire during plating to produce the uniaxial magnetic anisotropy. It should be mentioned that a plated wire with circumferential rather than axial anisotropy would perform in a similar manner. However, production of plated wire with axial anisotropy is a less complex process and is none-the-less suitable for this application.

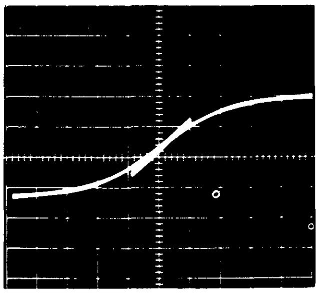
The magnetostrictive characteristics of the permalloy are critical with regard to producing material of good uniformity. Insensitivity to stresses introduced during assembly also necessitates low magnetostriction. The magnetostriction is a function of the nickel-iron ratio and can be minimized by proper adjustment of the plating temperature.

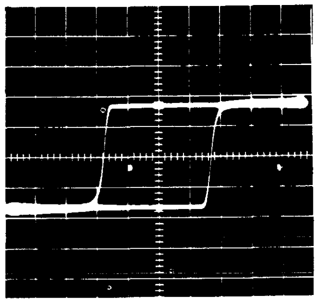
Typical low-frequency B-H characteristics are shown in Figures 4(a) and 4(b). For a 1.2 micron-thick plate, the coercive force, H_c, is normally 1.5 oersteds and the anisotropy field, H_k, is 2.5 oersteds, when measured from the B-H loop.[7] The threshold in the easy direction is above 90% of H_c, and the hard direction loop remains closed up to saturation.

V. THEORY OF OPERATION

A phenomenological theory of operation for a cylindrical, thin-film parametron may be based on the concept of the permeability tensor of the plated film. If the film possesses axial anisotropy, a pumping current applied down the substrate generates a circular H field which causes $\vec{M}$ to rotate in an effort to align itself with the applied field. As this field is varied sinusoidally, $\vec{M}$ executes appropriate oscillatory motion. Thus, the film permeability, which is the slope of the B-H loop presented to the subharmonic tank coil, is made to vary, and a time-variable inductance due to pumping results. The variation is made symmetrical by superimposing the pump current on a dc-bias current. Using films with relatively high anistropy and low H_k, it is possible to meet the criteria of parametric oscillation. The nonlinearity may be enhanced by use of a film with $H_k \approx H_c$.

Because the steady state subharmonic tank current is considerable (100 ma pp), the effect of the axial field produced by this current must be considered. The combined axial and transverse fields produce a three-dimensional B-H


(4 A) Hard Direction


(4 B) Easy Direction

SCALE: 1 oe/division

Figure 4. B-H Loops.

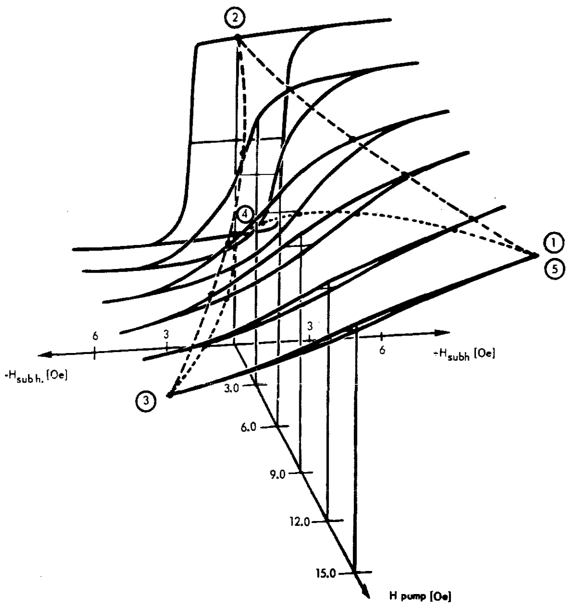


Figure 5. Three Dimensional B-H Trajectory.

trajectory (Figure 5) whose shape may be deduced from Figure 6.

The area traced out by points ①-②-③-④-⑤ over a subharmonic cycle may be considered as representing a power loss. Since the pump excitation field and the axial fields are sufficient to drive the film to the near-saturation region (saturation pumping), any increase in applied field (axial or transverse) will result in only

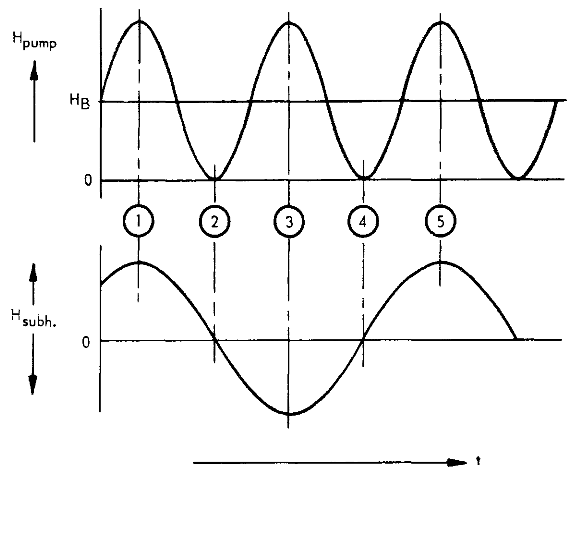


Figure 6. Pump and Subharmonic Field Variations.

a slight increase in area. Losses are therefore relatively constant if sufficient pump field is applied for "saturation" in the transverse direction, and the subharmonic tank current provides the axial field to "saturate" in the axial direction. The shape of this area is determined by B_{max}, H_k, and H_c of the film, which are carefully controlled and highly reproducible.

The perimeter of area ①-②-③-④-⑤ also represents the locus of varying permeability presented to the tank coil over a pump cycle. This varying film permeability determines the subharmonic oscillation buildup which is of the form[1]

$$i = KeA\ (\gamma - \delta)t$$

where

$$\gamma = \frac{2\Delta L}{L_0}$$

$$\delta = \frac{1}{Q}$$

The constant area caused by pumping to "saturation" results in a fixed γ which, with the equally fixed δ, tends to stabilize the oscillation amplitude and buildup rate.

VI. DEVICE PERFORMANCE

Previous mathematical treatments of parametron operation yielded only general, approximate solutions,[1] and little information of operation in the time domain was available. To gain additional insight, a low-speed parametron was simulated on an analog computer.[8] The data thus obtained was of considerable value for determining an advantageous operating point, judging the influence of phase shifts in signal and pump lines, and verifying results obtained with the actual high-speed parametron.

A. *Choice of Pump Frequency*

With the circuit Q's possible, approximately 5 to 10 cycles are required for the parametron to reach steady state after the pump is switched on. Thus, the clock or gating rate is directly dependent upon the pump frequency 2f. A high value of 2f is desirable from the viewpoint of high-speed operation. However, distribution of the pump current becomes more difficult with increasing frequency. The chosen frequency of 2f = 20 mc is an engineering compromise which allows relatively high operating speeds (up to 300 kc), yet requires relatively simple equipment for pump generation and distribution.

B. *Choice of Operating Point*

The operating point of a parametron is determined by the excitation (pump) current I_{2f}, and the bias current I_B. Figure 7 is a diagram of the region of operation of a typical parametron where the limit of operation was defined as an output current variation of ±4.2%, about a chosen nominal value of 4.8 ma. As will be shown later (Table II), a current variation of up to ±14% can be tolerated for a fan-in of 7. Obviously, the highest safety margin is obtained by choosing an operating point in the center of the "operational area". Additional factors influencing the choice are power consumption and limiting.

The power consumption, both rf and dc, should be kept to a minimum. Exploitation of the limiting mechanism, however, requires "saturation pumping", i.e., a power level higher than the desired minimum power consumption. Thus, the choice of the proper operating point is a compromise between three partly conflicting conditions—optimum safety margin, minimum power drain, and optimum limiting.

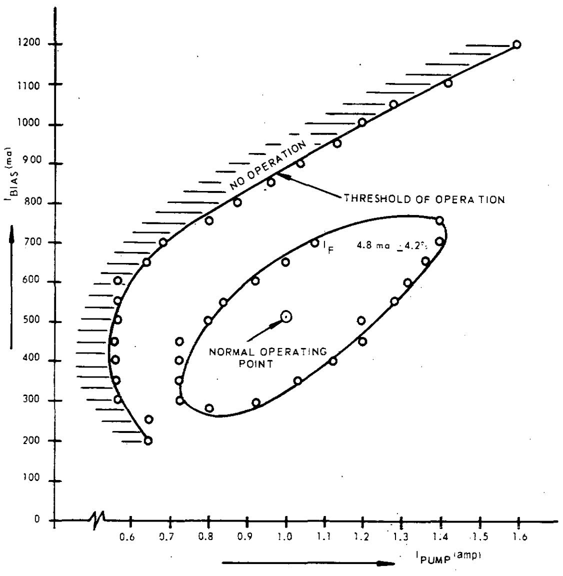


Figure 7. Region of Operation.

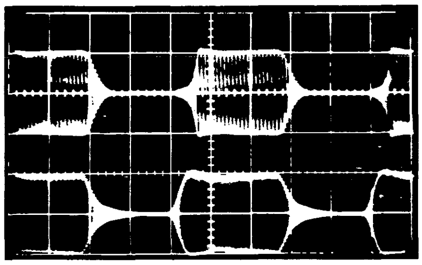


Figure 8. Switching Waveforms of a Typical Parametron.

Recognizing this, an operating point of I_{2f} = 1,000 ma pp, I_B = 500 ma was chosen. This results in pump power of approximately 80 mw (gated operation) which includes some 35 mw of copper loss in the substrate alone. These values were determined calorimetrically. This pump power results in practical requirements for system pump generators. Figure 8 shows the waveforms of a typical parametron gated at 200 kc.

C. *Fan-In Considerations*

Parametron majority logic is based upon the phase differences of an odd number of ideally equal input currents. The input currents derived from practical parametrons, however, spread over a certain amplitude range. The possible number of inputs is primarily determined by this range of tolerances. Table I shows the input current tolerance restrictions for majority gates of three to eleven inputs.

TABLE I

3 Inputs	—	2 I_{min}	>	I_{max}
5 Inputs	—	3 I_{min}	>	2 I_{max}
7 Inputs	—	4 I_{min}	>	3 I_{max}
9 Inputs	—	5 I_{min}	>	4 I_{max}
11 Inputs	—	6 I_{min}	>	5 I_{max}

Due to the high tangential sensitivity of the plated wire parametron, only a small excess current is required to assure reliable phase locking. Analog computer simulation[8] and experimental verification have shown that an excess current of less than 50 μa is sufficient to satisfy the requirements of Table I. For example, 7 inputs are feasible if 4 I_{min} exceeds 3 I_{max} by 50μa. Table II shows the allowable percentage deviations of the nominal input current I for majority gates with 3 to 11 inputs. Here again, the required excess current determines by how much the listed ideal tolerance limits have to be reduced.

TABLE II

3 Inputs	—	$I \pm < 1/3$ or	$\approx \pm < 33\%$
5 Inputs	—	$I \pm < 1/5$ or	$\pm < 20\%$
7 Inputs	—	$I \pm < 1/7$ or	$\approx \pm < 14\%$
9 Inputs	—	$I \pm < 1/9$ or	$\approx \pm < 11\%$
11 Inputs	—	$I \pm < 1/11$ or	$\approx \pm < 9\%$

In addition to input current variations, at least three more factors limit the number of inputs, and warranted investigation. First, phase shift along the signal line reduces the effective component of the input current. Second, coupling between signal lines can alter amplitude and waveform. Third, jump coupling (to be discussed below) requires an increased excess seed current to overcome the influence of the backward leakage signal.

The analog computer simulation(8) determined that the extent of allowable phase shift is $\approx 20°$. In a large system, some signal lines might exceed the maximum allowable length and excessive phase shift could result. In these cases, a buffer parametron, located next to the parametron to be controlled, can re-establish phase synchronism.

The buffer parametron, receiving only one input, is far more tolerant of phase shift of the input signal. However, in order to re-establish the correct propagation sequence, one additional clock cycle and two additional parametrons are sometimes required.

Coupling between signal lines and the resulting problem of amplitude and phase alteration can be kept within limits by judicious layout. To get a general feeling for the magnitude of this effect, the mutual inductance between two circular loops of six-inch diameter and three-inch spacing in the same plane was computed.

At 10 mc and a current flow of 4 ma pp in one loop, the induced voltage in the second loop will be 0.014 volt or 0.35% of the nominal output voltage of 4 volts pp.

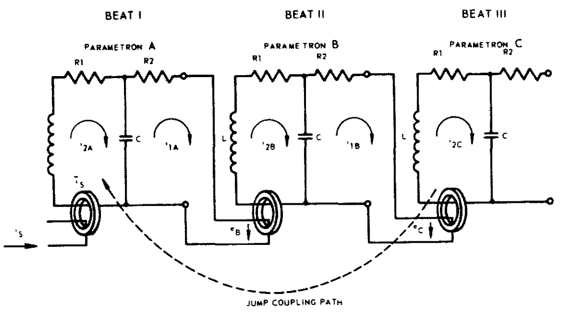


Figure 9. Jump Coupling.

In consideration of the above factors, the parametron was specified as capable of accepting 7 majority inputs.

D. *Fan-Out Limitations*

1. Jump Coupling. Figure 9 is the circuit diagram of three parametrons connected in series. Under normal operating conditions, parametron A will receive its input information from a parametron excited by Beat III. However, an undesired input may find its way from parametron C (which is also excited by Beat III) through the inactive parametron B into parametron A. An analysis of this network shows that a current, i_{2B}, is generated in the inactive parametron B by the active parametron C and is given by:

$$i_{2B} = e_C \frac{R_1 - j\omega\ (L + C\ R_1R_2)}{R_1^2 + 2R_1R_2 + \omega^2\ (L^2 + C^2\ R_1^2\ R_2^2)}$$

Substituting the values of the actual parametron

$$\begin{aligned} R_1 &= 10\,\Omega\ (Q = 3.14) \\ R_2 &= 1k\,\Omega \\ L &= 0.78\,\mu h \\ C &= 330\,pf \\ \omega &= 2\pi\,10^7\ (f = 10\text{ mc/s}) \\ e_C &= 100\text{ mv pp (measured)} \end{aligned}$$

one obtains

$$i_{2B} = (0.015 - j\,0.4) \times 10^{-3}\text{ a pp}$$

The voltage $e_C = 100$ mv pp is generated by a circulating current of $i_{2C} = 100$ ma pp in the active parametron C. The induced current i_{2B}, circulating in the inactive parametron B, is approximately 0.4 ma pp.* Assuming identical transformers, voltage e_B across the input transformer of parametron B will be smaller than e_C by the ratio i_{2B}/i_{2C}

$$e_B = 0.1 \frac{0.4 \times 10^{-3}}{0.1} = 0.4 \text{ mv pp}$$

Assuming, for purposes of this analysis, that the active parametron A presents an impedance network similar to its inactive one, the voltage e_B will then induce a current i_{2A} in parametron A of magnitude

$$i_{2A} = (0.015 - j\,0.4) \times 10^{-3} \times \frac{0.4 \times 10^{-3}}{100 \times 10^{-3}}$$
$$= (0.06 - j\,1.6) \times 10^{-6} \text{ a pp}$$

The effective value of this current is approximately

$$i'_{2A} = 1.6 \times 10^{-6} \text{ a pp}$$

The magnitude of the desired seed signal flowing in the i'_{2A} loop is $\bar{i}_s = 10^{-3}$ amp (as transformed to the secondary of the input transformer from i_s flowing in the primary). Therefore, the ratio of desired to undesired locking signals equals

$$\frac{\bar{i}_s}{i'_{2A}} = \frac{10^{-3}}{1.6 \times 10^{-6}} = 625$$

This means that up to 625 third-generation parametrons ("grandsons") could be connected. With the chosen fan-out of ten (which results in a maximum of 100 grandsons) jump-coupling is of second order importance only for low fan-in circuits. It should be noted that the above ratio is computed for unity fan-in. The worst-case available seed signal for a maximum fan-in of 7 is considerably less, restricting the logical designer from complete freedom to use maximum fan-in and maximum fan-out simultaneously for several levels of logic.

2. Phase Shift. Figure 10 shows the equivalent output circuit of a parametron. The phase of the output current, I_f, is determined by

$$\tan \alpha = \sum_{1}^{n} \frac{\omega L_n}{R_2}$$

* For a conservative analysis only the amplitudes have been considered, since any phase shift will weaken the influence of the jump coupling signal.

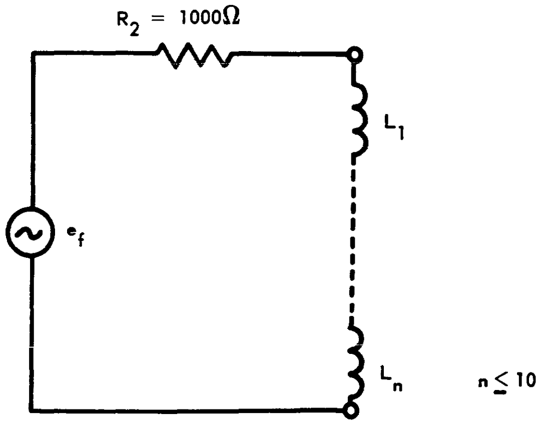


Figure 10. Output Circuit of a Parametron.

According to Reference (8), the required locking signal is essentially constant for phase angles between 0 and 20°

Thus, for $\alpha = 20°$ and $R_2 = 1000\ \Omega$,

$$\sum_{1}^{n} \omega L_n = 364\ \Omega$$

should not be exceeded

With $L = 0.03\ \mu h$ for a single input transformer, this allows a fan-out of

$$\frac{364}{2\pi\, 10^7 \times 3 \times 10^{-8}} = 19$$

In addition, series resistance of the actual circuit such as core and wire losses will reduce the phase shift so that even a fan-out of 19 would be conservative.

The chosen fan-out of 10 is clearly within the limits imposed by jump-coupling and phase shift.

VII. ASSOCIATED EQUIPMENT

Since the parametron approaches a true "universal" building block, all operations in a given system will most likely be performed in phase script. However, it is obviously necessary to provide interface to the "outside world". The necessity of a simple and inexpensive technique for converting to and from dc to phase script thus exists.

Parametrons, whose inputs are derived from saturable reactors operating at the subharmonic frequency, have been used as simple dc-

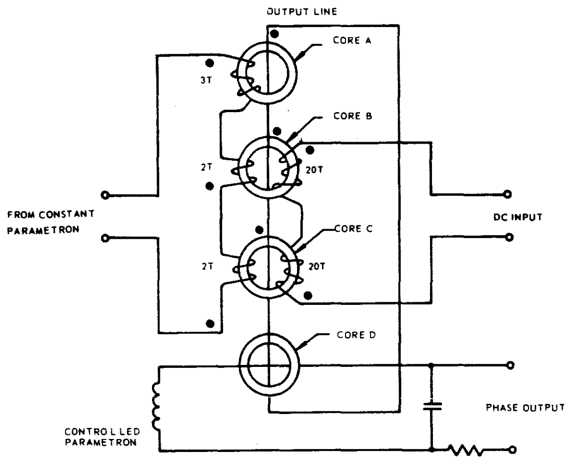


Figure 11. DC-Phase Converter Schematic.

phase converters.(9) The following is a description of such a device suitable for operation at a subharmonic frequency of 10 mc.

A. *DC-to-Phase Converter*

The schematic of a dc-to-phase converter is shown in Figure 11. Cores A, B, and C are mounted on a small, clip-on device which can be attached to a single parametron in the 12-parametron module, converting it to a dc-to-phase input stage. Core D is the input core of the controlled parametron. The number of turns on each core and their winding sense are shown in the schematic. Cores B and C are mounted together and have a common dc (control) winding. The relative sense of the control and signal windings is such that no rf voltage is induced in the control winding.

Core A injects three "unit weights" of input to the output line, while cores B and C provide four units in the opposite phase, resulting in a net output of —1 unit. As cores B and C are driven toward saturation by the dc control field, their contribution to the induced signal in the output line decreases. If, for example, it is reduced to more than 50% of its normal value, the phase reverses.

Tolerance calculations for the circuits have shown an extremely wide operating range with regard to the applied dc control current, largely due to the high tangential sensitivity of the parametron. The chosen dc control current of 300 ma assures reliable phase reversal. A photograph of the device is shown in Figure 12.

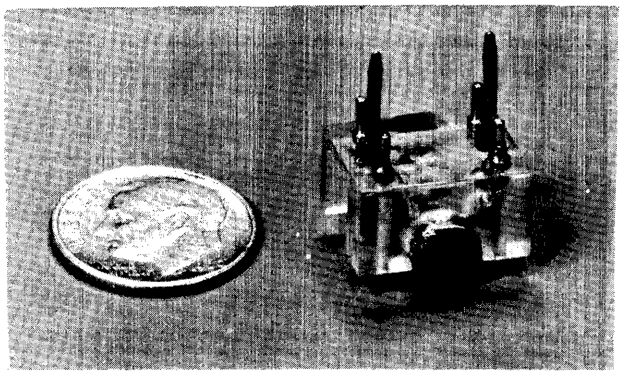


Figure 12. DC-Phase Converter.

B. *Phase-to-DC Converters*

The basic phase-to-dc converter requires a phase sensitive comparator. This is not as complex a circuit as one might expect. Since the phase and the amplitude are both quantized, the use of simple algebraic summing is possible.

With reference to Figure 13, basic operation is as follows: the outputs of two parametrons, one delivering a "constant" signal (for example, always "one"), the other, whose output is to be sampled, are fed to the input transformer of the comparator. Transformer coupling is preferred over direct coupling to eliminate any ground connections, to minimize ground loop noise, and to keep the entire parametron system floating and symmetrical to ground. This type of coupling is consistent with logical connections on the parametron modules proper. Input turns are kept to a minimum to allow the use of a standard logic output line for providing input signals to the phase-to-dc converter.

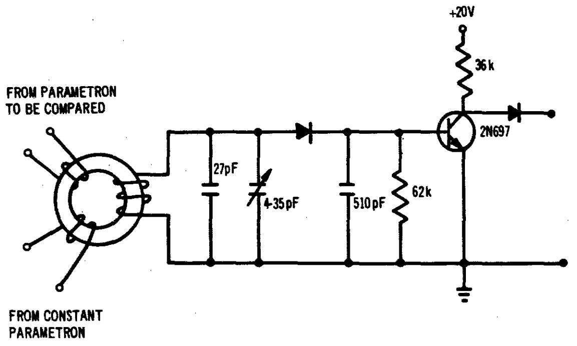


Figure 13. Phase-DC Converter Schematic.

If the input transformer is constructed with the same type core as used in the parametron modules, the inductive loading of the two input turns is equivalent to a fan-out of two. Since the input signal available from two turns is small, the secondary must be tuned to the subharmonic frequency (10 megacycles). Tuning rejects spurious noise and increases the output voltage. If the constant parametron provides input signals in the same phase as the unknown parametron, the two signals add and overcome the diode threshold, allowing rectification. However, if the constant signal is in opposite phase to the unknown parametron, the two signals cancel. The worst-case residual signal due to imperfect cancellation is insufficient to overcome the diode threshold.

Normal half-wave rectification is used with single stage RC filtering. Since the ratio of carrier frequency to modulator frequency is relatively low ($\approx$ 30:1), a pulse output up to 300 kc without "diagonal clipping" is difficult to achieve. Either more sophisticated carrier filters are necessary or the system design must be arranged to require no high duty-cycle outputs.

The dc voltage is now fed to a dc amplifier. Silicon diodes and a silicon input transistor are used to avoid I_{co} problems. The output of the amplifier is sufficient to drive further amplifying circuits.

C. *Phase Measuring Technique*

The testing of parametron logic systems is greatly enhanced by a method originally used and suggested by Japanese researchers.

Properly amplified, the output signal of a constant parametron is fed into the Z-input of an oscilloscope. Simultaneously, the output signal of the parametron to be tested is applied to the vertical input (Y-input) of the oscilloscope. Both input signals must be derived from parametrons excited by the same pump supply channel.

If both signals are in phase, only the upper half (positive portion) of the 10-mc signal under test will be displayed. If the signals are 180° out of phase, the lower half (negative portion) will be visible, the upper half being blanked out.

D. *Pump Supply and Distribution System*

The pump supply for operating parametrons corresponds to the power supplies for conventional logic. In this case, approximately 80 mw (at 20 mcs) per parametron is required in addition to dc-bias current.

A 20-mc supply, consisting of three identical channels capable of being gated at a 300-kc rate with appropriately overlapped pulses, was developed for use in a number of system experiments. The capacity of each channel is 25 watts cw. Each channel consists of three amplifier stages. The first stage, a 20-mc ECO, is common to all three channels. The output is distributed to each of three 7AK7 pentodes operating as gated amplifiers. The carrier is gated by 40-volt pulse signals applied to the suppressor grid. The third and final stage is a conventional, single-ended Class C amplifier employing a 6146 beam-power tube and driving a π-matching network to match to a 93-ohm output line. It should be noted that the use of a master oscillator-power amplifier combination is necessary since the preservation of phase coherence between all three channels is a requirement for parametron phase script logic.

A π output matching network is used because considerable sophistication is required in the pump distribution network for these frequencies. Distribution lines between the pump source and the parametron modules must be kept short. A special distribution transformer with a 93-ohm resistive input impedance was developed to allow driving from moderately long lengths of matched cable. For long runs, the cable lengths for each output (Beats I, II, and III) must be identical to insure matched phase shift due to cable delay in each line. This means that the pump supply can be located anywhere in the computer frame. Figure 14 is a

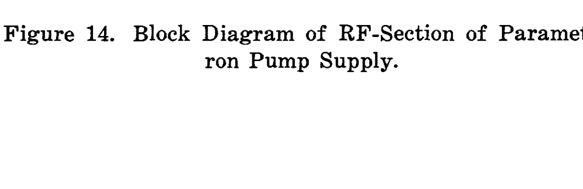


Figure 14. Block Diagram of RF-Section of Parametron Pump Supply.

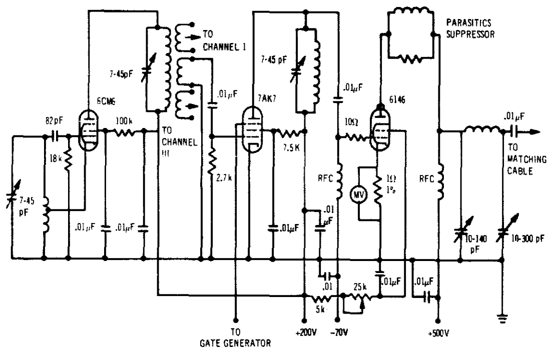


Figure 15. Parametron Pump Supply Schematic of One Channel.

block diagram of the entire rf section of the pump supply. Figure 15 is a schematic of a single channel.

Three different gating modes were incorporated into this pump supply: 1) 200-kc gating with nominal 33% overlap between adjacent beats, 2) manual gating with fixed overlap where alternately beats I, II, and III are energized, and 3) continuous mode where all channels run cw simultaneously. Operation in one of the three modes is controllable by a front panel selector switch. For the normal operation of parametron systems, 200-kc gating is used.

The 200-kc gating circuit block diagram is shown in Figure 16. It consists of a 200-kc Colpitts oscillator feeding sine wave signals to one-shot multivibrators through three phase shift networks so that each of the three one-shots is triggered with 120° phase shift. The output pulse width of each one-shot is adjustable so that the overlap between the turn-off of one channel and the turn-on of the adjacent channel is adjustable. This is ideally 33%. To compensate for the finite turn-on time of the parametron, a pump duty cycle greater than 33% can be utilized to make the overlap of the subharmonic signals equal 33% The output of each multivibrator drives a voltage amplifier providing a 40-volt gate for the 7AK7 stages.

Manual gating is provided by a scale-of-three counter, with manual advance by a front panel push-button switch. This is useful in trouble-shooting since the machine may be stepped through any cycle manually and any one state held continuously until manually advanced.

For the final mode, continuous gating, the 7AK7 gating tubes are switched to a dc level, allowing all channels to operate cw. This is useful for checking the phase of the 20 megacycle pump current in any parametron module.

In large systems with hundreds or thousands of parametrons, simple series connection of modules is unsuitable. Experiments have shown that the electrical length of the pump path in a single module of 12 parametrons is long enough compared to a wavelength, to cause severe standing waves if a number of these are connected in series. A distribution transformer scheme, shown schematically in Figure 17, was evolved, whereby a small group of modules, two in this case, are connected in series and driven from an isolated secondary winding. These are series-tuned to cancel module reactance; two such secondaries are coupled to a

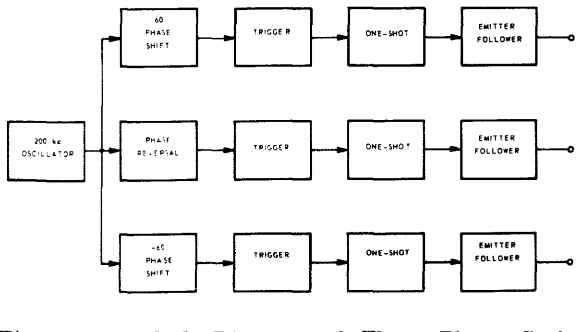


Figure 16. Block Diagram of Three Phase Gating System.

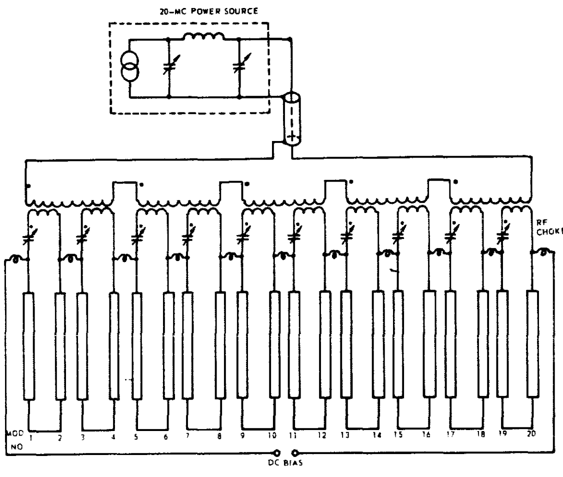


Figure 17. Diagram of Excitation Scheme.

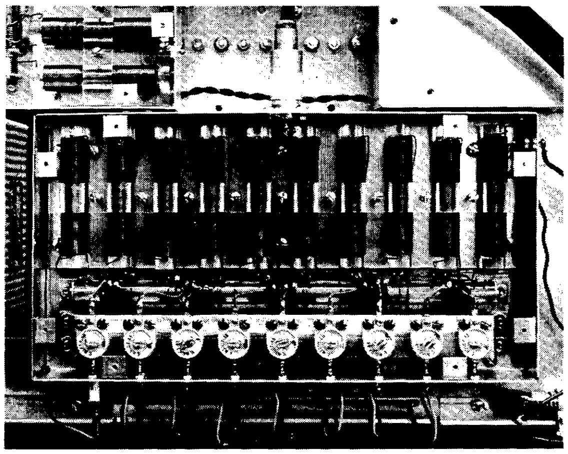


Figure 18. Pump Distribution Transformer.

given primary. All primaries are connected in series and driven from the power output stage through matching cable. All parametrons are connected in series for dc-bias. The various secondary circuits are isolated by rf chokes. The dc-bias is blocked from the transformer secondary by the tuning capacitors. The effect of this arrangement is that groups of parametron modules form a balanced network that is symmetrical to ground, minimizing the loading effects due to complex logic wiring. Ground loops are, of course, minimized by completely floating circuitry. Figure 18 is a photograph of one such distribution transformer.

VIII. SYSTEM EXPERIMENTS

A number of small systems, utilizing the 12-parametron modules, have been constructed and operated. One, a 6-bit shift register capable of automatic or manual circulation, is typical. The actual register is shown in Figure 19. Note

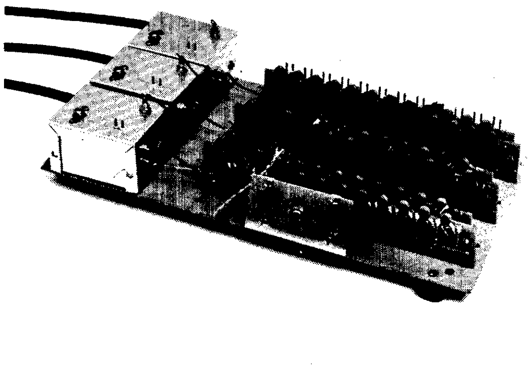
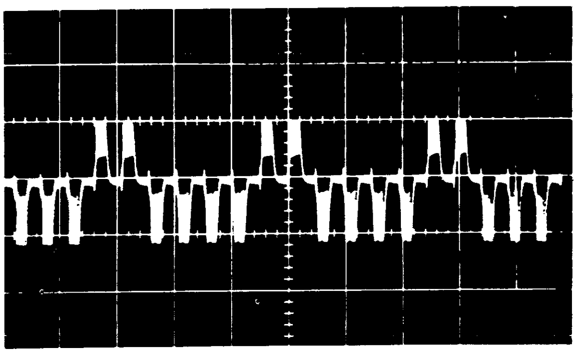


Figure 19. Six-Bit Shift Register.



SCALE

HORIZONTAL: 10 μs/div

VERTICAL : 2 ma/div

Figure 20. Six-Bit Shift Register Circulating Two "Ones".

the simplicity of the logic wiring which is basically point-to-point.

Also included in the register are a number of constant parametrons and input-output parametrons to allow simplified observation of the state of the register for laboratory studies. Waveforms of a 110000 pattern circulated at a 200-kc rate are shown in Figure 20. The blanking technique described earlier has been used for the display, making determination of the one and zero states quite simple.

A much larger network has been tested to allow demonstration of the worst-case fan-out capability. As discussed previously, the logical specifications for this parametron are a fan-in of 7 (majority gate of 7 inputs) and a fan-out of 10. The logical diagram of a worst-case network is shown in Figure 21. Basically, it consists of complementing flip-flop, parametrons P1, P2, P3, which form an alternating 1010 pattern. This output is used to lock parametron P4 which, in turn, drives the logical fan, P5 through P16. Each one of these latter parametrons in turn drives 12 more parametrons. It may be seen that 144 parametrons are operating in Beat I in the phase state opposite that of parametron P4. This delivers jump-coupling signals to P4 in phase opposition to the input signal from the complementing flip-flop.

Figure 22 is a photograph of a large array of parametrons, some of which are connected in the worst-case logic arrangement.

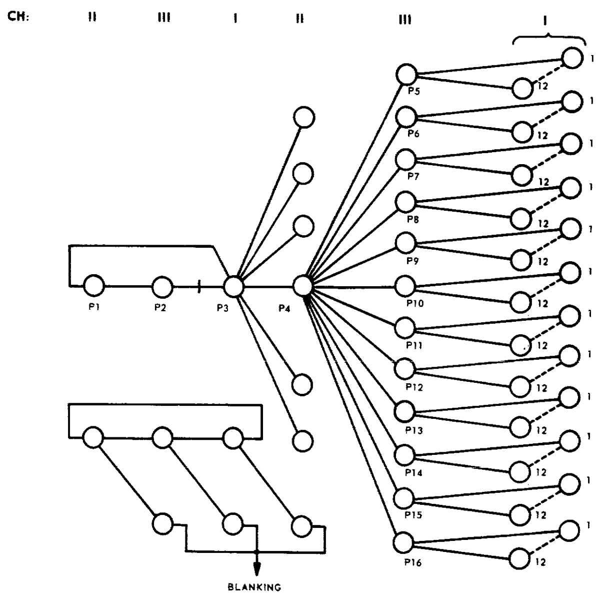


Figure 21. Worst Case Fan-Out Network.

IX. CONCLUSIONS

The preceding discussions and data have presented characteristics of a highly practical, cylindrical thin-film parametron. In great measure, the practicability of this device is due to the construction, where thin magnetic film has been electroplated on a wire substrate. This technique allows continuous high-yield processing from a simple manufacturing facility, reducing tooling costs for production of the device. The use of the substrate wire itself as the pump conductor allows maximum coupling between the pump field and the film, providing a highly efficient pumping mechanism and reducing the pump power required for parametric oscillation. The assembly of parametrons into a basic module of 12 units allows one length of plated wire to serve a number of parametrons. This reduces the assembly cost of parametron modules.

The module design has been established; component tolerances are understood and have been shown to be practical. The module design is highly amenable to automated production techniques. Since the parametron is suitable for all logical as well as storage functions within the parametron computer system, it is a true universal logic element.

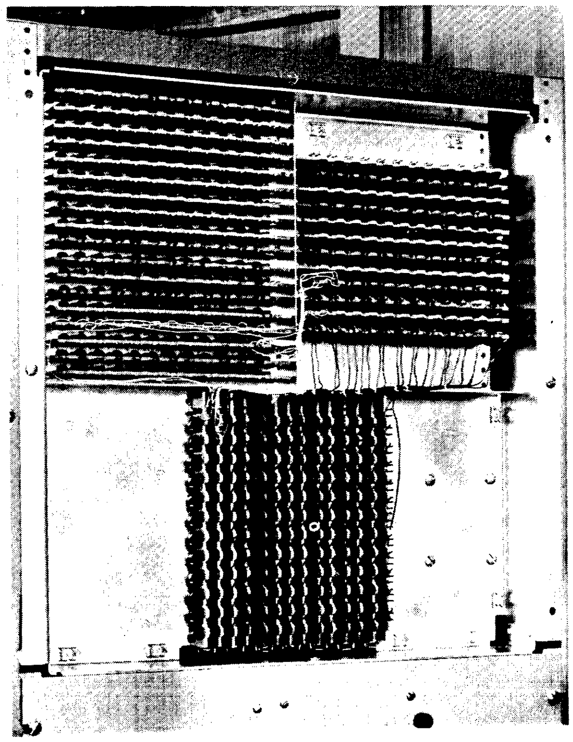


Figure 22. Module Tray.

Since the plated film characteristics are highly uniform and since saturation pumping is used as well, a high fan-in capability is achieved because of the highly uniform subharmonic output. The logical fan-in of 7 is obtained for standard pilot line units. This results in an extremely powerful parametron from the logical designer's standpoint.

The ability to construct and operate large systems has been achieved with simple and practical pump distributing systems. Power requirements of the parametron are such that simple vacuum tube circuits of moderate cost are suitable for pump distribution. In the very near future, high power, high frequency transistor circuits will be competitive in price with existing vacuum tube circuitry at these frequencies.

XI. DEFINITIONS[10]

A *phase-locked subharmonic oscillator* is a parametrically excited resonant system with n stable states of phase where n is the ratio of pump frequency to subharmonic frequency. The subharmonic frequency of the described PLSO's (parametrons) is ½ of the pump frequency. Thus, two stable states of phase are obtained, corresponding to "1" and "0" in a logic system. The subharmonic oscillation can be phase-locked by injecting a seed current of the desired phase into the tank circuit of the PLSO.

The *seed current* (input current) is a control current (small compared to the circulating tank current it controls) at the subharmonic frequency and is derived from the output of a controlling parametron and injected into the tank circuit of a controlled parametron which is excited by the following beat of the three beat excitation system.

The *three beat excitation system* or *pump supply* provides three channels of square wave modulated rf pump currents of sufficient amplitude to excite (pump) a given number of parametrons. If T is the period of the square wave modulation the turn-on times of succeeding beats are staggered by $\frac{T}{3}$ so that only two of the three channels are operating simultaneously. This excitation scheme assures unidirectional information flow in the parametron majority logic system.

Majority logic is based upon the comparison of an odd number of input propositions. In parametron systems the phases of the seed currents are compared. The vector sum of the phase-states of the seed currents determines the phase of the output current of the controlled parametron.

X. ACKNOWLEDGEMENTS

A project of this scope is obviously the work of a number of people. In particular, we would like to acknowledge the efforts of Mr. R. Sloppy in many of the experimental aspects of the work; of Mr. L. Douglas for preparing the large amounts of plated wire to our specifications; of Mrs. J. Greenwell and Miss M. Roberts for their skillful assembly of modules and system wiring.

Discussions with Japanese scientists and engineers engaged in parametron research, in particular Dr. E. Goto of the University of Tokyo, Mr. Y. Hata of TDK, and Mr. K. Mori of Kanematsu, greatly added to our understanding of this new art.

XII. REFERENCES

1. E. GOTO, "The Parametron, A Digital Computing Element Which Utilizes Parametric Oscillation", *Proc. IRE,* Vol. 47, No. 8 (August 1959).
2. J. v. NEUMANN, U. S. Patent No. 2,815,488.
3. A. V. POHM, A. A. READ, R. M. STEWART, JR., and R. F. SCHAUER, "High Frequency Magnetic Film Parametrons for Computing Logic", *Proc. of NEC,* pp. 202-214 (1959).
4. R. LINDAMAN, "A Theorem for Deriving Majority Logic Networks within an Augmented Boolean Algebra", *IRE Trans. on Electronic Computers,* Vol. EC-9, pp. 338-342 (September 1960).
5. M. COHN and R. LINDAMAN, "Axiomatic Majority Decision Logic", *IRE Trans. on Electronic Computers,* Vol. EC-10, pp. 17-21 (March 1961).
6. I. W. WOLF and V. P. MCCONNEL, "Nickel-Iron Alloy Electrodeposits for Magnetic Shielding", *Forty-Third Proc. Am. Electroplaters Society,* Vol. 1-4, p. 215 (1956).
7. G. F. SCHRADER, "M-H Loop Tracer for Circumferential Fields", *The Review of Scientific Instruments,* Vol. 32, p. 429, No. 4 (April 1961).
8. V. K. RANDERY, "Parametron Simulation on an Analog Computer", to be published.
9. H. TAKAHASI and KIYASU-ZEN'ITI, Editors, "Parametron", Parametron Institute, Tokyo, Japan (1960), pp. 139-144.

SINGLE CAPSTAN TAPE MEMORY

Mr. R. A. Kleist, Mr. M. A. Lewis, and Dr. B. C. Wang
Ampex Computer Products Company
9937 West Jefferson Boulevard
Culver City, California

1 INTRODUCTION

Digital tape memories, because of their large capacity and their ability to store data on removable reels at an extremely low cost per bit, are a powerful component in electronic data processing systems. A typical capacity is 10^8 to 10^9 bits, typical cost is \$3.6 $\times$ 10^{-5} cents per bit for a reel of tape and 1.25 $\times$ 10^{-2} cents per bit for a tape memory equipped with one reel of tape.

The digital tape memory will need to keep pace with demands of the broadening spectrum of cost and performance for new computing systems, providing cost consistent with each performance requirement, and showing increases in data reliability and machine maintainability at all performance levels.

The heart of a digital tape transport is the tape drive, consisting of a tape drive mechanism and tape guiding elements. The sole function of the tape drive is to move tape in a forward or reverse direction at a specified speed, or to keep the tape at a standstill, in response to appropriate commands. Each change of speed must be accompanied by controlled start and stop distances accomplished in a given time. The tape guiding elements constrain the tape along its prescribed path with a minimum of skewed motion and physical displacement from prescribed recording track locations. Performance is determined by:

1. The maximum tape velocity (a limitation on peak data transfer rate),
2. the speed with which the tape drive can respond to commands, i.e., start/stop times, as well as any further restrictions on program sequence (a limitation on effective throughput rate),
3. the nominal value and accuracy to which start/stop distances can be controlled (a limitation on the inter-record gap and, therefore, effective throughput rate), and
4. the accuracy to which tape can be guided and tape speed can be maintained (a limitation on bit density and, therefore, peak data transfer rate).

For performance parameters that determine the capability of a particular tape drive, the ultimate design criteria are:

1. High data reliability as evidenced by low data error rates and long tape life, and
2. high machine reliability as evidenced by long MTBF, minimum down time, and minimum periodic service requirements.

All present day digital tape drives have start/stop characteristics which are determined by sliding frictional properties of tape and/or tape guiding and driving elements. These properties can only be controlled within wide limits, giving a range of 25% to 50% in start/stop distances, and placing a limitation on the minimum inter-record gap. This, and the complexity of required mechanisms, limit the inter-record gap as well as data and machine reliability.

This paper describes a new development in digital tape drives and notes its product application in a low cost 556 bpi 20 KC peak character rate tape memory. The tape drive is a significant advancement beyond present digital tape drives, offering distinct advantages of gentle tape handling, precisely controlled start/stop characteristics, and mechanism simplicity which yields increased machine reliability. The tape drive provides the capability to read and write at 36 inches per second tape velocity, as well as providing high speed rewind. Start/stop characteristics yield a ¾ inch inter-record gap.

This new tape drive utilizes a single capstan and low friction tape path. The tape is held in contact with the capstan at all times by uniform tension derived from vacuum columns. Thus, tape motion over the head directly follows that of the capstan surface regardless of wide changes in friction properties of the tape or mechanism. It is only required that the static friction between tape and capstan be above a minimum figure. The capstan is servo driven to give a uniform and closely controlled acceleration or deceleration to the desired velocity; this is believed to be the unique contribution.

There is no "dead time" and no tape velocity overshoot during the start/stop transient, allowing the full start/stop time to be utilized for uniform, and therefore gentle, tape acceleration. Fig. 1.1 shows start characteristics of the single capstan drive versus a typical present day tape drive. Start/stop characteristics and nominal tape velocity can be held within ± 5% regardless of normal environmental and line voltage and frequency variations.

The tape path is otherwise comprised of vacuum storage columns, and four guiding elements, with no mechanical adjustments required. Tape threading is simple and high speed rewind is accomplished by capstan drive with reel servo control, giving a uniformly controlled tension regardless of tape velocity. The oxide side of the tape is in sliding contact only with the read/write head.

2 SINGLE CAPSTAN TAPE DRIVE MECHANICS

2.1 *Single Capstan Drive Concept*

Considering geometry of the single capstan drive, the tape is wrapped over the circumference of the capstan to maintain contact at all times. This is shown in the layout of the TM-7 tape transport, Figure 2-1. The single capstan is driven directly by a servo-controlled high

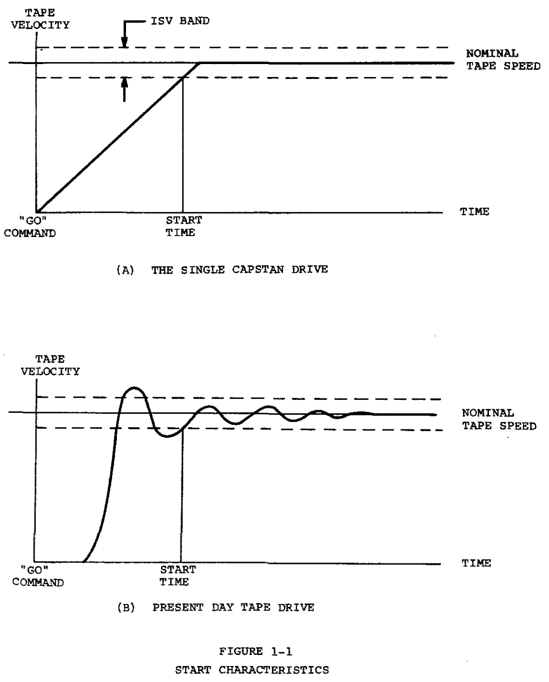


Figure 1–1. Start characteristics.

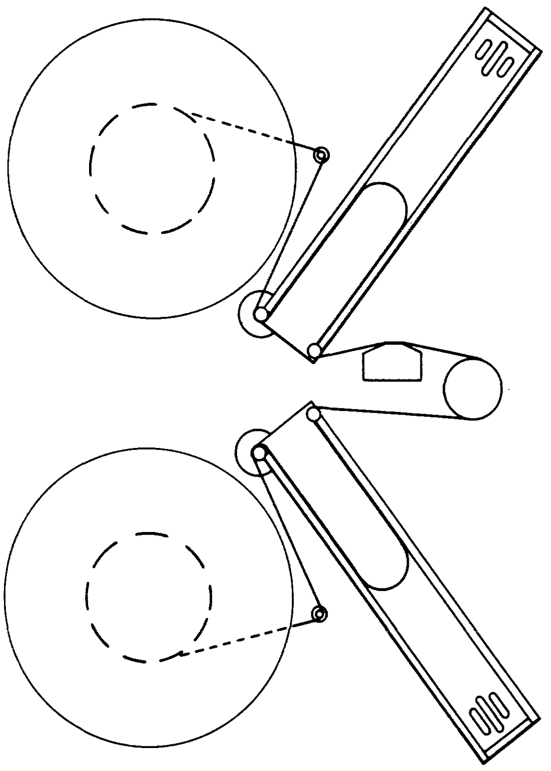


Figure 2–1. TM–7 Transport Layout.

torque-to-inertia-ratio DC motor. The driving force on the tape is provided by static frictional force developed between the tape and the capstan surface. This frictional force is produced by equal tape tension on each side of the capstan. These tensions are produced by vacuum pressure in the storage chambers. It should be pointed out, that although the tape driving force is derived from friction, the magnitude of the force that can be developed before slipping between tape and capstan occurs is always much greater than the required driving force. Therefore, there is no tape slippage over the capstan under any operating condition. Furthermore, the driving force is distributed over a large capstan surface area, resulting in low and uniform shearing stresses on the contacting surfaces. Equal tape tension on both sides of the capstan forms a self-balancing force system; thus the magnitude of nominal tape tension imposes no additional load on the capstan drive. The single capstan has the following distinct advantages:

1. Distributed driving force produces gentle tape handling and low tape wear.
2. Uniform tension across the tape minimizes skewed tape motion and eliminates the need for precise and periodic machine adjustment.

Considering driving of the capstan; start, stop, and reverse directions are accomplished by a servo-controlled current to the motor as detailed in Section 3. The magnitude and shape of the current waveform determine the start/stop characteristics. In order to attain gentle tape handling, full use is made of the start time, gently to accelerate tape to its terminal velocity. For a given start time and terminal velocity, a constant acceleration produces the lowest peak tape tension. Constant deceleration is likewise used for stopping the tape. During stop, brush friction in the DC motor is adequate to prevent tape from creeping. Advantages of this approach are:

1. Mechanism simplicity,
2. a consistent and readily selected start/stop time and terminal velocity, and
3. absence of high impact force resulting in quiet operation and mechanism life.

Since there are no impact forces on the drive elements, precise capstan and guide alignments are maintained over long periods of operation.

Starting and stopping of the tape are accomplished independently of friction in the tape path. Therefore, the single capstan drive permits the use of a low friction tape path, which reduces the required driving force. Also, tape tension and its variation are reduced, contributing to gentle tape handling. Tape guiding is accomplished by low-friction roller guides at the chamber exits.

2.2 *Single Capstan Drive Design Considerations*

In a digital tape drive design, the major dynamics problem deals with capstan and tape motion. The ultimate performance objective is to have tape moving over the read/write head with constant speed and minimum skewed motion along its prescribed path. This section outlines the key technical considerations and their relation to transport performance with particular reference to the Ampex TM-7 transport design; more detailed analysis conducted in the course of the development is omitted. Emphasis is placed on findings germane to the topic.

1. *Torsional Oscillation of Capstan.* The capstan motor shaft assembly can be considered as a spring-mass system. The system will oscillate when subjected to transient torques applied on the motor armature during start and stop. Since the capstan drives tape without slippage, capstan oscillation will produce tape vibration, and hence instantaneous speed variation over the read/write head. The amplitude of capstan oscillation depends on the magnitude and period of the applied torque, the amount of damping in the system, capstan and armature inertia, and motor shaft stiffness. The amplitude of oscillation is small when the period of applied pulse is made much longer than the fundamental period of oscillation of the capstan-motor shaft assembly. Increasing the damping can also reduce the magnitude of oscillation; however, damping is often difficult to implement and increases the input power and heat dissipa-

tion. In the TM-7 transport, the ratio of start period to the fundamental period of oscillation was approximately 40:1, and oscillation was rendered negligible by the capstan servo compensation.

2. *Tape Vibration.* Tape vibration set up by the start and stop transients is a major concern during write and read operations, especially in a low-friction tape path since vibrations are readily propagated. The two types of vibration important to digital recording are the longitudinal (in the direction of the tape path) and transverse (normal to the tape path). The longitudinal vibration induced by the start transients manifests itself as a spurious velocity variation of the tape at the head, whereas transverse vibration produces variation in the head-to-tape spacing, resulting in read and write signal fluctuation.

Longitudinal Vibration and Rigid-Body Dynamics. In the exact analysis of tape motion the tape is considered as an elastic medium with distributed mass along its length. This analysis enables calculation of vibrational characteristics of tape when it is subjected to a transient acceleration pulse. When a length of tape is being slowly accelerated to its terminal velocity, the tape behaves essentially like a rigid body in which every particle moves with the same velocity. In order to determine the condition under which the rigid-body analysis becomes valid, the solution to the one-dimensional longitudinal wave equation in terms of force and velocity was obtained for an acceleration pulse of variable duration. The results of this analysis show that if the ratio of the acceleration period to the wave propagation period is greater than 10, the tape behaves essentially as a rigid body. In the case of the TM–7, the acceleration period is 10 ms, and the wave propagation time is approximately .125 ms, giving a ratio of 80.

Transverse Vibration. The transverse vibration of tape is of concern when program rates are on the order of the natural frequency of vibration. The natural frequency of the fundamental mode of transverse vibration of a moving tape is given by the following equation.[1]

$$f = \frac{1}{2L} \sqrt{\frac{T}{\rho}} \left(1 - \frac{\rho V_0^2}{T}\right) \quad (2\text{-}1)$$

where

f = natural frequency, cps

T = tape tension, pounds

ρ = mass per unit length of tape, lb-sec^2/in^2

L = length of unsupported tape, inches

V_o = tape velocity, ips.

To minimize the effects of transverse vibration, the natural frequency should fall well above the transport frequencies due to start/stop program, pneumatic, and brush noises. The predominant excitation is given by the program rate. Therefore, the natural frequency should ideally be an order of magnitude higher than the maximum program rate. The above equation shows that for a high natural frequency the unsupported length should be short, and the tape tension should be such that any undesirable effects of transverse vibration are eliminated. The determination of such a tape tension can be accomplished through experimental study of the effect of vibration on the amplitude of a read signal for different values of tape tension when the transport is being operated at its maximum program rate. In the TM-7 transport, the maximum program rate is 50 cps and the natural frequency of the fundamental mode of vibration was approximately 350 cps at 0.4 pounds tape tension; no detectable effect of vibration was noted on the read and write operations.

3. *Condition of No Tape Slippage Over the Capstan.* The single capstan drive, the driving force transmitted to the tape is derived from friction between the tape and the capstan, and there is no relative motion between the tape and the capstan at any time. The no-slip condition can be met when the force that can be developed before slipping occurs is always greater than the required driving force. At the

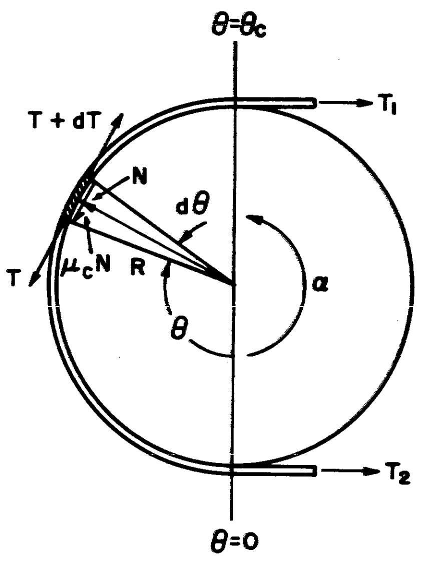


Figure 2–2. Friction Force over Capstan.

point of impending slip during capstan acceleration for the TM-7, the relation between the tensions, T_1 and T_2 (Figure 2-2) is given by the following equation:

$$\frac{T_1}{T_2} = e^{\mu_c \theta_c} \qquad (2\text{-}2)$$

where

μ_c = coefficient of friction between and capstan surface,

θ_c = wrap angle over capstan, radians.

For a given value of T_2, the maximum driving force, $(F_d)_{max}$, that can be developed before slipping occurs is given by

$$(F_d)_{max} = T_1 - T_2 = T_2\,(e^{\mu_c \theta_c} - 1) \qquad (2\text{-}3)$$

In the single capstan drive, the tape is allowed to slip under abnormal force imposed along the tape path. The slip should cease as soon as the abnormal force is removed (this is a protection against tape damage). Under this condition, the coefficient of friction should be the dynamic value at the speed of slipping. Since the dynamic value is generally smaller than the static value, μ_c, in equation (2-3), should be the lowest value, probably occurring at the highest tape speed during re-

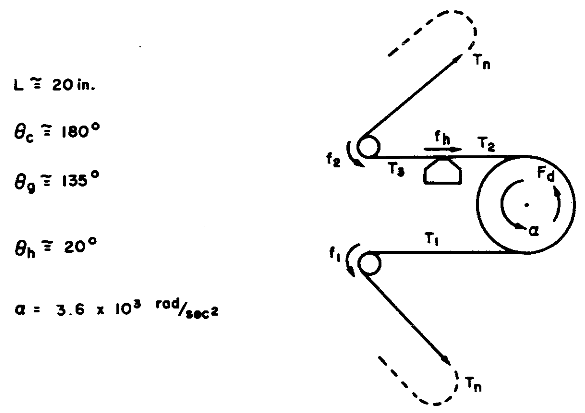


Figure 2–3. Force Distribution in the TM–7 Tape Path.

wind. $(F_d)_{max}$ based on this value should represent a conservative value. The tape tension T_2 is a function of nominal tension T_n set by the vacuum pressure in the storage chamber, and of force distribution along the tape path. As an example, the force distribution along the TM-7 tape path is shown in Figure 2-3. f_1 and f_2 are given mainly by the forces necessary to accelerate and decelerate the rollers to their terminal velocity during start and stop.

4. *Dynamic Skew Analysis.* In multi-track digital recording, one bit is written for each track across the tape; the line of bits is called a frame. Ideally, all bits in the same frame should be in line and perpendicular to the tape path so that they can be read simultaneously as the tape is passed at a given velocity over a fixed read head which has heads located in each track. For many reasons, this ideal condition is not realized. One bit will arrive at a read head slightly earlier or later than the others. Their time difference is called the Inter-Channel Time Displacement (ITD). ITD is caused by static skew (which can be corrected by a deskew register in the data electronics), and dynamic skew due to time-variant tape motion. In the TM-7 transport, the major excitation of dynamic skew is capstan runout. The magnitude of dynamic skew is governed mainly by the dimensional tolerence between the guides and the tape.

From geometrical considerations of the TM-7 tape path and these dimensional tolerances, ± 153 μinches of dynamic skew may be anticipated when reading a tape written with no skew. Therefore, the maximum skew, considering effects of both writing and reading, could be ± 306 μinches for the minimum tape width of 0.496 inches for a ½ inch tape. Experimental results justify neglecting other possible causes of dynamic skew.

3 SINGLE CAPSTAN SERVO

As described in Section 2, the single capstan is mounted directly on the shaft of a DC motor. The motor is controlled by an electronic servo. This section discusses the requirements for the capstan drive, the design of the electronic servo, and the way in which the objectives were met.

3.1 *Capstan Drive Requirements*

The capstan drive should accomplish the following:

1. On receipt of a start command, accelerate the tape to within the ISV specification during a specified tape displacement. This requirement, when considered with the acceleration characteristics and nominal tape velocity, determines start time.
2. Having achieved its nominal velocity, the tape must be constrained within the specified ISV limits in the presence of variations of load torque and other disturbances.
3. While the tape is in motion, the drive system must respond to a stop command to decelerate the tape to a standstill during a specified tape displacement.
4. The capstan drive must be capable of reversing the direction of tape motion upon command.

Note that the requirements of acceleration and deceleration are defined in terms of a start and stop distance, respectively, as necessary to meet inter-record gap requirements. Although acceleration and deceleration are derived from distances, for any given acceleration characteristic these may be translated into a start and stop time specification. For example, with constant acceleration the start time T_s is given by

$$T_s = \sqrt{\left|\frac{2D_s}{a}\right|} \quad \text{or} \quad T_s \doteq \frac{2D_s}{V} \qquad (3\text{-}1)$$

where D_s is the required start distance, and a is the constant acceleration. The final tape velocity will be

$$V = \sqrt{2aD_s} = aT_s. \qquad (3\text{-}2)$$

Typical control command requirements are that the system operate from a two-wire input, one wire being "run-stop," and the other being "forward-reverse." These commands are generally in the form of step voltage changes (logic levels).

3.2 *Servo Motor Drive Considerations*

For a chosen capstan geometry, it is possible to optimize the capstan inertia, in terms of the known armature inertia, to yield a maximum ratio of circumferential capstan acceleration to input energy. Limitations on the capstan geometry are imposed by considerations of rigidity. Thus, the optimization should take account also of changing geometry against radius. In any event, it is possible to arrive at an optimum figure for capstan radius and inertia which gives a high acceleration to energy ratio, and satisfactory rigidity of capstan structure.

Having found the capstan inertia, and taking account of the tachometer inertia, the total inertia to be accelerated is now known. Using the capstan radius, the angular velocity for a given circumferential capstan velocity is calculated. For a constant acceleration, the start time determines the angular acceleration, and this in turn determines the torque required to accelerate the inertia. To this torque must be added the friction torque opposing motion (friction torque is almost entirely due to motor brush friction, since tape path friction was designed to be negligible). The sum of these torques is required to accelerate the tape to its final velocity in the required start time. The current necessary for producing the required torque can be calculated using the torque-current characteristic of the motor. The voltage input required to maintain this current must increase as the speed increases to take account

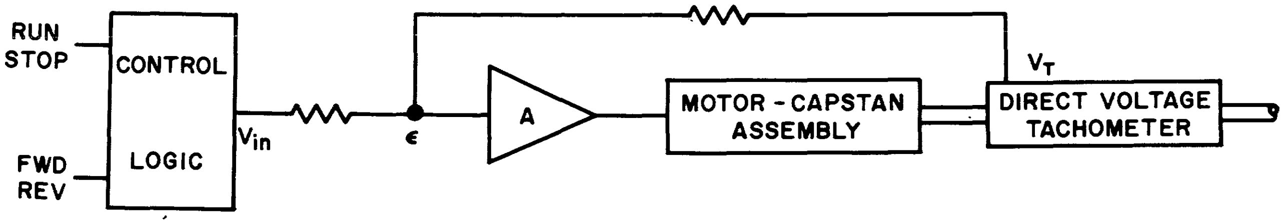


Figure 3–1. Block Diagram of Capstan Servo System and Control Logic.

of the motor back emf. The maximum required voltage V_M is given by

$$V_M = I_A R_A + e_{gM}$$

where

I_A is the accelerating current

R_A is the armature resistance

e_{gM} is the back emf generated at the final veloctiy of the motor,

and where the motor is chosen to have negligible armature inductance.

The motor drive requirements are now fixed. Worst case motor drive requirements are determined by giving the parameters maximum and minimum values, whichever are appropriate.

The type of motor best suited to this application is the printed armature motor. It has low armature inertia, high torque-per-unit current, low armature resistance, and negligible inductance.

3.3 *Electronic Servo*

The electronic servo consists of a high gain power amplifier capable of supplying the voltage and current calculated for worst case motor drive. The feedback signal is taken from a DC tachometer mounted on the motor shaft. The solid-state power amplifier is capable of establishing its full output voltage, and its full output current in tens of microseconds into the load presented by the motor. This time is negligible in comparison with the start time of a typical tape drive.

A block diagram of the electronic servo is shown in Figure 3-1.

The control logic interprets commands which are in the form of step changes of voltage, and converts them to a reference voltage V_{in} at the input of the servo. V_{in} takes the form of a ramp which increases or decreases at a predetermined rate to a predetermined final value. This is the desired tape velocity versus time characteristic. The time of the ramp is the desired start/stop time, and the final value of the ramp voltage is the reference for steady running velocity. Figure 3-2 is a diagram of typical input commands and the corresponding velocity reference.

Since the servo amplifier is very much faster than the rate of change of the input voltage, and since the power output is capable of accelerating the motor at the desired rate, the relation between the input voltage, V_{in}, and the tachometer voltage, V_T, will be

$$\frac{V_T}{V_{in}} = \frac{-Z_2}{Z_1}$$

assuming amplifier gain, A, is high. This relation holds both during the acceleration period and the steady run period even in the presence of line voltage and frequency variations and changing environmental conditions. Thus, the output velocity, which is linearly related to V_T, will follow the input command very closely, independent of friction load variations. If the friction load increases, the output current of the amplifier also increases, under feedback action, to provide the extra torque to overcome the increased friction. A similar argument applies for reduced friction load.

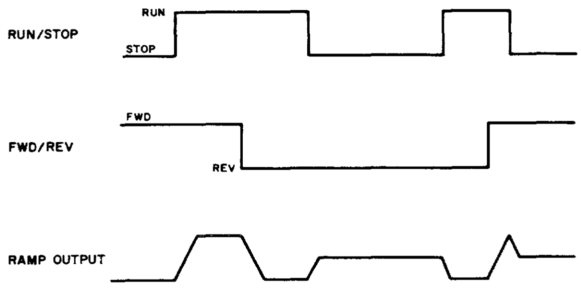


Figure 3–2. Inputs and Output of Ramp Generator.

During the start period, the motor current jumps to a constant value, and as soon as the final speed is reached, the current reduces to the value required to produce the torque which overcomes the opposing friction. When the command to stop is received, the input to the amplifier generates a downward ramp terminating at zero. Since the rate of the ramp is the same for either start or stop, start time is equal to stop time.

One vital feature of the servo amplifier is that it has a defined dead band centered at zero input voltage. The dead band is necessary in order that the motor does not creep (due to amplifier drift) when a stop command is present. The dead band is large enough to cover anticipated worst case amplifier drift. As noted in Section 2, brush friction prevents tape motion in the absence of servo motor drive.

3.4 *Servo Stability*

The printed circuit motor, driven from a zero source impedance, has the frequency response shown in Figure 3-3. The lower break frequency, f_1, is determined by the armature resistance, the torque-per-unit current, the back emf, and the armature inertia. The upper break frequency, f_2, is determined by the armature resistance and the armature inductance. If the source impedance is raised, the effective armature resistance is increased, f_1 is lowered, and f_2 is increased in proportion to the increase in effective armature resistance. A high source impedance can be simulated by employing feedback around the servo amplifier which is proportional to the output current. In this way, the output current, rather than the output voltage, is controlled by the error signal ϵ.

When a capstan and a tachometer are added to the motor shaft, the upper break frequency

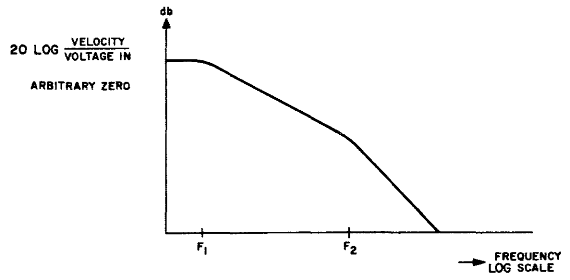


Figure 3–3. Motor Frequency Response.

is unchanged, but the lower frequency is further reduced by the increased inertia. In addition to these effects, the mechanical system comprising armature inertia, capstan inertia, tachometer inertia, and the torsional rigidity of the two sections of connecting shaft form a high Q resonant system having two resonant frequencies. These resonances lie between f_1 and f_2. These frequencies would cause the servo to oscillate unless servo compensation is employed.

With the effect of the resonance removed, the servo is quite stable, showing no sign of instability either when following input commands or when adjusting to step changes in load disturbances.

4 AMPEX TM-7 DIGITAL TAPE TRANSPORT

4.1 *Performance Specification*

1. Tape: ½″ wide Mylar (either 1 mil or 1.5 mil base) on IBM or NAB 10½″ reels.
2. Tape Speed: 36 ips ±5% instantaneous speed variation.
3. Head: 7-channel IBM 729 compatible, 0.3″ read-write gap.
4. Bit Density: 200 or 556 bits per inch.
5. Start and Stop Distances: Compatible with IBM 729 inter-record gap. This implies a 10 milliseconds start/stop time.
6. Rewind Speed: Approximately 180 ips.

In addition to these quantitative specifications, primary design emphasis was laid on data and machine reliability.

4.2 *Description*

The TM-7 transport is shown in Figures 4-1 and 4-2. The photographs clearly show the simplicity of the machine layout and construction. The main plate is made of a single casting with integral air ducts for the vacuum system. The tape has its oxide in sliding contact only with the head assembly. Two constant width vacuum chambers are used for tape buffer storage. Low-inertia roller guides at the storage chamber exits to the capstan are a main factor in achieving low tape path friction and low tape

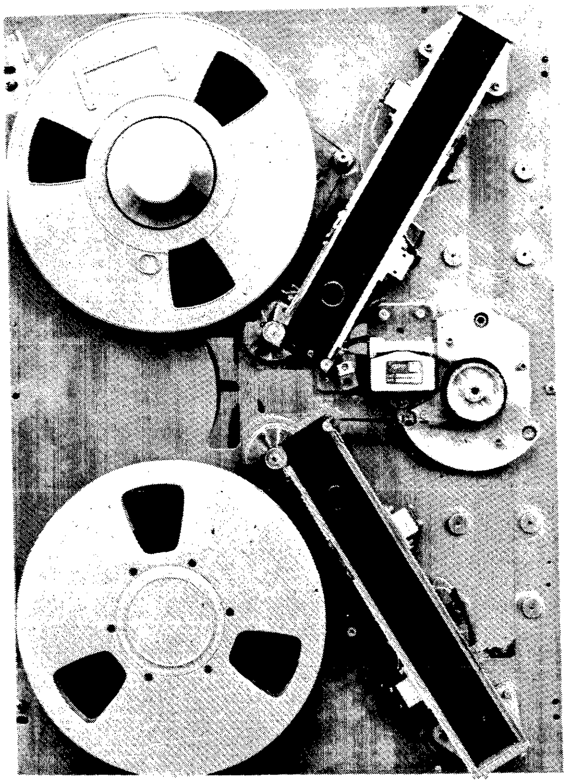


Figure 4–1. TM–7 Plate Model—Front View.

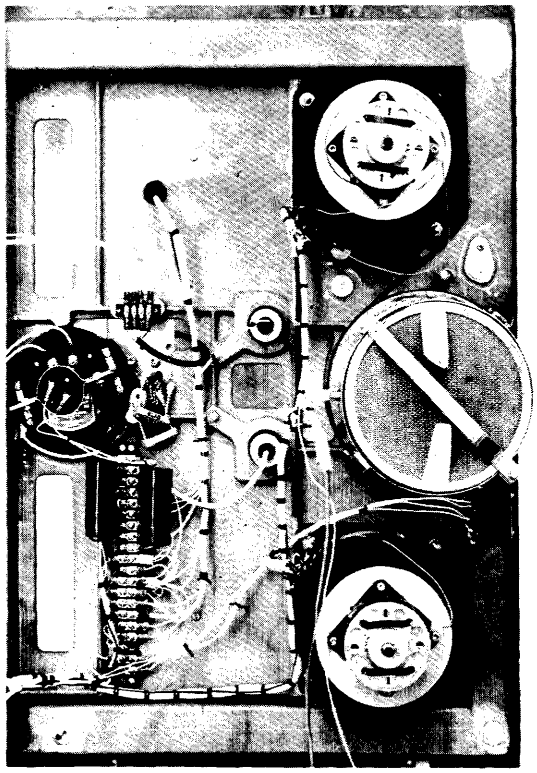


Figure 4–2. TM–7 Plate Model—Rear View.

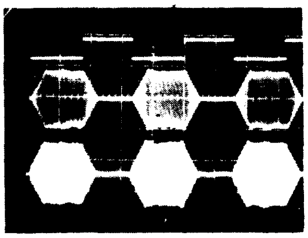
tension variation. The reel servo tachometers are located at the opposite chamber exits.

Tape is driven by a capstan and motor assembly which has an inertia of 0.0058 oz-in-sec^2. The tachometer inertia is negligible. The motor (Printed Motors Incorporated Model 368) has a torque-per-unit current of 3 oz-in per ampere, an armature resistance of 0.63 ohms, a back emf of 2.25 volts per 1000 rpm and an inductance of approximately 50 microhenries. These figures, together with the start time requirement, determine the output voltage and output current required of the servo amplifier. The capability of the amplifier is 12 volts at 12 amperes. Since the DC tachometer determines the magnitude of instantaneous speed variations, the tachometer is chosen to have ripple consistent with the desired performance. Control lines are run-stop, forward-reverse, and rewind, as determined by level sensitive inputs. Based on the start time of 10 milliseconds, the machine is capable of operating a start/stop sequence at a maximum repetition rate of 50 cps. It will operate a forward-reverse sequence at a maximum repetition rate of 25 cps. The machine will receive commands at higher rates in any sequence without damage to the tape or transport; a faster rate or incorrect sequence would simply prevent the machine from reaching the specified velocity before the next command. The rewind command will cause the machine to go to high speed reverse operation, stop at the load point, and give a "transport ready" signal when this sequence is complete.

The reel servo is a type in which the tape velocity to and from each reel is measured by a DC tachometer and compared with a reference velocity. This reference is the output of the capstan servo tachometer. Thus the reel servo is at all times given a precise reference of the velocity at which the tape is being driven. In addition to the velocity feedback, position feedback is obtained at two points in the vacuum chamber by optical sensing. Since the reference velocity is the capstan velocity, the reel servo is capable of operating over a very wide range of tape drive speeds. If the rate of change of capstan velocity is made slower, (the reel servo cannot accelerate as rapidly as the capstan servo), the machine will operate in the high speed rewind mode with the reel servo in operation. This feature also gives the potential of operating in a high speed search mode. The rate of change of capstan velocity is adjusted by changing the slope of the capstan drive ramp generator. The final capstan velocity is selected by changing a resistor value in the capstan servo. The advantage of rewind with the tape in the vacuum chamber is that controlled tension is maintained.

4.3 *Experimental Results*

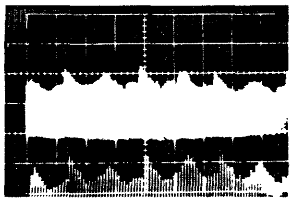
4.3.1 *Start/Stop Characteristics*. The shape of the start/stop velocity characteristic corresponds closely to the shape of the velocity reference, Figure 4-3. Also shown is the command signal and the read envelope representing the tape velocity. Figure 4-3 reveals the following significant results:



Sweep Rate:	20 ms/div.
Direction:	Forward 36 ips
Top Trace:	Command Signal
Middle Trace:	Capstan Velocity (AC Tachometer)
Bottom Trace:	Tape Velocity (Read Envelope)

Figure 4-3. Capstan and Tape Velocity Envelope, Unidirectional Program, 36 milliseconds on; 32 milliseconds off.

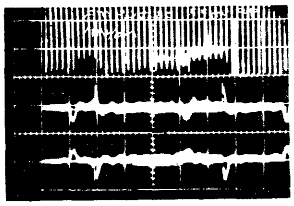
1. Coincidence of capstan motion, tape motion, and velocity reference. The fact that the velocity of tape over the head follows identically that of the capstan establishes the validity of treating tape as a rigid body.
2. The 10 ms start/stop time with constant acceleration has been accomplished by the capstan drive.
3. There is no significant dead time between the initiation of a start or stop command and the beginning of the acceleration period. The start time is fully used to gently accelerate the tape to its terminal velocity.
4. Absence of capstan velocity oscillations during start and stop indicates that torsional oscillation of the capstan-motor shaft assembly can be ignored. The repeatability and consistency of curves such as Figure 4-3 indicate the accomplishment of controlled start/stop characteristics.



Sweep Rate:	10 ms/cm
Tape Tension:	2 oz.
Upper Trace:	Read Envelope (Tape Velocity)
Lower Trace:	Capstan Speed; 36 ips ISV; 10%/cm

Figure 4-4. Tape and Capstan Velocity.

4.3.2 *Speed Variations.* The capstan and tape speed variation are shown in Figure 4-4. It is noted that the tape speed varied in the same manner as that of the capstan and the magnitude of the variation was about ± 5% at 36 ips. The cause of capstan speed variation was conclusively related to the brush noise of the DC tachometer used in the capstan servo loop as shown in Figure 4-5.

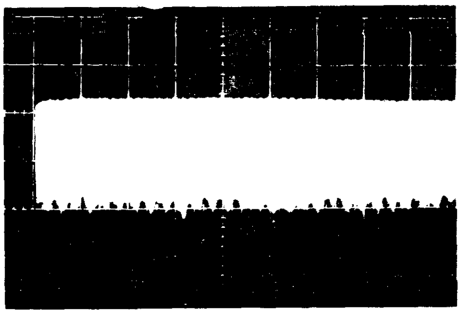


Sweep Rate:	5 ms/cm
Tape Tension:	3 oz.
Top Trace:	Capstan Speed: 36 ips, Forward ISV: 3.6%/cm
Middle Trace:	Feedback DC Tachometer (Servo-Tek) Output: 0.1 V/cm
Bottom Trace:	Capstan Motor Current, 2 amp/cm

Figure 4-5. Cause of Capstan Speed Variation.

4.3.3 *Dynamic Skew.* Dynamic skew measurements were made with the Ampex Tape Unit Dynamic Analyzer. This instrument measures magnitude of skew and its frequency components. By correlating frequencies generated in the transport, it is possible to determine individual causes of dynamic skew. Measurements were made during read-after-write, and read in both forward and reverse directions. Figures 4-6 and 4-7 show the typical records. The magnitudes of dynamic skew across the outer tracks (0.420″ apart) were: ± 63 micro-inches for read-after-write, ± 143 micro-inches for forward and reverse read. The capstan rotational frequency was observed to be the predominant skew frequency. The results are within the ± 306 micro-inches predicted in Section 3 on the basis of worst case dimensional tolerance between tape and guide widths.

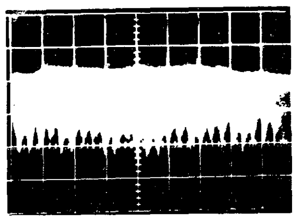
4.3.4 *Start/Stop Distance and Inter-Record Gap Computation.* Start and stop distances



READ-AFTER-WRITE AT 36 IPS, AMPEX 832 TAPE

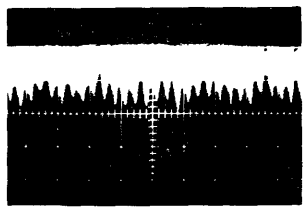
Tape Tension:	2.2 oz.
Sweep Rate:	0.1 sec/cm
ITD:	7 μsec/cm

Figure 4–6. Dynamic Skew (Plate Model Using Flanged Roller Guides, Ampex 832 Tape) Read-After-Write.



READING FORWARD AT 36 IPS

Tape Tension:	2.5 oz.
Sweep Rate:	0.5 sec/cm
ITD:	7 μsec/cm



READING REVERSE AT 36 IPS

Tape Tension:	2.2 oz.
Sweep Rate:	0.5 sec/cm
ITD:	7 μsec/cm

Figure 4–7. Dynamic Skew (Plate Model Using Flanged Roller Guides, Ampex 832 Tape).

were determined from the velocity-time relation. The inter-record gap distance and its tolerance were calculated from the following relationship, on the basis of TM-7 performance of ± 5% variation in acceleration and ± 5% in tape speed, the dimension of read/write gap at 0.300″ ± 0.005″.

IRG = read/write gap + stop delay distance + stop distance + start distance (4-1)

Based on these requirements, the maximum and minimum IRG and its tolerance band were calculated in the following table and compared with the IBM IRG.

	MAXI-MUM	MINI-MUM	TOLER-ANCE BAND
TM-7 IRG	0.809″	0.693″	0.116″
IBM 729 IRG	0.906″	0.687″	0.219″

It is seen from the above table that the TM-7 IRG adequately meets the requirement for IBM 729 compatibility. The tolerance band on the TM-7 IRG is about 30% of IBM IRG tolerance, through tighter control of start/stop characteristics.

5 AMPEX TM-7212 TAPE MEMORY

A typical application of the Ampex TM-7 Digital Transport is in the Ampex TM-7212 tape memory. It is a dual density (200/556 bpi) IBM 729 compatible system with a 20 KC peak character transfer rate. This recording format utilizes seven tracks of "OR clock" NRZI recording on a ½ inch tape width with a nominal ¾ inch inter-record gap. The system consists of four TM-7 tape transports with associated Control Electronics which control tape transports and the system power distribution, Data Electronics for writing and reading, and an electronic power supply. These components are functionally grouped in a "1 × 4" configuration wherein one set of data electronics is shared with four transports.

Each transport is equipped with front panel operator controls and indicators for local and remote control. When a transport is switched to the remote position, the Control Electronics select and operate the transport upon commands from the computer. An indication of the selected unit is provided on the front of the enclosure. On remote command, the Data Electronics will read or write on any one of the four transports which is selected with input or output data parallel by bit and serial by character at a rate controlled by the tape memory; any or all of the other transports can be in a rewind operation during this time.

Increased machine reliability follows directly from the inherent simplicity of the tape drive and the absence of impact forces.

Typical data reliability figures utilizing Ampex hard binder computer tape are:

(a) write check mode—maximum of one of 4×10^6 errors.

(b) read only operation—maximum of one of 2×10^8 temporary errors and one of 10^{11} permanent errors.

BIBLIOGRAPHY

1. "The Vibration of a String Having Uniform Motion Along its Length" by F. R. ARCHIBALD and A. G. EMSLIE, *Jour. Appl. Mech.*, Vol. 25, *No. 3*, 1958.

EVOLUTION OF DIGITAL MAGNETIC TAPE SYSTEMS FOR USE IN MILITARY ENVIRONMENTS

D. H. Tyrrell
CCIS70 Systems Office
USAELRDL
Fort Monmouth, New Jersey

D. J. Morrison
Computer Systems Section
Bureau of Ships, Navy Department
Washington, D. C.

J. J. Staller
Sylvania Electronic Systems—East
189 B Street
Needham Heights 94, Massachusetts

INTRODUCTION

The development of the electronic data processor has far outstripped similar developments in peripheral devices that feed, store, and receive data from the central processor. This imbalance has resulted in overall systems whose capability, performance, and size is primarily limited by the peripheral or ancillary devices. While a factor in commercial applications, these short-comings are particularly serious for equipment that must operate in military environments, under the control of military personnel. The Army and Navy, and their suppliers, have learned through experience that commercial or semi-militarized equipment will not operate reliably, nor can it be readily maintained under these conditions. These problems, when coupled with the severe size and weight limitations placed on transportable equipment, provided the impetus for this development program.

Since 1956, the Navy has been developing large-scale data processing systems for command and control applications. During this time the Navy has played an important role in advancing the state-of-the-art in this field, particularly in the area of militarized equipments.

This period of time has seen the shrinking of the central processor from a room-sized, vacuum tube machine to a table-top size, microelectronic unit. It has seen the reliability improve from a 30-hour mean time between failure (MTBF) to the 1500-hour MTBF currently being achieved in the CP-642A shipboard computer. It has not, however, seen comparable improvements in size and reliability of computer peripheral equipment.

Since the Navy is dependent on tape transports in most of its computer installations, it has been acutely aware of the need for highly reliable tape transport. The Navy has utilized automatic tape testing, ultrasonic tape cleaning, and special test equipment to improve the system "down-time" caused primarily by tape transport failures.

The U. S. Army has also been a large-scale user of data processing equipment. The variety of uses to which the computer systems have been placed such as stock control, intelligence evaluation, fire control, personnel management, etc., dictated the necessity for developing equipment capable of operating over a wide range of environmental conditions. The mobility and flexibility of the Atomic Age Army require lightweight, small, and easily maintainable equipment packages. The Army's experience in interim use of commercial or semi-militarized data processing equipment has served to pinpoint the need for using fully militarized equipment whenever there is a requirement for mobility or even a slight deviation from a commercial environment.

The Army began its militarized computer system development in 1958 with the Fieldata family of computers (MOBIDIC, BASICPAC, etc.). The peripheral equipment development commenced at the same time and has been directed towards a group of peripheral devices that are completely interchangeable within the family. The early portion of this program was fraught with many problems. Nevertheless, this experience formed the basis for the writing of a detailed description of Army requirements in a militarized tape transport.

As a result of this experience, in 1962 both the Army and Navy embarked on individual programs to improve the performance and ruggedness of tape transports.

Stimulated by the needs of the services for militarized peripheral equipment, demonstrated by their own experience as prime contractor for the Army's Mobile Digital Computer (MOBIDIC), Sylvania in 1960 undertook a company-sponsored research and development program to identify the peripheral equipment problem areas and to initiate programs to solve these problems. The initial study clearly indicated that the magnetic tape transport systems should be the initial point of attack. A program to establish the particular specifications and requirements, and to carry out research and development in significant technical areas was established. Among the major areas of initial study and development were the following:

a. Tape drive methods to meet military computer requirements.
b. Tape loop buffering and tape supply and take-up control.
c. Recording and playback techniques, systems, and circuits.
d. Data reliability and error detection and correction methods.
e. Capstan speed control and synchronizing methods.
f. Operational features of human and machine handling of the tape medium.
g. Mechanical and electronic packaging methods for sige and weight reduction.

In mid 1962, as a result of this development program and in response to requirements defined by the Army and the Navy, Sylvania was awarded contracts to deliver service test models to the two services.

GENERAL DESIGN OBJECTIVES

The general objectives defined by the Army and Navy specifications were quite similar except where each service had operating requirements peculiar to its own mission.

The major common design objectives were the following:

Improved Reliability and Performance by:

a. Elimination of human handling of the recording medium.
b. More positive machine control of the recording medium.
c. Major reduction in tape loading time (cartridge and automatic loading).
d. Major reduction of critical machine parts by simpler design.
e. Increase in effective data handling rates (less tape skew).

Improved Operational Simplicity and Program Flexibility by:

a. Designing foolproof tape loading and unloading procedure.
b. Increase flexibility through duplexed or time-shared elements.
c. Elimination of programming restrictions reflected by low dynamic machine response.

More Effective Design Utilization of Space by:

a. Reducing system inertia loads (storage reels and prime movers).

b. Reducing power and air-cooling requirements.
c. Increasing packaging density of electronic and mechanical components.

There were two major Navy design requirements not shared by the Army. First, the individual transport must be small enough to fit through a standard shipboard hatch. Second, the machine had to withstand, while in operation, the shipboard shock and vibration caused by high-energy underwater explosions close to the hull of the ship. Such a "near miss" explosion causes severe shock peaks and causes the ship structure to go into vibrational modes. The Navy requires that the equipment while operating, be tested on test equipment designed to simulate these conditions.

The Army specification has one requirement that is unique among requirements for digital tape transports. It has been determined that a data link channel is necessary for communications between computers. In a field situation, it is probable that this data link will have a bandwidth similar to a normal voice communications channel. To transmit over this narrow bandwidth channel the data rate must be reduced from its normal 45,000 characters per second to 300 characters per second. The magnetic tape transport was selected to provide this facility. The buffering equipment between the tape transport and the transmitting device has a 30-character memory. Thus, as a safety factor, transmission into and out of the buffer must be synchronous, within ± 12 characters per block of data. Data must be recorded at 300 characters per second and be readable at the higher rate and vice-versa; conversion of a transport to or from low-speed mode must take less than one hour.

Army and Navy Technical Requirements

Although procured under separate R & D contracts, the Army and Navy Tape Transports are similar in their operating characteristics. Table I summarizes these characteristics and shows both Army and Navy specifications.

OVERALL SYSTEM DESCRIPTION

The basic tape transport module without the cartridge dust and RFI cover is shown in Figure 1. The Army and Navy models are physically similar but have a number of operational differences. As illustrated, the tape is contained in a pre-loaded cartridge. The operator simply plugs the cartridge into the opening in the front panel and presses the tape load control. The balance of the loading and threading cycle is completely automatic under machine control including locating the tape at the beginning-of-tape marker. Total loading and threading time will not exceed 10 seconds. The cartridge can be unloaded and removed at any time in the computation cycle without rewinding the tape. Rewinding under controlled tension is provided by an off-line rewind device.

The physical characteristics of the basic tape transport module are:

a. Height: $21\frac{1}{4}$ inches
b. Width: $17\frac{1}{4}$ inches
c. Depth (overall): $22\frac{27}{32}$ inches
d. Weight: 340 pounds

The Tape cartridge physical characteristics are:

a. Height: $16\frac{5}{8}$ inches
b. Width: 1.660 inches
c. Depth: 8 inches
d. Weight (with tape): 6.6 pounds

When withdrawn on its slides, access to the major electronics elements is provided by the openings on the right side of the electronics package mounting door as indicated in Figure 1. Access to the other electronic packages, in-

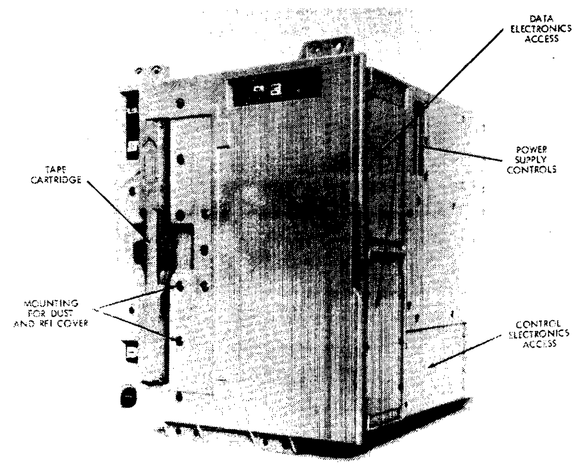


Figure 1. Basic Tape Transport Module.

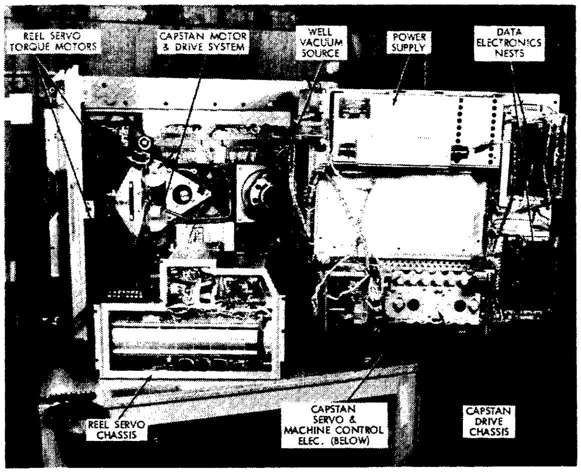


Figure 2. Basic Tape Transport Showing Details of Electronics.

terconnection wiring, and mechanical and electromechanical elements is provided by swinging the electronic package mounting door out of normal position as illustrated in Figure 2. The tape metering drive elements and maintenance control panels are accessible from the opposite side of the main deck.

Tape drive and rapid start-stop is achieved by the use of a pressure clamp air blanket coupling the tape to the counter-rotating capstan on command. Excess loop tape buffering is controlled by a 13-inch vacuum well with continuous photo-cell tape position sensing.

All mechanical motions associated with the automatic threading and loading are performed by pneumatic actuators under control of the rotary central control pneumatic valve. By performing the high-speed tape drive functions and all mechanical motions with air pressure, only a single air source is required rather than both vacuum and pressure sources as would be required with a vacuum capstan machine.

Tape transport modules can be arranged in different combinations to fit particular applications and data processing system requirements. Figure 3 and 4 illustrate two particular system arrangements being provided. Figure 3 represents a four-module Navy system with integral compressed air source, cooling air distribution system, power control panel, and logical tape transport selection and decoding matrix. The system is designed to operate during shipboard

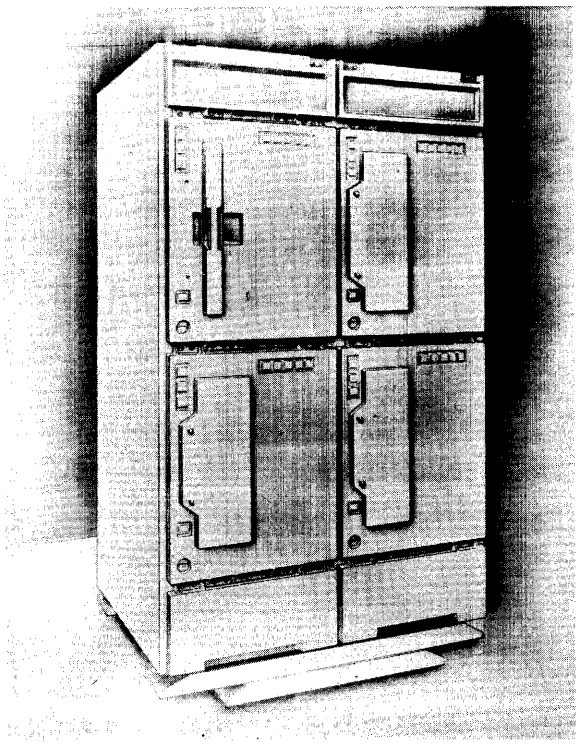


Figure 3. Navy Magnetic Tape Transport System.

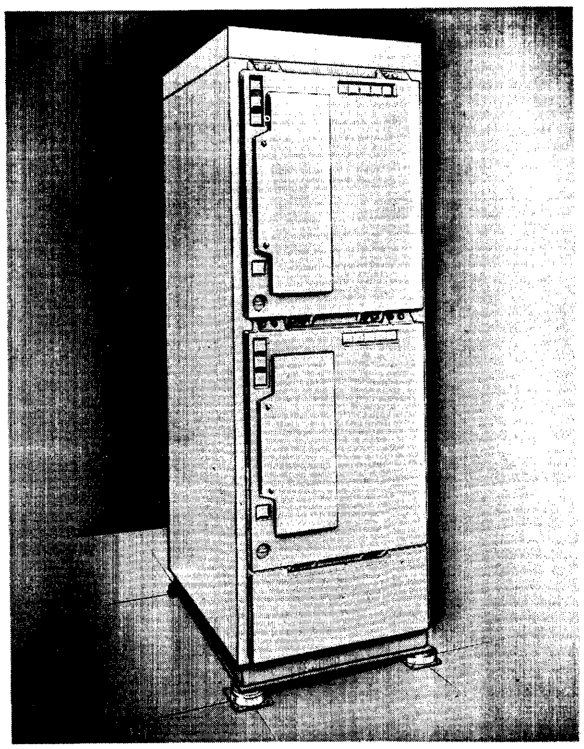


Figure 4. Army Magnetic Tape Transport System.

shock and vibration. The cabinet can be disassembled for loading through shipboard hatchways.

Figure 4 shows a two-module Army system with an integral air source. A structural rack adapter is provided to permit the system to be mounted in a 56⅛ inch vertical opening in a standard 19-inch panel mounting configuration. Cooling air pressure and volume is provided externally. This permits either mounting the tape system and adapter to standard 19-inch rack structures, built as integral with a shelter structure, or into a cabinet as shown in the illustration.

General Approaches to Meeting Specification Requirements

1. Error rate not to exceed one character error in 10^7 characters read-in Army system. This is achieved by the pneumatic handling of the tape, built-in single bit error correction and specialized read-write circuit and magnetic head design.
2. Data rate capability for the Army system of 45,000 and 300 character per second. At the 300 characters per second read mode, the number of characters read out cannot deviate from the number of cycles of the 300 cycle external synchronizing frequency during the same period by greater than ±12 characters in any one block (maximum block length to be 35,810 characters). This is accomplished by providing a 150/1 difference in tape drive speed with high speed (45,000 char/sec) at 100 inches per second, 450 bits inch and low speed (300 char/sec) at ⅔ inches per second, 450 bits per inch. At low speed read, the capstan drive speed is continuously controlled by a comparison of the data read from the tape and the synchronizing signal, increasing or decreasing the tape speed to stay within the ±12 characters tolerance. Special read head and circuit techniques are utilized to maintain proper signal-to-noise ratios at the low speed.
3. Capability for 45,000 or 90,000 characters per second operation or dual 45 KC redundant required by the Navy system. This is accomplished by providing a 16-channel system at 450 bits per inch, 100 inches per second. The redundant 45 KC recording can be utilized in an automatic error correcting system which is provided in the external Tape Control System.
4. Normal operation during shock and vibration. This is accomplished by the structural rigidity of the basic machine and cabinet, the non-critical adjustment design approach and low dynamic sensitivity of the drive system and the use of a tailored shock isolation system to reduce shock levels transmitted to the tape transports.
5. Operational simplicity. This is provided by the cartridge loading and automatic threading where the operator does not handle the tape and performs no intricate threading operations.
6. Reduced size. High density electronic and mechanical packaging are used to reduce volume. The vacuum well is reduced to about ⅓ of conventional length by reel-servo control techniques.
7. Reduced tape wear. The only contact between the tape oxide surface and any fixed or moving surface is at the magnetic head. In addition, the stresses on the tape introduced by high tensile and compressive forces associated with normal pinch roller drives are eliminated by the pressure clamp approach.
8. Improved maintainability. By the use of modular design, all electronic sub-units are readily replaceable. Test points and an internal maintenance control panel are provided. All mechanical sub-assemblies are readily accessible and minimum adjustments are required.

SIGNIFICANT TECHNICAL DEVELOPMENTS

A brief technical discussion of four of the more unusual technical developments associated with the tape transport is provided below.

Pneumatic Pressure Clamp Tape Drive Method

Tape is normally imparted its high-speed acceleration, deceleration, and running forces by the use of dual counter-rotating capstans.

The tape is maintained out of contact or in light contact with the capstans during non-drive. One of the following methods is normally utilized to marry the tape to the capstan causing rapid acceleration of the tape to the capstan surface speed.

a. Electromagnetically actuated pinch rollers.
b. Vacuum introduced into the capstan to draw the tape to the capstan.

   Or a single capstan in constant contact with the tape may be utilized.

After evaluating and testing models of the various approaches to rapid acceleration-deceleration of magnetic tape and equating them against the military environments, it was concluded that pneumatic drive techniques offered the best potential for the planned militarized application. Extensive experience with pinch-roller type machines in semi-military environments indicated that the device reliability and tape stresses induced would be undesirable in the military application. Vacuum drive systems required an additional vacuum source and a precision rotary joint that would be susceptable to damage in the dust environment. The study and subsequent tests also indicated that a unique pneumatic drive utilizing pressure rather than a combination of vacuum and blow-off pressure would provide the optimum drive.

The operation of the pressure clamp drive is illustrated in Figure 5. A capstan surface was developed that provides a self-generating film of air (hydrodynamic) during periods of non-motion of the tape. The surface is also designed to permit rapid collapse of the air film under external pressure. To drive tape, air pressure is rapidly introduced through a slotted manifold surrounding an arc of the capstan, which forms a blanket over an approximate 80° arc of the capstan, collapsing the capstan-tape air film and forcing the tape into contact with the capstan. The result is a smooth, controlled start with no overshoot or excessive instantaneous accelerations, as shown in the Figure 6 upper start oscilloscope picture. The tape is not sub-

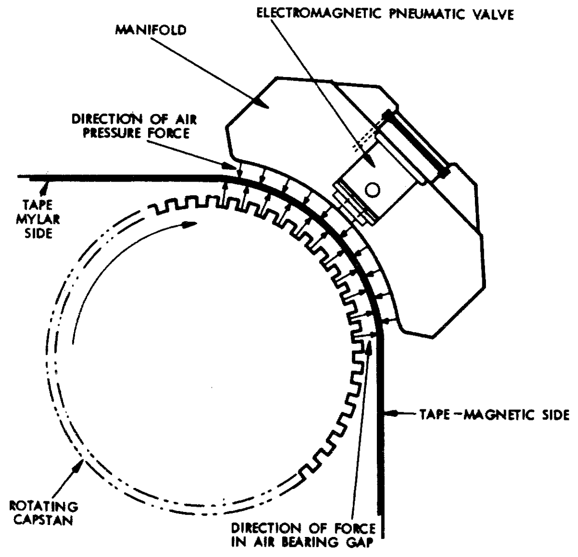

ELECTROMAGNETIC PNEUMATIC VALVE
MANIFOLD
DIRECTION OF AIR PRESSURE FORCE
TAPE MYLAR SIDE
TAPE -MAGNETIC SIDE
ROTATING CAPSTAN
DIRECTION OF FORCE IN AIR BEARING GAP

Figure 5. Sylvania Pneumatic Tape Drive.

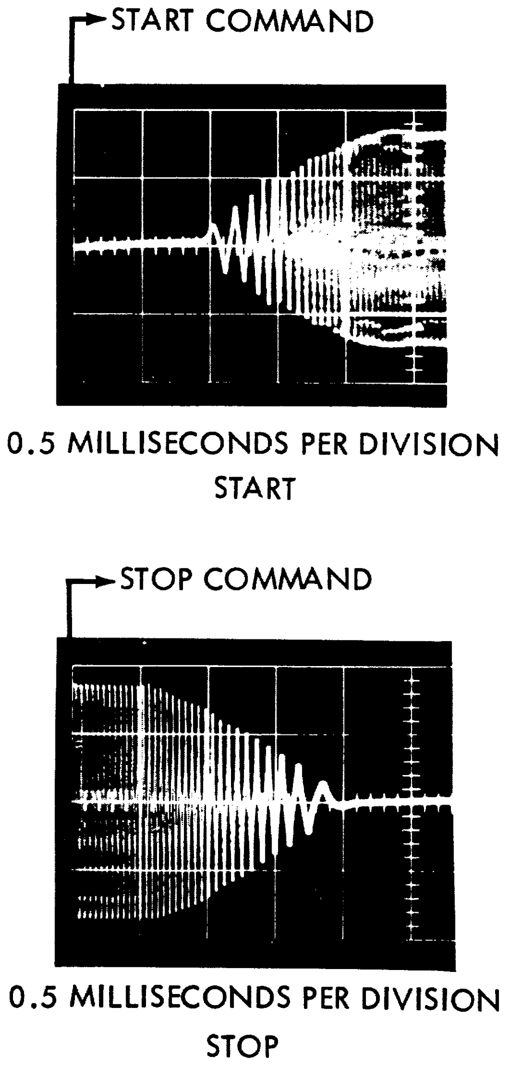


Figure 6. Start-Stop Characteristics.

jected to excessive tensile or compressive forces. Approximately 3/4 millisecond is required for the valve actuation and air propagation and 1 3/8 milliseconds is required for the tape to come from the stopped position to full operating speed. The total start time is, therefore, approximately 2 1/8 milliseconds. The mechanism for accomplishing the tape start is amazingly simple with the only moving part being the high-speed magnetic valve disk. The high-speed pneumatic valve, which was a major development under this program, has been reduced from the original 2 1/2 inch diameter by 5-inches long valve to the "mini-valve" shown in the lower right of Figure 7 which is approximately 3/4-inch diameter by 5/8-inch long, while increasing speed and life. The moving element is a simple magnetic iron disk 5/8-inch by 1/16-inch thick. Driving is accomplished by applying a shaped current pulse which first applies a short duration high current peak for fast action followed by a low current steady holding level to the magnetic coil terminals. The magnetic disk is drawn about 0.014-inch to the coil against the air pressure, opening the output air path and releasing compressed air from an accumulator at approximately 35 psi into the manifold and through its distribution system to the tape surface. There is no adjustment required and valves have been operated more than 100 million cycles without failure. To stop the tape drive, the power is simply removed from the valve. The air pressure forces the disk back onto the seat, cutting off the drive air. The built-in tape path system friction plus a controlled vacuum tape drag, adjacent to the head, brings the tape to a rapid, controlled stop without the need of additional braking. The stop characteristic is shown in the lower view of Figure 6. This also provides a fail-safe arrangement in that unplanned loss of power causes both valves to be forced into the stop condition by the residual air pressure preventing simultaneous drive in two directions.

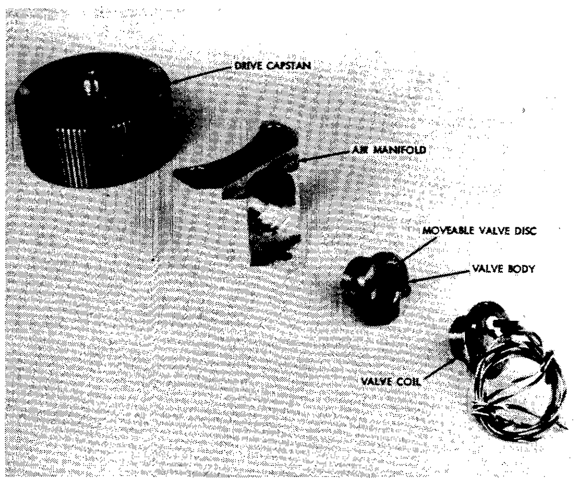


Figure 7. Exploded View of Capstan, Manifold, and Valve.

The complete valve, manifold, and capstan assembly is illustrated in Figure 7. The valve coil is in the lower right with the valve body above showing the air inlet hole. The air control disk is in place in the valve against its O-ring seat. Above the body can be seen the manifold with a radius matching the capstan radius. The air distribution slots are directly below the opening for the valve. A relationship of approximately 0.003 inch is maintained between the manifold and the capstan. A major requirement of the capstan surface is its ability to quickly regenerate an air film between itself and the tape mylar surface when drive pressure is removed and, conversely, allow rapid collapse of the air film when drive pressure is applied. Following a theoretical aerodynamic analysis of the relationships between the valve, manifold, tape and capstans, a variety of surfaces were investigated. The surface shown provided the proper characteristics and offered lower production costs.

Cartridge Tape System with Automatic Loading and Threading

A major problem in the utilization of magnetic tape is its handling during loading, threading, and unloading from the tape handling mechanism. Mishandling of the tape during this operation is one of the major causes of tape errors. The threading requires good operator dexterity. This condition is aggravated when the operation is carried out by military personnel in a less than perfectly clean environment. A second factor involves the loss of machine availability during the load, thread, unload, and rewind cycles.

Utilizing a preloaded tape cartridge with the tape ends semi-permanently attached to the in-

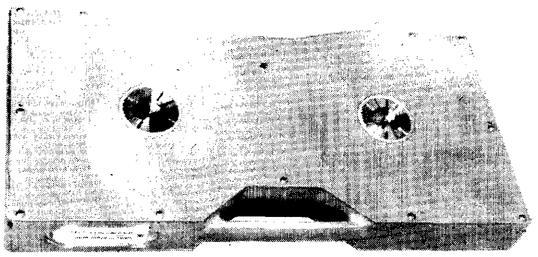


Figure 8. Tape Cartridge, Rear View (Dust Cover Removed).

ternal reels eliminates the handling problems, significantly reduces the load and unload times, excludes the necessity to rewind the tape prior to removal and requires no operator skills. In addition, the tape in the cartridge is protected during external handling and transportation. The preloaded cartridge with 7½ inch diameter reels containing 1200 feet of 1½ mil tape or 1800 feet of 1 mil tape one inch wide is shown in Figure 8. A snap-on dust cover is provided to protect the tape during handling while it is out of the machine. The fully automatic control system for operating the cartridge operates as follows.

The compressed air source for the tape drive is also used to provide the pressures and motions required to automatically load and unload tape. The mechanical functions performed are as follows:

a. Disengage reel from cartridge and force it into driving contact with reel servo drive motor.
b. Shuttle magnetic head and manifolds out of the tape path during load and unload and back into position during normal tape operation.
c. Move head up and down to remove it from contact with the tape during shuttling and high-speed rewind.

A pre-programmed rotary pneumatic control valve was developed to carry out the above controls in minimum space, with maximum reliability and providing positive interlocking to prevent simultaneous or out-of-sequence operations. Linear, pneumatic actuating cylinders are utilized to perform the motions. A schematic diagram of the air control and motion operations is shown in Figure 9. The central control valve less its front cover, rotary stepping solenoid, and wafer controls switch is shown in Figure 10. The grooves on the spindle provide the air switching between the inlet and outlet ports on the periphery. Each operation rewinds the application of pressure to one side and exhaust to the other side of a pneumatic cylinder. Head up-down positions are controlled by the linear solenoid on the left end of the valve shifting the spindle axially. This allows head up during high-speed rewind and return to normal without changing the load-unload sequence of the valve. During operation, the pressure is maintained on the proper cylinders to hold them firmly fixed in position. The cartridge load and control functions are as follows:

a. Inserting the cartridge fully into the front panel recess causes a mechanical latch to lock it in against accidental gravity release and switches sense proper seating.
b. Operator presses front panel automatic load control and machine control takes over and performs the balance of the load functions automatically as follows:
c. Air pressure applied to reel load cylinders separates the reels from the cartridge brake and couples them positively to the servo motor.
d. The reel servo torque motors then slowly feed tape off the reels into the vacuum well until the tape loop is properly sensed by the photocells at the central position in the vacuum well.
e. Air pressure is then applied to move the head and manifold shuttle forward, positioning the head in its precise position above the tape.
f. The rotary valve spool is moved axially to locate the head down to the proper contact angle with the tape.
g. Completion of the head down position signals the machine control system to

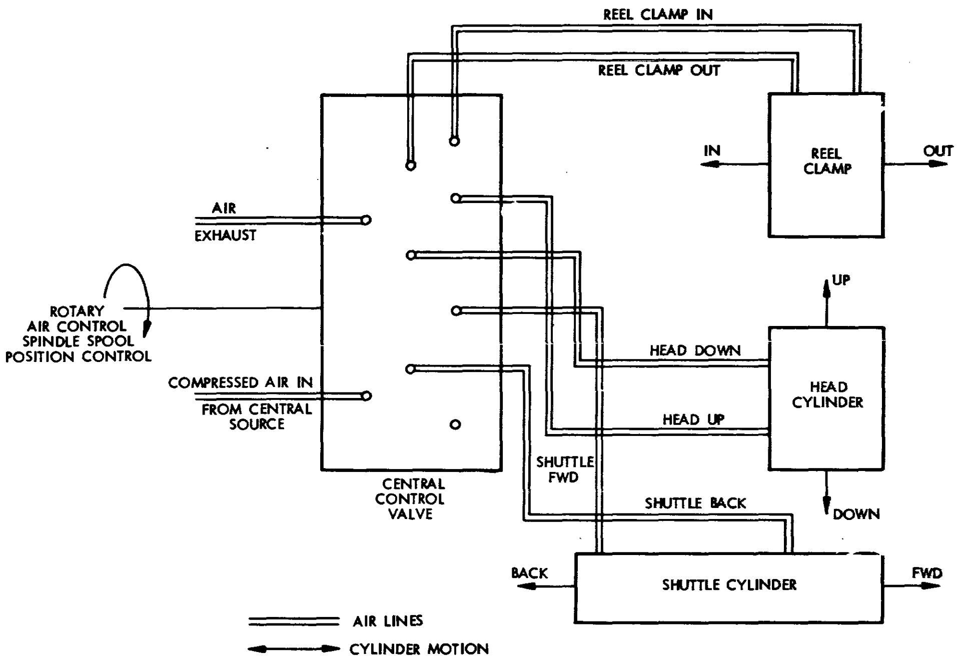


Figure 9. Pneumatic Control System.

energize the "locate Beginning-of-Tape sequence circuit" and closes the reel servo control loop.

The locate Beginning-of-Tape sequence circuit first drives the tape forward briefly to assure that the beginning-of-tape marker is past the position sensor and then in the reverse direction until the precise beginning-of-tape mark is located. At this point, the Tape Ready front panel indicator lights and the transport control is switched to remote control (computer or tape control unit). This total operation takes approximately ten seconds, plus actual rewind time.

The unload cycle operates essentially in reverse of the load cycle and can be initiated at any point in tape use without rewinding tape.

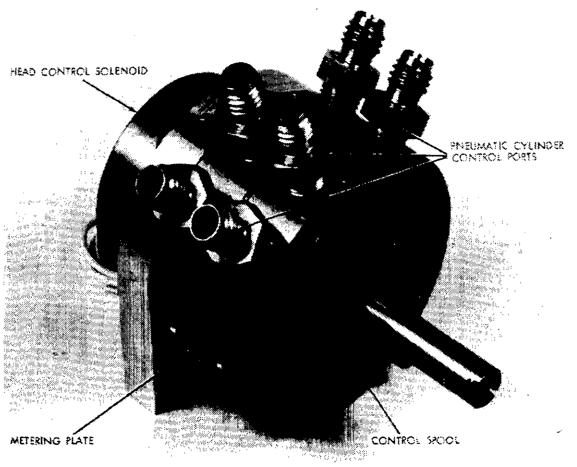


Figure 10. Pneumatic Control Valve.

The Capstan Control System

General Requirements

For normal computer data handling operation (as well as during low-speed data transmission) the synchronous capstan motor is powered by a magnetic frequency standard and a power amplifier. (Low-speed capability is achieved by replacing the capstan motor with a motor-gearhead, a minor field-change.) The capstan must move tape at 100 ips ± 1 percent, irrespective of the computer program, and therefore the capstan motor speed must be constant, well within one percent. Since load disturbances occur each time tape motion is started or stopped, electronic means were developed to anticipate and damp-out the resultant speed variation. For low-speed operation, the capstan must operate at $\frac{2}{3}$ ips with read data synchronized to the data terminal reference signal.

Description of High-Speed Capstan Control

The synchronous capstan motor acts as a torsional spring. That is, a variation in load torque (caused by starting or stopping tape) requires a corresponding variation in torque angle (the angle between rotor magnetic field and the stator rotating magnetic field) before the motor can provide the called-for load torque. This spring action, together with the total inertia of the motor and capstan-drive system comprises a mechanical resonant circuit, essentially undamped. When the load disturbances occur, a train of oscillations results, continuing until the motor settles at the new torque angle. One technique for reducing the effect of transient loads is to use a large flywheel. It was not possible, however, to design the required inertia into a device of reasonable size. Also, a large flywheel makes it difficult to achieve rapid re-wind speeds. In the Sylvania magnetic tape transport, these oscillations are minimized by stepping the stator magnetic field to the new torque angle, and damping (electrically) any residual disturbances.

Stepping the stator magnetic field is accomplished by advancing the phase of stator excitation as load is increased, and retarding the phase when load is removed.

Damping is accomplished by an additional phase-shift, proportional to change in motor speed, sensed by a tachometer.

Phase-shifts are generated in a delay multivibrator, whose timing is controlled by an amplifier. Amplifier inputs are tachometer variations and Forward or Reverse commands.

Description of Low Speed Capstan Control

The specific requirement of the low-speed capstan servo is to match the data-rate read from the tape to the 300 pulse per second data terminal clock to within ±12 characters in a maximum block of 35,810 characters. This tolerance is dictated by the 30-character buffer in the data terminal and provides a ±3 character safety margin. The servo, then, strives to maintain a one-for-one relationship between "clock-in" and characters read by adjusting tape speed. It corrects for both short and long-term effects of tape slippage, tape-dimension changes, recording speed variations and any other errors that result in a variation in the data-storage density on the tape. The servo is designed to compensate for such variations up to a maximum of ±3 percent of nominal data rate.

A block diagram of the speed-control loop is shown in Figure 11. A small synchronous motor is used to drive the capstans through a gearhead. The speed of the motor-gearhead is closely controlled by the frequency of its drive power. The synchronous motor is of the hysteresis type. Although these motors are relatively inefficient, they are particularly useful for this application in that they supply a relatively constant torque all the way up to synchronous speed, and very good damping at synchronous speed. These characteristics make it possible to accelerate the motor by slowly changing the power frequency without loss of synchronism or introduction of phase oscillations. Should the motor lose synchronism, it

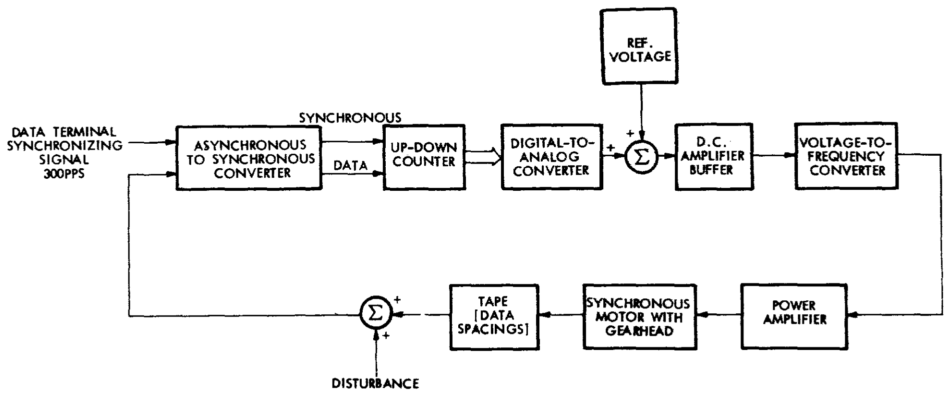

REF. VOLTAGE
DATA TERMINAL SYNCHRONIZING SIGNAL 300PPS
ASYNCHRONOUS TO SYNCHRONOUS CONVERTER
SYNCHRONOUS
DATA
UP-DOWN COUNTER
DIGITAL-TO-ANALOG CONVERTER
+
+
Σ
D.C. AMPLIFIER BUFFER
VOLTAGE-TO-FREQUENCY CONVERTER
Σ
+
+
TAPE [DATA SPACINGS]
SYNCHRONOUS MOTOR WITH GEARHEAD
POWER AMPLIFIER
DISTURBANCE

Figure 11. Capstan Servo Block Diagram.

will simply accelerate until synchronous speed is regained.

The rate of the data pulses from the tape reader is proportional to tape speed. Close control of the data rate could be achieved simply by keeping the frequency of the motor power constant except that data spacing on the tape does vary from normal as outlined above. These variances cause disturbances as indicated in Figure 11. The purpose of the closed-loop control is to compensate for the effect of these disturbances, since a close tolerance is required on the data rate when the recording system is transmitting its data via the data terminal.

Closed-loop control is achieved by letting the input synchronizing pulses serve as a reference for tape speed This is done by comparing the synchronizing pulses and data pulses in an up-down counter as shown in Figure 11. The counter will store a number which increases or decreases depending on an excess of deficit, respectively, in data rate. By means of circuits, the stored number is used to increase or decrease the motor speed to achieve a data rate exactly equal to the input synchronizing rate.

The electronics include a digital-to-analog (D/A) converter which converts the count into a d-c voltage. This voltage is added to a reference voltage in a d-c amplifier which is used as a buffer to provide the necessary impedance level and stability to control the frequency of a voltage-to-frequency (V/F) converter. This converter is an electromagnetic device with an extremely linear V/F transfer characteristic. The output of the V/F converter is raised in a power amplifier to a sufficient level to drive the synchronous motor. Linearity and drift in the electronics is kept within a close tolerance to minimize these additional disturbances. Any small drift in the V/F converter frequency will be quickly compensated-for by the integrating action of the up-down counter. The asynchronous-to-synchronous converter in front of the counter is used to synchronize the asynchronous tape and reference signals. This prevents race conditions in the counter.

Reel Servo System Requirements

a. To follow any sequence of commands supplied to the tape without exceeding the limits of the vacuum well. (Approximately 10 inches working length.)
b. To operate without excessively accelerating or decelerating tape on the reels to the point where tape damage could result. A maximum acceleration or deceleration of 3000 in./sec/sec was established.
c. To operate with motors of minimum size and power requirements.
d. To be capable of very slow open loop operation for load and unload.
e. To be capable of double speed operation for rapid rewind.

General Description

The reel servomechanism is a Type 1 control system. That is, a constant rate-of-change of tape position (at the reel) requires a constant tape-loop error-position in the buffer well. This type of control, utilizing tachometer feedback (characterized according to diameter of tape on the reel so that feedback is tape speed rather than motor speed), was chosen to allow the tape-loop excursion (during a capstan reversal) to encompass the whole buffer well rather than half the buffer well as would occur with a position servo. This choice results in halving the required torque, so that the reel servo motor (torquer) is little more than half the size of that for a comparable position servo. It uses half the power and subjects the tape to smaller acceleration, thereby preventing tape damage.

The choice of a Type 1 servo, in combination with controlling the motor through a bilateral-switching power amplifier, results in the optimum servo; one that performs the control function in a minimum buffer well using minimum power. A block diagram of the reel servo is shown in Figure 12.

Error Sensing

a. Position Error. The amount of tape reserved in the vacuum buffer well is sensed by an array of photoconductors, equally spaced along the length of the buffer well, connected in a bridge circuit (see Figure 13), and illuminated by lamps similarly spaced on the opposite side of the well.

   Since the magnetic tape is opaque to visible light, the number of photoconductors illuminated is determined by the position

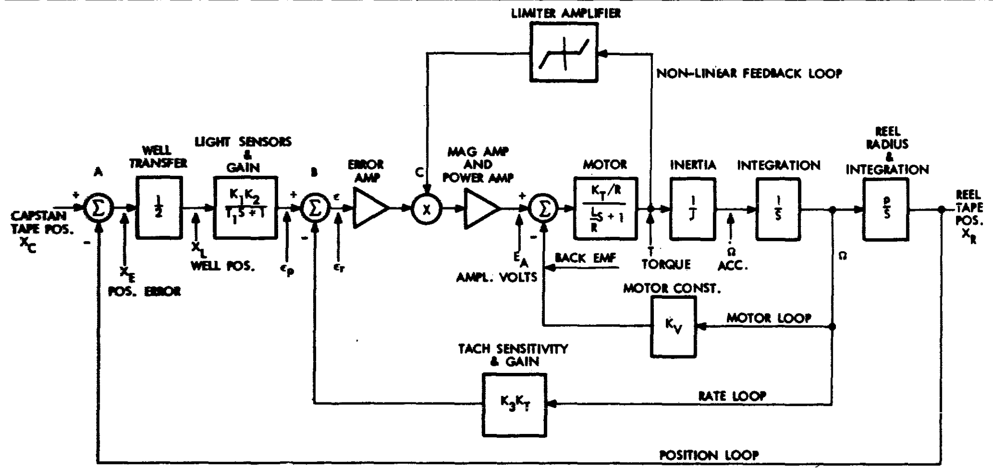


Figure 12. Reel Servo Block Diagram.

of the tape-loop in the buffer well, fixing the amount of current flow into a summing resistor. The voltage drop across the summing resistor, then, is proportional to the tape-loop position, and is called the position-error signal.

b. Rate Error. Tape speed at the reel is not easily sensed, and therefore, is computed from the product of reel shaft speed and the radius of tape on the reel.

Reel shaft speed is sensed by a DC tachometer mounted integrally with the reel torquer, whose output is a DC potential directly proportional to shaft speed.

Radius of the tape on the reel is sensed by an array of photoconductor illuminated via fiber optics (see Figure 14) whose input faces are located between the reel and the buffer well. The fiber optic faces are sequentially masked from their source of illumination as the tape content of the reel increases. Masking of the illumination causes individual photoconductors to act as an open circuit, sequentially disconnecting characterized resistors from the tachometer circuit, so as to multiply the tachometer signal by the tape radius. This product is called the Rate Error Signal.

The error amplifier is a direct-coupled amplifier that uses internal series voltage feedback to stabilize gain. Inputs to its first stage, a differential amplifier, are position-error signal, rate error signal, and zero-level correction.

A magnetic amplifier, controlled by the error amplifier is used as a pulse-width modulator to control the power stage. The latter consists of

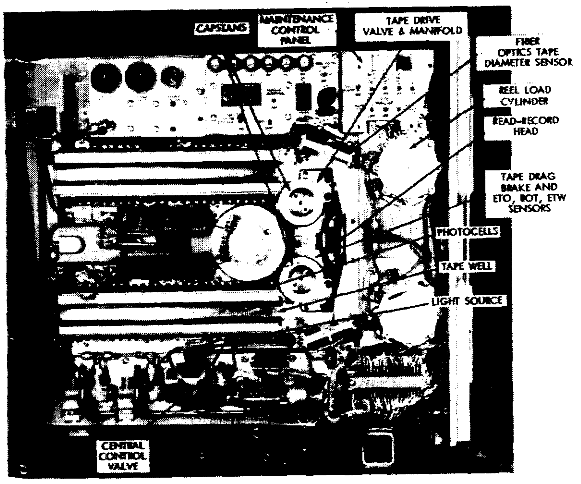

CAPSTANS
MAINTENANCE CONTROL PANEL
TAPE DRIVE VALVE & MANIFOLD
FIBER OPTICS TAPE DIAMETER SENSOR
REEL LOAD CYLINDER
READ-RECORD HEAD
TAPE DRAG BRAKE AND ETO, BOT, ETW SENSORS
PHOTOCELLS
TAPE WELL
LIGHT SOURCE
CENTRAL CONTROL VALVE

Figure 13. Interior of Army Tape Transport Module (Close-Up).

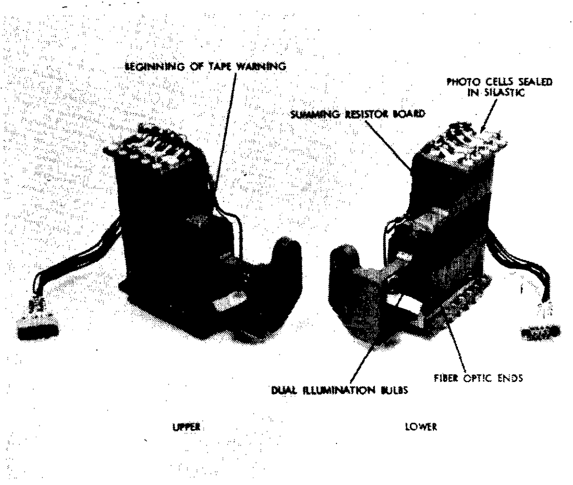


Figure 14. Details of Fiber Optic Tape Sensing System.

pairs of cascaded switching transistors which connect either +30 or —30 volts to a smoothing filter at a switching rate of 4KC. The pulse-width modulation causes the output of the filter to vary in amplitude and phase as required to control the direction of rotation and the torque of the reel motors. The reel motors are DC torquers, with permanent-magnetic excitation rated for peak speeds of 1300 rpm and peak torques of 567 inch-ounces.

A current-limiting loop limits the maximum motor torque so that tape cannot be damaged by excessive accelerations.

Servo Operation

Utilization of the reel servo motors at low speed during the automatic loading, threading, and unloading has been described earlier. The following describes the operation of the reel servo for normal tape running.

When the Start button is pushed, the reel motors operate in open loop in a sequence that threads the tape into the transport. Subsequent operation is under the command of the computer. The reel speed is slaved to the capstan tape feed. The only error sensing in the system is the position loop, which gets its signal from 26 photocells spaced along each buffer well. After the tape is prepositioned in the well according to direction, any change in tape position creates an error signal which appears at summing point A. (Refer to Figure 12.)

Because the tape speed is constant and the reel diameters vary as tape is payed off and taken up, it is necessary to apply correction to the reel tachometers. This is done by the fiber optics above the wells which sense the tape radius. The signal they generate modifies the tachometer output. The resultant is then fed back to summing point B. A computer command to stop or reverse will create a large error signal which in turn would cause excessive acceleration leading to tape damage. A non-linear feedback loop senses excess current and applies a signal at summing point C, opposing those from the position and rate loops to limit acceleration.

The dynamic braking capabilities of d-c motors are utilized to effect quick reversal. At the command for reversal, the motors act as if they are connected in series, dynamically braking each other; the current is limited to prevent excessive deceleration. When the motors come to rest, reverse polarity voltage is applied to accelerate. This method significantly reduces the power consumed during reversal, a reduction from 500 to 100 watts in this case.

CONCLUSION

In this paper, it has only been possible to present a broad picture of the technical developments of the magnetic tape system, and to describe, briefly, four of the significant technical areas. Future papers and articles will describe the details of such functions as read-write, error correction and detection, and machine control electronic and mechanical advances.

All major operating sections of the tape transport were built and tested during the development program. Two complete feasibility models of the electromechanical system were assembled and performance tested. The tests, which demonstrated that the design and reliability objectives were met included the following.

Data Reliability

In order to establish preliminary error rates, a tape transport, representative of the final tape handling and read-record head-tape relationship, was used to read and write complex blocks of data. Test data was provided and monitored by a complete computer simulator and error detection and recording system. Runs of 50 and 100 million error-free characters were achieved without the use of error correcting schemes. These results exceeded the design objectives by a factor of approximately two orders of magnitude.

Tape Wear

A tape life test was also run on the same tape transport. One block of data, containing the maximum number of Fieldata words and six different random characters, was written and then read during half a million passes over that block. At this point, the test was discontinued. The tape showed no signs of failure and the signal was down only about 10 per cent, a figure it reached early in the test run. In addition, the error rate did not increase at any time during the test, nor did the head show any discernable wear.

Low Speed Capstan Servo Speed Correction

Integration tests on the low speed capstan servo were conducted to test its ability to correct for data timing errors. Errors in data positioning on the tape were recorded in varying amounts up to four percent. (The design specification was three percent maximum.) These errors were recorded under open loop with the tape at slow speed. When replayed under servo control, the accumulated error in bit deviation over the maximum block length was below the ±12 characters allowed.

High Speed Capstan Speed Control

To insure that the high speed capstan maintained tape speed of 100 inches per second, ±1 percent, tests were conducted taking advantage of the accurately known read-to-write head spacing. The write repetition rate was adjusted so that its period equalled the time it took for a recorded pulse to travel between the write and read head. Tape speed was then directly calculated from the spacing and the time, and under extensive testing, found to meet the specification.

Reel Servomechanism and Tape Position Sensing System

Utilizing the computer simulator and covering a range of worst case programming conditions, the reel servo system controlled the tape loop, keeping it within the confines of the vacuum well.

Read-Write and Control Electronics

All electronics were designed for worst case conditions, and operationally tested in preliminary form. These tests included high and low temperature operating extremes and high and low power supply voltage extremes.

Equipment Status

As of July 1963, fourteen tape transport modules, representing a variety of Army and Navy types had been fabricated, assembled, and were in the final test cycle. Twelve of these will be delivered to the services for operational tests and two will be retained by Sylvania for extensive reliability and performance tests.

ACKNOWLEDGEMENTS

The authors wish to thank the Sylvania Engineering Staff for its technical contributions on the project and assistance in preparation of this paper. In particular, the following personnel deserve special mention:

L. CHERNIKOFF and N. VANBUREN—Reel and Capstan Servo

J. SIMON and P. DELUCA—Mechanical and Pneumatic Design

G. BIRD—Machine Control Systems

A. ASHLEY and R. MEDNICK—Data Electronics and System Logic

S. WOODSUM—Magnetics and Magnetic Head

IBM 7340 HYPERTAPE DRIVE

R. A. Barbeau and J. I. Aweida
Development Laboratory, Data Systems Division
International Business Machines Corporation, Poughkeepsie, New York

INTRODUCTION

The trend toward increased speed and greater capabilities in computers has created the need for tape drives with increased data rate and decreased access time. The IBM Hypertape Drive (Fig. 1) was developed to meet these demands with the added objectives of increased reliability and automatic tape handling.

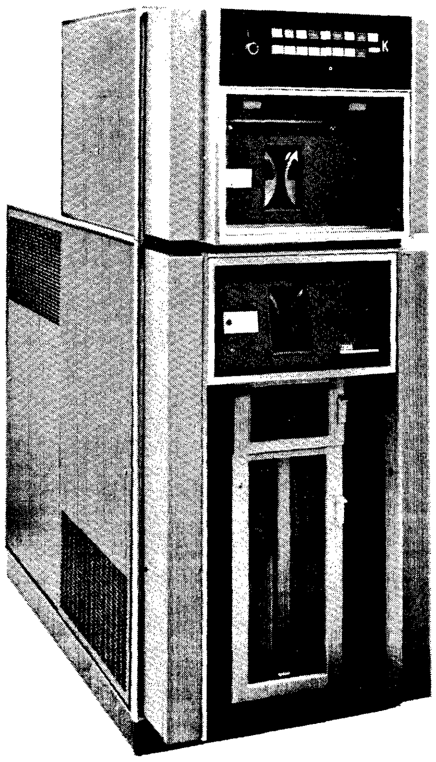
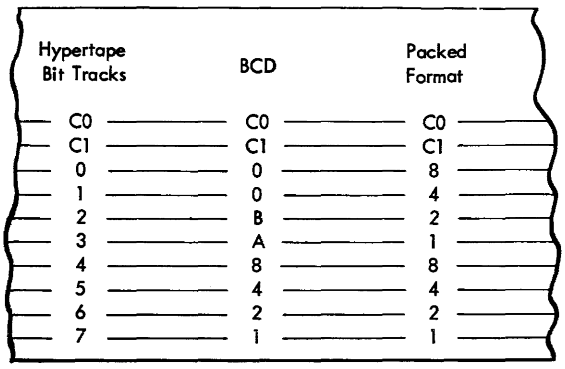


Figure 1. IBM 7340 Hypertape Drive with Automatic Cartridge Loader.

The Hypertape Drives, controlled by the IBM 7640 Hypertape Control Unit, can presently be used with the IBM 7074, 7080, 7090, and 7094 systems. The tape used is 0.1 mil oxide on a 1 mil Mylar* base; it is one-inch wide with 1800 feet on a reel. Ten tracks are written across the tape—eight for information purposes and two for error detection and correction. In alphanumeric mode, six of the information tracks are utilized per character. In packed numeric code, two 4-bit characters are written side by side across the tape (Fig. 2). The character density is 1511 per inch; tape speed is

* Trademark of E. I. duPont deNemours Co., Inc.



Hypertape Bit Tracks	BCD	Packed Format
C0	C0	C0
C1	C1	C1
0	0	8
1	0	4
2	B	2
3	A	1
4	8	8
5	4	4
6	2	2
7	1	1

Figure 2. Hypertape Character Formats.

112.5 inches/sec. This results in an alphanumeric data rate of 170,000 characters/sec, or a packed numeric data rate of 340,000 characters per second with an average access time of 4.2 milliseconds and an average inter-record gap of 0.45 inch.

The drive utilizes a single fluid-coupled capstan to control the tape motion. The use of such a capstan greatly simplifies the tape path and makes possible the use of simple channel guides to give proper tape tracking and to minimize tape skew.

The tape tension supplied by the vacuum in the columns holds the tape tight around the capstan. Moving (or stopping) the tape is accomplished by moving both the tape and the capstan. The read-write head is the only component in the tape path that is mounted on the recording side. This head is hydrodynamically air lubricated; thus, the moving tape is separated from the head by a thin layer of air.

Transport Design and Cartridge

To achieve high reliability and automatic handling of tape, the tape is packaged in a cartridge; the height of the drive is kept low to facilitate the manual loading and unloading of the cartridge in the drive. Limiting the height of the drive posed restrictions on the length of the tape buffers and the tape reel control system.

The use of a cartridge satisfied the following objectives:

1. Minimization of tape load and unload time.
2. Unload without rewind.
3. Protection of the tape from damage caused by operating personnel or contamination by foreign particles.

The cartridge as shown in Fig. 3A houses the "supply reel", the magnetic tape, and a "take-up" reel in a dust-resistant environment. It is constructed such that the sides, back, and top (including handle) are assembled into one unit. The front cover of the cartridge is removable to permit reel replacement. This cover (Fig. 3B) pivots at the front top and is secured by two latches. The latches also furnish the cover closing pressure. A flat tension spring holds each reel against a rubber ring that serves as a dust seal and brake. The back of the reel hub contains teeth; the tension spring serves to keep them engaged to the machine teeth when the cartridge is transport oriented.

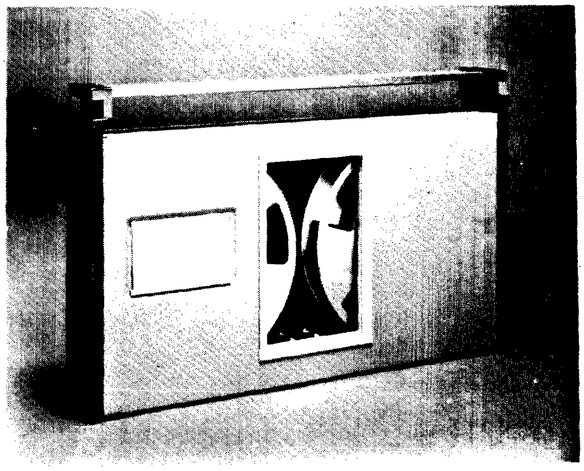


Figure 3a. Hypertape Cartridge—Front View.

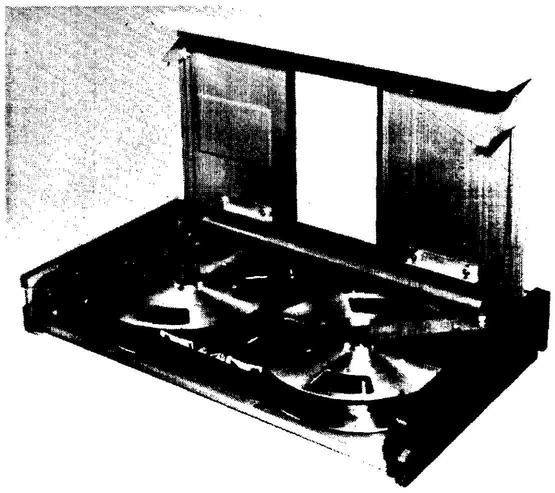


Figure 3b. Hypertape Cartridge—Front Cover Open.

The tape transport accepts this cartridge (Fig. 4A), automatically opens the cartridge cover, engages the tape reels (Fig. 4B), and causes the tape to thread into operating position. Packaging both supply and take-up reels in the cartridge increases the capabilities of the drive by eliminating the need for rewind before unload. The tape can be unloaded at any position in the reel; then loaded at a later time to approximately the same location to continue processing or, if desired, it could be rewound on an off-line drive.

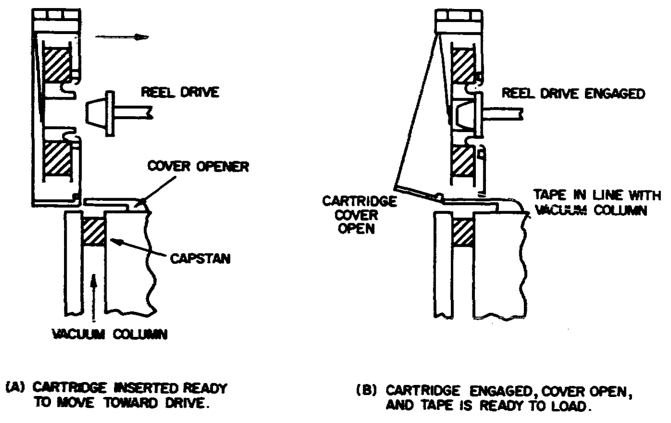

REEL DRIVE
COVER OPENER
CAPSTAN
VACUUM COLUMN
(A) CARTRIDGE INSERTED READY TO MOVE TOWARD DRIVE.
REEL DRIVE ENGAGED
CARTRIDGE COVER OPEN
TAPE IN LINE WITH VACUUM COLUMN
(B) CARTRIDGE ENGAGED, COVER OPEN, AND TAPE IS READY TO LOAD.

Figure 4. Cross-Section View of Cartridge at the Reel.

The basic dimensions of the cartridge, including the carrying handle, are:

17″ long
10.2″ high
2.2″ wide

The cartridge is equipped with a file protect device (Fig. 5) that, by command, either manual or computer programmed, prevents writing on the tape. The setting of file protect by the computer causes a solenoid to depress a plunger, and the indicator will reveal the letters FP. The status of the device is readily apparent to the operator when looking at the rear of the cartridge. Once a cartridge has been file protected, it can only be removed from the file protect status manually under operator control with the cartridge removed from the transport.

TAPE MARKS

Processing of tape in a cartridge necessitates the presence of safeguards against completely unreeling tape off the reels. Therefore, tape prepared for usage on Hypertape drives has three tape marks which, besides defining the usable limits of tape, guard against tape reel run-off.

Two of the marks sense the (BOT) beginning of tape and (EOT) end of tape. They are located 15 feet from the physical ends with 1800 feet of tape between them. The third mark, which is referred to as (EWM) Early Warning Mark, is located 40 feet away from the EOT in the BOT direction.

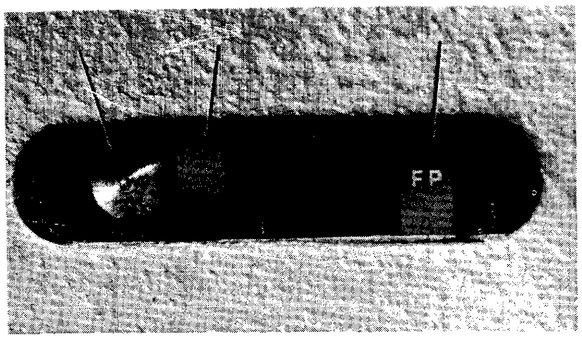


Figure 5. File-Protect Device Showing (Left to Right) the Plunger, Slide, and Indicator.

When a reel of tape is installed in a cartridge, a few layers, up to or including BOT, are wrapped around the cartridge reel. No adhesive in any form is used to anchor the tape to the hub, since the increased thickness due to the adhesive results in embossing the layers of tape above it. Embossed tape has a detrimental effect on the read-write reliability.

The EWM marker is used to signal the computer that the end of tape is approaching. By locating it 40 feet from the EOT, almost ¾ of a million characters could be recorded before the EOT is reached.

Each of the three tape marks consist of a row of holes punched in the tape as shown in Figure 6. BOT and EOT marks are made of 12 holes, 0.093 inches in diameter; no usable information can be stored on that section of tape. The EWM on the other hand, is made of 23 holes, 0.040 inches in diameter, and is located between track 1 and the edge of tape. Thus it does not interfere with the recorded data.

Sensing of tape marks is done optically by the use of a lamp and photocells in each of three positions. The sensing station is located just below the data head. The lamps are fixed in the tape path, while the photocells assembly is moved to position with the head. The drive

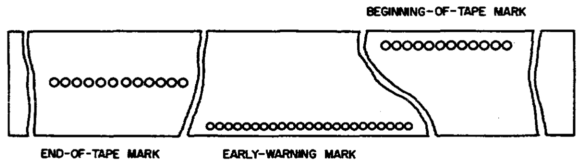


Figure 6. Tape Marks.

interlocks, in case of sense-unit malfunction, to guard against tape reel run-off.

AUTOMATIC TAPE LOADING AND UNLOADING

Factors that affect the ease of loading and unloading tape were carefully studied and were reflected in the design of the tape path shown in Fig. 7. Note the absence of rollers or bearings on the recording surface of the tape, and how the data head is the only surface touching the recording surface. During loading and unloading this head is retracted into the drive, away from the tape.

Automatic tape loading and unloading have simplified the operation of the Hypertape drive and drastically reduced the time needed to complete a load or unload cycle. The computer can start processing tape ten seconds after the operator inserts a cartridge in the drive.

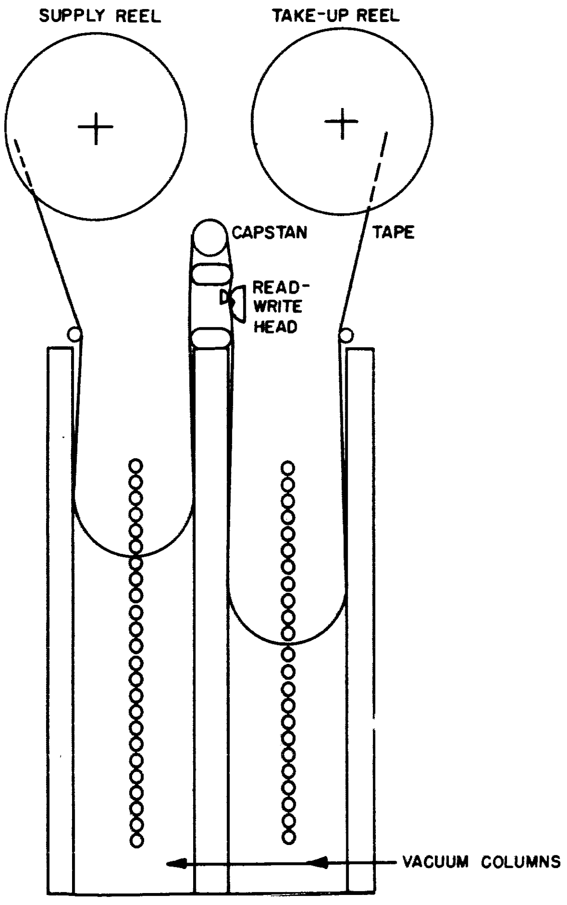


Figure 7. Tape Path of Hypertape Drive.

The Automatic Cartridge Loader (ACL) feature further increased the capabilities of the drive and made it possible to load and unload cartridges into the drive automatically under the control of the computer.

A brief description of the load-unload cycles and the automatic cartridge loader follows:

LOAD CYCLE

To process a reel of tape, the operator places a cartridge in the drive and closes the pressurized housing door. This starts the automatic cycle by retracting the cartridge, meshing the reel hubs to the reel shafts, and opening the cartridge cover (Figure 4). The steps followed during the rest of the loading cycle are summarized with reference to Figure 8.

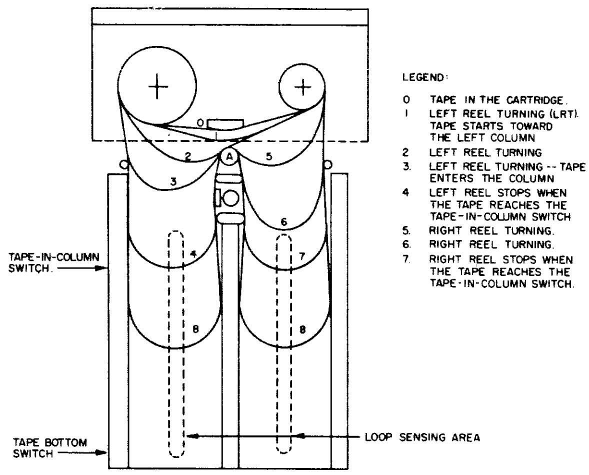

LEGEND:
0 TAPE IN THE CARTRIDGE.
1 LEFT REEL TURNING (LRT). TAPE STARTS TOWARD THE LEFT COLUMN
2 LEFT REEL TURNING
3. LEFT REEL TURNING -- TAPE ENTERS THE COLUMN
4 LEFT REEL STOPS WHEN THE TAPE REACHES THE TAPE-IN-COLUMN SWITCH
5. RIGHT REEL TURNING.
6. RIGHT REEL TURNING.
7. RIGHT REEL STOPS WHEN THE TAPE REACHES THE TAPE-IN-COLUMN SWITCH.
TAPE-IN-COLUMN SWITCH.
TAPE BOTTOM SWITCH
LOOP SENSING AREA
A

Figure 8. Method of Tape Loading.

1. The right reel is braked while left reel lowers the tape into the left column (steps 0-3).
2. When tape reaches the tape-in-column switch, the left reel is braked and the right reel lowers tape into the right column (steps 4-6).
3. When tape in right column reaches the tape-in-column switch (step 7) two functions take place simultaneously:
   a) The auxiliary vacuum applied to the tape in the left column at point (A) is turned off. (This vacuum was applied at the beginning of the load cycle to guard against tape bottoming).

b) Reel motion is under control of the reel servo system and tape in both columns is lowered to the center (step 8).

4. The data head is brought into the read-write position with the tape.
5. Tape is moved toward BOT for approximately one second, or until BOT is sensed, to guarantee having the EOT mark to the left of the mark sensor. After the tape load cycle is complete, the drive is ready to process tape under computer control.

UNLOAD CYCLE

The unload cycle is started by a computer command or by manual control and consists of a) retracting the data head, and b) removing the tape from the columns by winding it on the right reel (the left reel is braked during this operation). When the end of the tape loop approaches the cartridge, the motor torque is limited to insure a safe tape tension between the two reels in the cartridge. The cartridge is then moved to the discharge position during which the reels are unmeshed and the cover is closed.

AUTOMATIC CARTRIDGE LOADER

In the standard Hypertape drive, the cartridge is loaded and unloaded manually by the operator. The Automatic Cartridge Loader feature (ACL) makes it possible to automatically change cartridges under the control of the computer within seconds, thus eliminating computer idle time caused by unavailability of operator.

The ACL consists of a unit which mounts on top of the drive and allows the sequencing of two cartridges automatically. The basic components of this unit are: three cartridge positions on one level, two moving arms that have the ability of advancing the cartridges from one position to the next, and an elevator that loads and unloads cartridges into the Hypertape drive.

The method of operation is seen in the sequence of sketches in Fig. 9. With cartridge A in use in the drive, cartridge B is in the ACL. When a "Change Cartridge" command is given, cartridge A is unloaded to the cartridge holder in the drive and the elevator of the ACL brings it up to its upper position. Next, both cartridges are moved forward a step. This leaves cartridge A in the discharge pocket and places cartridge B on the elevator's platform which lowers it into the cartridge holder of the drive.

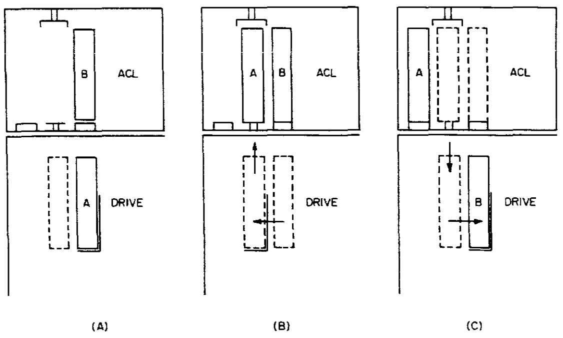


Figure 9. ACL Sequence of Operation—Side View.

Following this cartridge loading sequence, the tape drive takes over. During the processing of cartridge B, cartridge A can be removed, and another cartridge loaded in the ACL.

CAPSTAN DRIVE MECHANISM

A single clutched capstan is used to control the motion of the tape as shown in Fig. 7. Acceleration (or deceleration) of the tape is accomplished by accelerating both the tape and the capstan assembly. The tape is kept from slipping over the capstan surface by wrapping the tape 180° around the capstan, using one pound tape tension, and by covering the capstan surface with a thin layer of grooved rubber.

The mechanism that controls the motion of the tape and capstan was designed in a single subassembly, as shown in Figs. 10 and 11. The

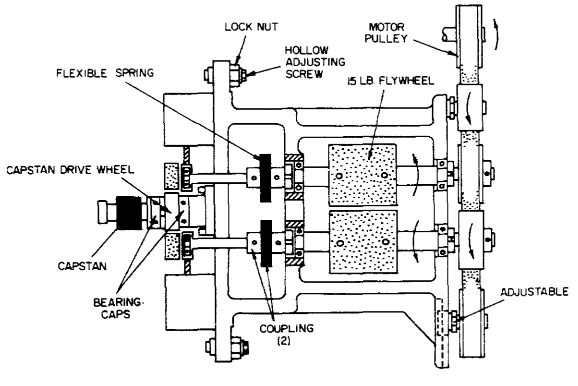


Figure 10. Capstan Drive Assembly—Top View.

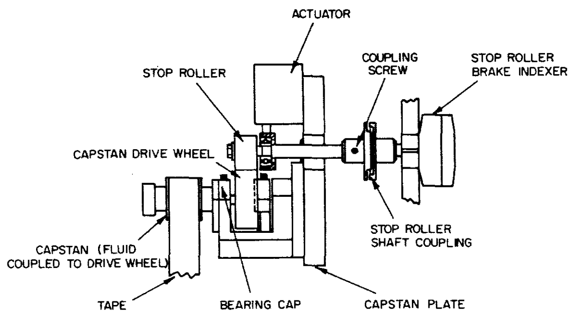

ACTUATOR
STOP ROLLER
COUPLING SCREW
STOP ROLLER BRAKE INDEXER
CAPSTAN DRIVE WHEEL
STOP ROLLER SHAFT COUPLING
CAPSTAN (FLUID COUPLED TO DRIVE WHEEL)
TAPE
BEARING CAP
CAPSTAN PLATE

Figure 11. Capstan and Stop-Roller Assembly.

sealing of this area prevents loose wear particles from entering the tape compartment. To eliminate the problem of flexing a rigid shaft during this operation, flexible couplings are mounted on the rubber roller shafts. Flywheels are mounted on the drive roller shafts to minimize speed variation due to high start-stop ratio. A two-speed synchronous motor belt drives both the forward and reverse drive roller shafts and produces tape speed of 112.5 in/sec. for normal operation, and 225 in/sec. for high-speed rewind.

Fast starts and stops were produced by the mechanism described; however, it was necessary to minimize tape vibration and overshoot. This was accomplished by unique design of the capstan which has a layer of fluid between its tape driving surface and drive wheel shaft. The reasons for and contributions of this unique design follow.

FLUID-COUPLED CAPSTAN

In the early stages of development, when a solid (non-viscous) capstan was used, the start back end of the capstan (drive wheel) is surrounded by three rubber-covered rollers, two of which are continuously running and give forward and reverse motion; the third is stationary and is used to stop the capstan. The rollers are brought in contact with the capstan's drive wheel by the action of moving coil actuators.

To execute a start operation, the forward or reverse roller is brought in contact with the drive wheel while simultaneously retracting the stop roller. All three rubber rollers act on one common surface away from the tape; proper time to 112.5 in/sec. was less than the specified 3.0 milliseconds. However, extreme tape velocity variation occurred at the data head after the above velocity was initially reached. The damped periodic variation was as much as 30% above and below the nominal velocity. Stop times were also within specification, but periodic oscillations around zero velocity were present.

Experimental analysis of the system showed that the velocity variation was mainly due to:

a) Excitation of the natural frequency of longitudinal vibration of the tape.
b) Excitation of the natural frequency of torsional vibration of the capstan assembly acting as a free shaft with an inertia on each end.
c) Torsional vibration (overshoot) of the capstan assembly due to elastic torsional deformation of the drive-roller rubber and drive shaft during capstan acceleration.
d) Undamped vibration of the drive roller actuator arm which caused the roller to repeatedly lose contact with the drive wheel.

Due to the complexity of the vibrational system (several mass, spring and damping elements) and the discontinuous forcing functions, a largely experimental approach was taken to the analysis and reduction of the velocity variation.

The problem was divided into two parts; the first consisted of controlling the actuator arm and roller bounce away from the capstan's drive wheel without affecting the short transfer time. This was successfully accomplished by using rollers covered with $\frac{3}{32}''$ nitrile-base rubber of 85-90 Durometer hardness, and by controlling the actuator current pulse as described in the section on actuators.

The second part, which consisted basically of excitation of the natural frequency of the tape in the column, proved to be one of the major technical problems in the Hypertape drive.

Analysis showed that the torque pulse as applied by the drive roller to the capstan dropped sharply when the roller surface and capstan speed became equal. The sharpness of this drop is above the natural frequency of the tape in the column and excited it into vibration. Theoretically, the response of a spring-mass-damper system acted upon by a pulse-type forcing function is largely determined by the pulse's shape, amplitude and duration.[1] Generally, the smoother and more symmetrical the pulse, the lower the amplitude of the residual vibration. Thus, modification of the torque pulse was necessary to control the vibration.

Based on the above, a successful solution was obtained by decoupling of the tape and its driving surface from the rest of the capstan by a fluid coupling. The use of such a coupling appropriately changed the shape and amplitude of the input torque pulse applied to the drive surface, and thus eliminated the velocity variation due to the longitudinal vibration of the tape and reduced the variation caused by other sources. Typical tape velocity vs. time curves before and after using the fluid coupled capstan are shown in Fig. 12.

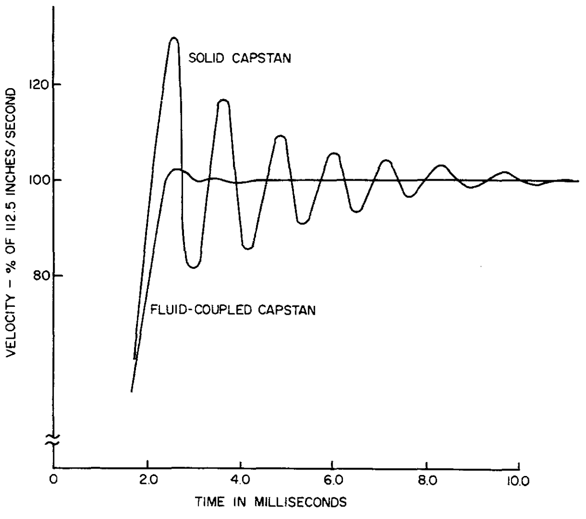


Figure 12. Fluid-Coupled Capstan Versus Solid Capstan.

The fluid coupled capstan as shown in Fig. 13 has a thin layer of silicon fluid separating the tape driving portion of the capstan (capstan's front end) from the capstan shaft and drive wheel; thus, the capstan's front end can rotate with respect to the drive-wheel shaft. This motion is restrained only by the viscosity of the silicon oil layer and the friction in the bearings and oil seal separating the two subassemblies of the capstan.

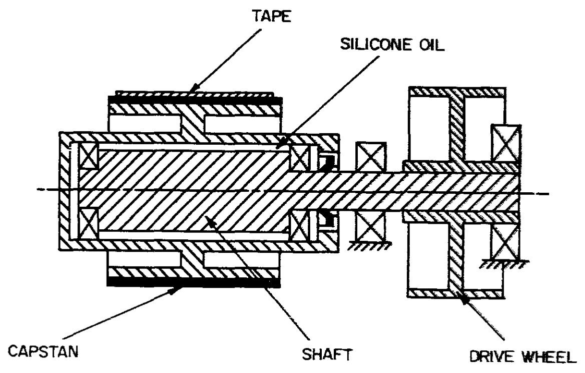


Figure 13. Schematic of the Viscous Coupled Capstan Assembly.

The torque transmitted through the fluid coupling is a function of the differential velocity between the capstan's front end and the drive-wheel subassembly; therefore, an appreciable amount of slippage occurs while accelerating and decelerating the capstan. When the capstan is moving tape at constant velocity, on the other hand, a slippage of 0 to 0.2% of the nominal velocity occurs between the two surfaces, as determined by the frictional characteristics of the bearings, oil seal, and tape path.

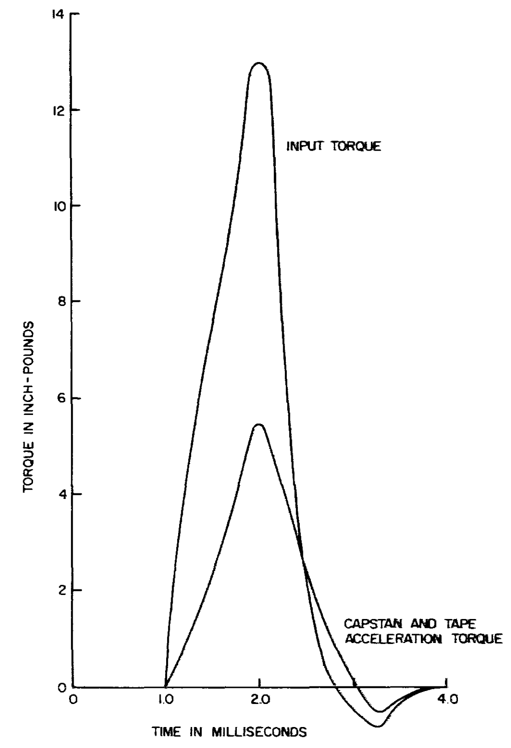
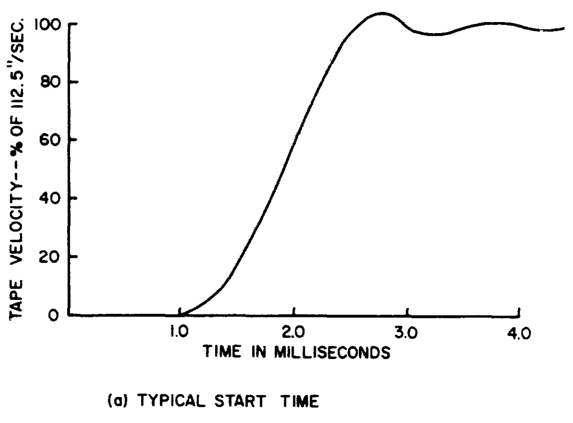


Figure 14. Torque versus Time Curves.

In summary, the fluid coupled capstan has smoothed the input torque pulse as seen by the tape in comparison to the input torque pulse as supplied by the drive rollers as shown in Fig. 14. The use of this capstan made it possible to accelerate the tape to 112.5 in/sec. in an average of 2.5 milliseconds with a velocity variation less than 5%. Typical velocity vs. time curves during acceleration and deceleration are shown in Fig. 15.

ACTUATORS

The three rubber rollers that drive or stop the capstan are brought in contact with the capstan's drive wheel by the action of three actuators. These actuators utilize the moving-coil principle, and were chosen because of their fast response and relatively constant force with stroke. It takes less than one millisecond to transfer the roller from its retracted position to the capstan drive wheel. The constant force, on the other hand, reduces sensitivity to the roller gap.



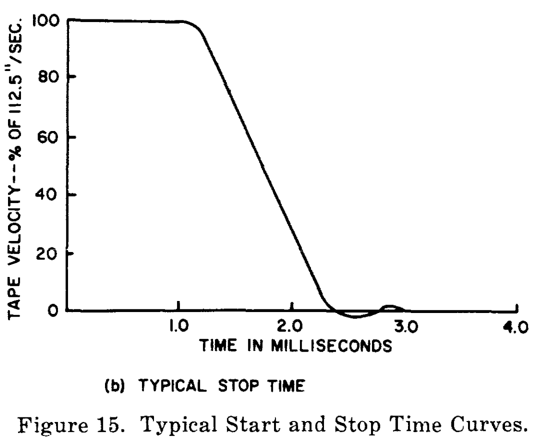


Figure 15. Typical Start and Stop Time Curves.

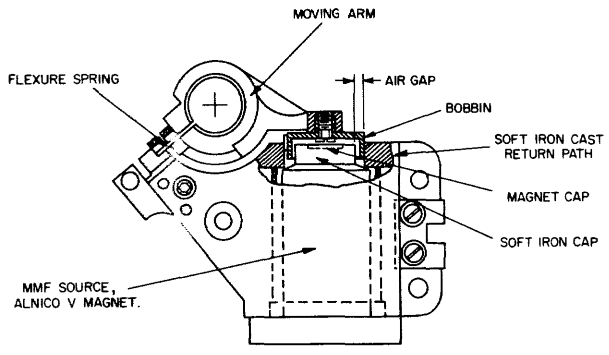


Figure 16. Actuator Arm Assembly.

The actuator, as shown in Fig. 16, consists mainly of a permanent magnet, return path, bobbin and moving arm. The moving arm was designed to give a high output force at the rubber roller. Pivoting of the arm to the assembly is accomplished by the use of a flexure plate. The use of such a plate eliminated the problems associated with pivot points such as *fretting corrosion* which would cause jamming. It also made possible the preloading of the actuator arms. Forward and reverse rollers are preloaded away from the capstan drive wheel, while the stop roller is preloaded towards it. Thus, in case of power failure, the capstan is stopped and no tape damage occurs.

For the actuator to supply the force and transfer time required, the arm design and driving current were optimized as described below:

1. *Arm Design:* The actuator arm was designed to give the minimum transfer time by:
   a) Keeping the mass of the bobbin, arm and rubber roller as small as possible, and by designing a short moment arm; i.e., the roller was located as close to the pivot as possible.
   b) The transfer distance of the roller before contacting the capstan's drive wheel was made only 0.003 inches.
2. *Driving Currents:* The actuator's driving current is controlled to give the desired acceleration and damping. Two high-current pulses are used to properly control actuator motion when the roller is driven into contact. The first high-current pulse gives quick transfer time; the second pre-

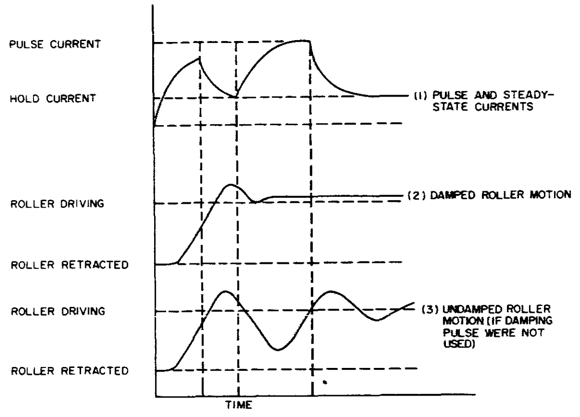


Figure 17. Pulse Current and Roller Action versus Times.

vents the rubber roller from rebounding. A low holding current is applied after the transfer damping period; it provides the roller contact force necessary to maintain constant velocity and minimized coil heating. The shape and effect of the current pulses on the rubber roller motion is shown on Fig. 17.

Life testing of these actuators has shown exceptionally good results. So far, over three billion cycles have been successfully accrued on several actuators.

BRAKE-INDEXER

To stop the tape and capstan, a stationary rubber roller is brought in contact with the capstan's drive wheel. The rubber-roller shaft is connected through a flexible coupling to a hydraulic brake-indexer, as shown in Fig. 11. The purpose of this device is to insure stopping of the capstan in approximately 2.5 milliseconds while allowing slight rotation of the rubber roller during every stop operation.

The brake-indexer was designed so that its mass moment of inertia is much greater than that of the capstan. When the stationary stop roller is brought in contact with the rotating capstan drive wheel, part of the kinetic energy of the capstan is converted into heat due to slippage. At the same time, the roller is accelerated so that the remainder of the energy is dissipated as heat in the indexer itself.

The internal construction of the brake-indexer consists of a finned rotary hub which works in shear against a stationary finned housing. The viscous media in this unit is a high-viscosity silicon fluid. To obtain indexing of approximately 0.010 inches, oil viscosity of 40,000 centistrokes was used.

Indexing the stop roller has given especially good results by distributing the wear and heating caused by successive stops over the circumference of the roller. The use of a hydraulic device with no mechanical contacts, on the other hand, has given consistent stop times and accurate indexing. It also eliminated the problems associated with dimensional changes and frictional wear associated with mechanical indexers.

REEL-DRIVE SYSTEM

The establishment of tape cartridge loading height in the Hypertape drive posed restrictions on the vacuum columns and type of reel-drive system that could be considered. This led to the use of continuous control and a non-center seeking system. The non-center seeking system would make available maximum tape buffer length for tape reversal; with tape stopped, the system would be center seeking to allow for tape movement in either direction from static. With tape moving in the forward direction, the loop position is high in the left column and low in the right so that full advantage of the tape buffer can be realized during a dynamic reversal; the opposite applies when the tape is moving in the reverse direction.

The continuous control, on the other hand, with its ability to apply the correct amount of power when needed and for as long as needed, made it possible to operate within a 10-inch effective vacuum column length. The reel motor is controlled by the motion of the loop in the column. The design of a capacitive sensing device (described later) as a tape--loop sensor gave the desired result. Through this unit both the loop position in the column (X), and the rate of change or velocity of the tape in the column ($\dot{X}$) were obtained.

The magnetic tape reels are controlled by two identical isolated reel control systems. The right reel control system, as shown in Fig. 18, consists of a standard d-c motor which is connected to the reel hub through a 4:1 pulley

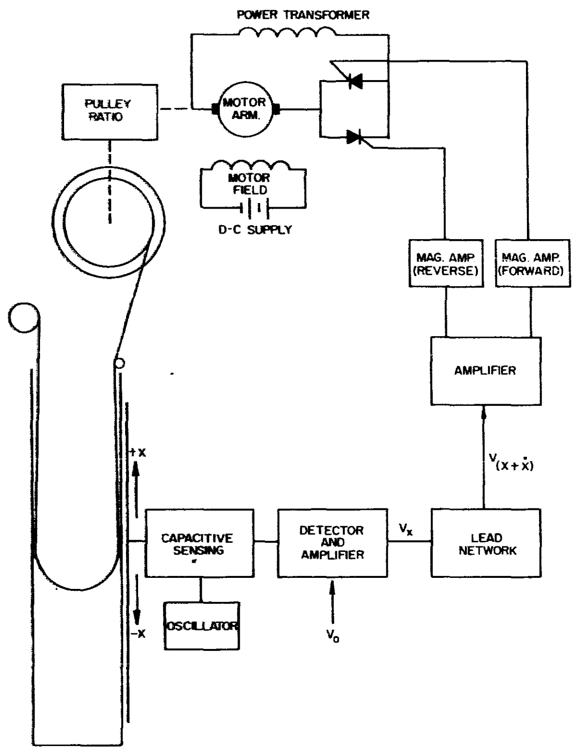


Figure 18. Reel Drive System.

ratio, thus making it possible to use a relatively small motor with only 1/4 the torque required to control the tape. To control the power to the motor, the output of the loop position sensor is detected, amplified and added to a reference signal (V_0), then differentiated by the use of a lead network. The output of the lead network is a function of the position (X) and velocity ($\dot{X}$) of the loop in the column. The net signal is then amplified and used to control the current to the motor's armature by the use of magnetic amplifiers and silicon controlled rectifiers.

The loop position signal (X) controls the power to the motor when the loop is stationary; i.e., when the velocity of the tape leaving or entering the column equals that of the tape over the capstan. When a dynamic reversal takes place, an appreciable signal ($\dot{X}$) develops due to the loop's velocity. This signal results in an increase of the power to the motor.

The motion of the motors is also controlled during the tape load and unload cycle. During these operations the torque applied is controlled by the various column and catenary sensing stations.

CAPACITIVE LOOP-POSITION SENSOR

Sensing the location of the loop in the column is accomplished by the use of the following capacitive method. This method was chosen because of its high reliability coupled with simplicity of construction and linearity of output.

A capacitance value that is a function of the loop's position in the column is obtained by utilizing the pressure differential in the vacuum columns to activate a moving diaphragm between two metal plates, as shown in Fig. 19. The diaphragm consists of a metal layer with Mylar on both sides.

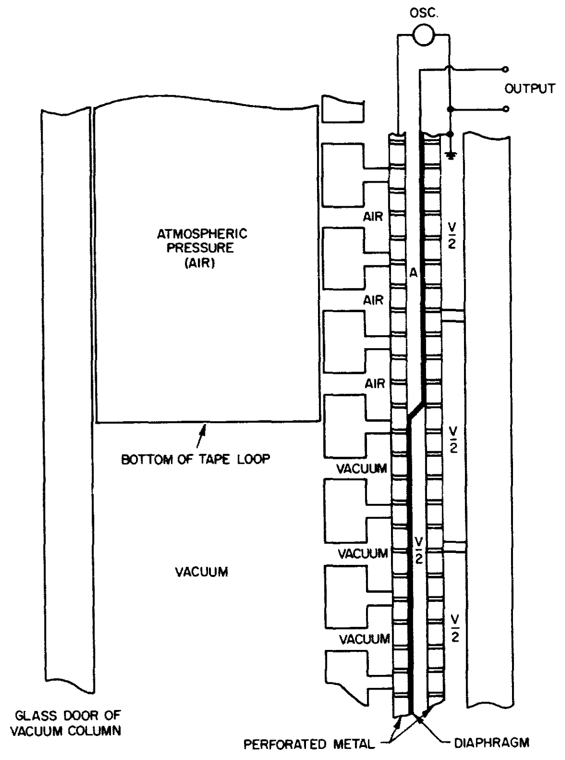


Figure 19. Capacitative Loop Position Sensor—Exploded View.

An insulating plate connects the diaphragm area to the back surface of the vacuum column through 1/16-inch holes spaced 1/4 inch apart. The perforated metal plate on the back side of the unit is connected to a common chamber which has a vacuum level equal to half that of the column vacuum. The half vacuum is obtained from the main column vacuum source, and therefore any change in the column vacuum is reflected in the half vacuum, thereby maintaining a constant ratio.

To obtain a large capacity ratio, the entire rear of each column was utilized for this unit, giving an approximate capacitor range 0.002 to 0.020 μf.

The method of operation of this sensor is seen by referring to Fig. 19 which shows a loop of tape in the column. The section of the diaphragm above the loop is pulled against the back perforated plate due to the differential of pressure which is atmospheric in front and half vacuum in the back. The section of the diaphragm below the loop, on the other hand, is pulled against the front perforated plate, since the front is at full vacuum versus half vacuum for the back.

READ-WRITE SYSTEM

The Hypertape system uses phase encoding type of recording. There is a signal for both ones and zeros, as shown in Fig. 20. Hence, when the tape is read, the sensing and decoding of a bit depends on the phase of the recorded signal rather than its magnetic strength.[2]

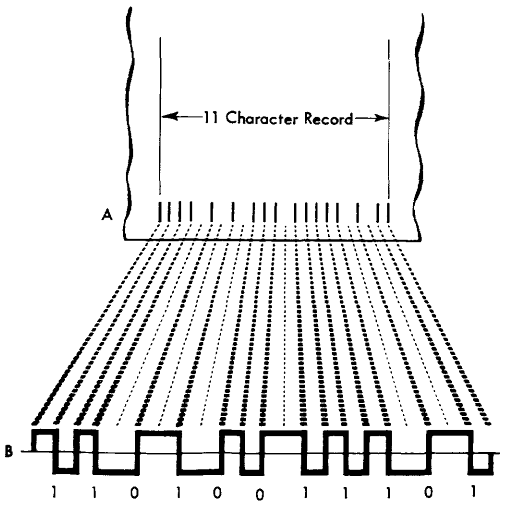


Figure 20. Waveforms for a Series of Bits.

The data head (Fig. 21) is a two-gap write-read head with a .5 inch radius around each gap to hydrodynamically air lubricate it when the tape is in motion.[3] The distance between the write-read gaps is 0.150 inches. The width

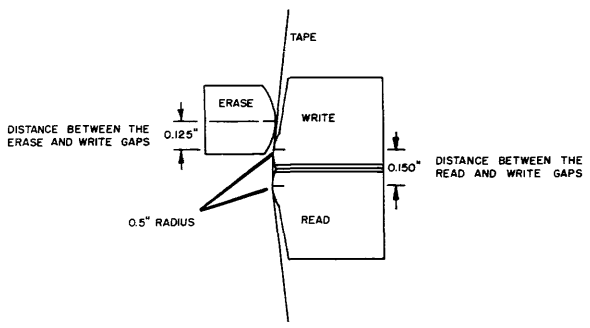

TAPE
ERASE
WRITE
READ
DISTANCE BETWEEN THE ERASE AND WRITE GAPS
0.125"
0.150"
DISTANCE BETWEEN THE READ AND WRITE GAPS
0.5" RADIUS

Figure 21. Data and Erase Heads Configuration.

of the tracks is 0.060 inches for write and 0.050 for read. An erase head (Fig. 21) is located 0.125 inches from the write head and is positioned such that erasure through the base material takes place while maintaining a slight clearance between the erase head and the back of the tape. The erase head is active any time the transport is in write status.

Sensing the amplitude of the recorded signal during writing plays an important part in controlling errors. Should the signal fall below 25% of the nominal level, the record (under program control) is erased and rewritten on a new section of tape.

Error detection and correction while reading are dependent on the redundancy check provided by the two check bits written on tape and the amplitude of the read signal.[4] The two check bits C_0 and C_1 (Fig. 2) are generated during the write operation according to the two equations:

C_0 Equation $C_0 \veebar I_0 \veebar I_1 \veebar I_3 \veebar I_4 \veebar I_6 \veebar I_7 = 1$

C_1 Equation $C_1 \veebar I_0 \veebar I_2 \veebar I_3 \veebar I_5 \veebar I_6 = 1$

Where $\veebar$ means exclusive OR

The bits are divided into three zones from the two check bit equations:

Zone 1 Contains only the bits that appear in equation C_0 and not in equation C_1

Zone 2 Contains only the bits that appear in equation C_1 and not in equation C_0

Zone 3 Contains only the bits common to equations C_0 and C_1

Hence,

Zone 1 Contains C_0, I_1, I_4, I_7

Zone 2 Contains C_1, I_2, I_5

Zone 3 Contains I_0, I_3, I_6

During the read operation, the ten bits of each tape character are checked for parity against equations C_0 and C_1. A single incorrect bit within a tape character will be detected by one or both of the check bit equations.

The check-bit equations determine if a particular character contains an error, and the envelope detector monitors the amplitude of the signal. If the signal drops below a certain level in one track, the track is "dead tracked". This essentially resets that track for the rest of the record. Thus, if a parity error is found to occur in the zone containing the "dead track", the bit in the error correction register is inverted; a zero is made a one, or vice versa.

If two or more tracks in the same zone are dead tracked an uncorrectable error results. However, equations C_0 and C_1 were written so that no adjacent tracks would be in the same zone. Hence, all adjacent double errors are correctable along with some non-adjacent double errors. If a double error does occur, it is more probable it will be an adjacent double error.

There are 45 possible combinations of double errors. Of these 33 are correctable. All corrections are performed on the fly without program interruption or loss of processing time.

REFERENCES

1. L. S. JACOBSEN, R. S. AYRE, "Engineering Vibrations", McGraw-Hill, 1959, p. 158 ff.
2. R. K. RICHARDS, Digital Computer Components and Circuits, D. Van Nostrand and Company, Inc., 1957.
3. H. K. BAUMEISTER, Nominal Clearance of the Foil Bearing, *IBM Journal of Research and Development,* April 1963.
4. W. W. PETERSON, Error-Correcting Codes, The M.I.T. Press and John Wiley & Sons, Inc., 1961.

COMPUTER APPLICATIONS AT THE FRONTIERS OF BIOMEDICAL RESEARCH

W. R. Adey
Brain Research Institute
University of California at Los Angeles and
Veterans Administration Hospitals
Long Beach and Los Angeles, California

1. INTRODUCTION

It would appear that the rapid growth of computing facilities in major life science institutions in this country has brought with it an obligation on those enjoying the privilege of these aids to experimental design and data analysis to view critically the utilization of these methods, as they may determine broad research philosophies in particular problem areas, and thus also, both the conduct and probable outcome of such research.

At the outset, it would appear that there are certain tenets which necessarily govern the optimal utilization of computing analyses of physiological data. Firstly, the analysis should contribute at a conceptual level to the progressive development of that particular field of physiological research. There can surely be no justification for the use of computer analysis to guild poor data in a poorly designed experiment. Secondly, the methods of analysis should be selected as optimal for the particular area of research, and should be tailored in close collaboration between physiologists and mathematicians to suit the needs of the particular research project. It is obvious, therefore, that there must be prior consideration of the proposed methods of analysis in the course of the experimental design. It is scarcely feasible to hope that the serendipidous collection of data in neurophysiological experiments will assure capability for later complex analyses critically dependent on meeting stringent experimental formats. Thirdly, it is both important and highly desirable that the analytic techniques themselves progress in sophistication as the physiological research likewise evolves progressively more complex frames of organization. In other words, it would be hoped that the present major gaps in mathematical techniques appropriate to the analysis of complex living systems would sufficiently challenge the mathematician so that he would contribute significantly to his own field at the same time as he participates in the frontiers of life science research (1).

2. ANALYSIS TECHNIQUES

Our attempts to achieve a truly interdisciplinary approach to problems of information transaction and storage in brain systems have proceeded on the basis of close collaborative activity between life scientists, mathematicians and engineers. These studies have involved novel applications of computing techniques to patterns of electrical brain wave records, including continuous measurement of phase and amplitude characteristics of single wave trains by digital filtering techniques, the use of cross-spectral analyses with calculation of complex transfer functions, and averaging of records during repeated behavioral performance with calculation of coherence functions in cross-spectral examinations. By these and ancillary techniques, such as establishment of probability bounds in cross-spectral analysis, with the use of polar coordinate displays, it has been possible to detect major differences in phase relations between different brain regions in the course of correct and incorrect responses to a learned task. These findings have suggested a stochastic model of the cerebral system, with a possibility that this electrical wave process may play a fundamental role in the handling and storage of information.

This hypothesis has been described in detail elsewhere (5, 6, 9, 10). We have suggested that the rhythmic wave process characterizing activity in many cortical and subcortical cerebral structures may be fundamentally correlated with the ability of these tissues to undergo permanent changes in excitability as a result of their prior participation in specified patterns of excitation. These electric wave processes appear closely associated with the physical overlap of the dendritic parts of nerve cells. We have discussed the possibility that the cell may function as a phase-comparator of waves arising "intrinsically" with those reaching the cell "extrinsically", or that it may exercise this comparator function in relation to complex spatio-temporal wave patterns sweeping across its surface. In either case, the initiation of cellular firing might depend on the precise recurrence of a spatio-temporal wave pattern, to which the neuron had been previously exposed, and which might determine the permanent physicochemical change underlying the "memory trace" (3).

These studies have attempted to define, by techniques not hitherto applied to the examination of electrophysiological data, components in the apparent imprecision of electrical brain wave records which are either phase-locked in leads from circumscribed cortical or subcortical zones in relation to repeated performance of a learned task, or which may have constant phase relations to wave patterns appearing simultaneously in other cortical or subcortical localities, or which may have aspects of phase or frequency modulation on a "carrier" wave train appearing in the course of attentive or discriminative behavior. It is obvious that any one of these three possibilities would provide a frame in which either the nerve cell or the organized cerebral system might adequately "sense" the informational basis of afferent volleys, and the data provide evidence that all three phenomena may have important roles in the various hierarchies from the microcosm of the individual neuron to the macrocosm of complexly organized but profoundly interrelated cortico-subcortical systems.

3. REALISTIC MODELS OF CEREBRAL ORGANIZATION

More importantly, however, some of the evidence from these studies would appear to bear significantly on the fundamental nature of essential cerebral processes, viewed as either deterministic or probabilistic phenomena. For example, the phase patterns in correct and incorrect responses have been examined, and in a population of correct responses, displayed a considerable scatter about a mean phase displacement. Similar, but less extensive data in incorrect responses also showed such a scatter. The calculations of complex transfer functions from cross-spectral analyses strongly emphasize the possibility that there is a stochastic mode of operation in the handling of information in the brain on the basis of a wave process. Such a scheme would envisage the excitability of the individual neuron as depending not only on its previous experience of complex spatio-temporal patterns of waves, as outlined above, but additionally, would suggest that the effectiveness of any subsequent wave pattern in eliciting neuronal firing might depend on its multivariate relationship to an "optimal" wave pattern, capable of inducing firing of that neuron at its lowest threshold.

Such a notion appears to significantly extend the basis for a mathematical model of the neuronal organization of cerebral systems over previous considerations of unicellular systems and nerve nets. There is, as yet, no categoric evidence relating the physiological events in any noncerebral neuronal system, or the phenomena in neurons in natural or artificial isolation from other nerve cells, to the essential processes of information storage. It would appear that a comprehension and utilization of data relating this process to stochastic aspects of both neuronal firing patterns and the wave process pervading the surrounding cortical domain may provide an exciting avenue directly approaching the essential questions of the uniqueness of cerebral structure and function in the memory process.

In a search for other evidence on the patterned relationship between rhythmic brain wave activity in different brain regions, we have examined the aspects of mutual information between two physiological records (11). The mutual information connecting two partially random functions is a general measure of interaction, and can be numerically estimated for physiological data. For two Gaussian processes, the only relations possible are linear ones. Here the cross-correlation function is sufficient to define the mutual information. Correlation, however, underestimates the mutual information for other distributions of amplitudes, or for such physiologically interesting relationships as non-portional responses, or modulation of one signal by another. Lagged mutual information generalizes this cross-correlation function, and can be assembled into a cross-information function, according to the technique of Balakrishnan. Additional insight can be obtained from multivariate mutual information, calculated for several lags at once. The degree of relationship indicated by cross-correlation versus that indicated by lagged and multivariate mutual information in pairs of related electroencephalographic traces has been compared, and has provided considerable additional insight into the subtly shifting phase patterns. Yet another approach initiated in these studies has been the calculation of equivalent-noise bandwidth, which has permitted assessment of deviation from a Gaussian distribution of brief epochs of electrical brain records at varying stages of consciousness, sleep and pathological unconsciousness (8).

These computer oriented studies have led to the construction of a series of realistic models of the cerebral system, which may be considered both in the large, from the point of view of the organization of the major aspects of cerebral systems, and also at the microcosmic level of the single cell. At the level of major cerebral system organization, the studies have disclosed a pacemaking function for the brain waves arising in the deep parts of the temporal lobe, with the induction of similar but not necessarily identical patterns of waves elsewhere in the process of informational recall. The question thus arises as to the relationship of these waves to physicochemical changes in the storage of information (2, 7). Interruption of the wave processes by small brain lesions or hallucinogenic drugs such as LSD has been shown to profoundly interfere with the processes of informational recall. This in turn has led us to the concept that the waves might change in frequency, and thus in pattern, by electrical impedance loading of adjacent tissue elements, including non-nervous tissue.

We have developed a technique for the measurement of electrical impedance in very small volumes of brain tissue in the course of the acquisition of learned habits. By the extensive use of computing techniques, this method permits the examination of brain impedance with currents as small as 10^{-13} amperes per square micron of electrode surface. With the aid of computational analysis, it is possible to detect changes in current flow of the order of 10^{-15} amperes per square micron. This technique is sufficiently sensitive to reveal changes in brain impedance associated with the performance of a previously learned task, and these changes appear only after the learning of the habit (4). These findings have led to the concept that the non-neural parts of brain tissue, the neuroglia, may be involved in the storage of information, and that the "memory trace" may involve a consideration of the interface between the neural tissue and the neuroglial elements. Neuroglial tissue provides a preferred conductance pathway for the impedance measuring currents, since the membrane resistance of glial cells is but one hundredth to one thousandth of that of nerve cell membranes.

The model of the cerebral system proposed from these electro-physiological and impedance studies would therefore have non-linear and stochastic characteristics. This hypothesis of a probabilistic mode of operation would infer that the excitability of the cortical neuron would depend upon the relationship of the spatio-temporal pattern of wave phenomena at the cell surface to an "optimal" pattern of waves for which its firing threshold would be lowest. This "optimal" pattern of waves would be determined by the previous experience of the cell, with the wave phenomena intimately concerned in the physicochemical changes associated with the deposition of the memory trace. It is a characteristic of such a model that it would exhibit a distributed memory function through the totality of those cells which display a participation in a rhythmic wave process. At the same time it would separate sharply from consideration in the storage function those parts of the nervous system, such as the spinal cord, which do not exhibit a rhythmic wave process. It would lay particular emphasis on the interrelations between one nerve cell and another through proximity effects, and between nerve cells and neuroglial tissue, based on interactions through slow graded analog wave processes, rather than the restrictive aspects of pulse coded phenomena.

4. CONCLUSIONS

In conclusion, if we turn away from the specifies of detailed data analysis to the broad philosophies which may guide us into the future in this type of research, we may examine the general purview of the growth of mathematical analysis in physiological research, asking ourselves what may be the most appropriate frame for the conduct of computer-oriented research in the life sciences. It may be argued with some validity that it is far from sufficient for the mathematician to participate on a consultative basis, and that it is necessary for him to be a full member of the research team, closely associated with every phase of the research on a daily and even hourly basis.

In achieving such an arrangement, it is inescapable that the mathematician or computer scientist should find in life science institutions the requisite opportunities allowing him to shape his own career with full and appropriate recognition of the significance of his role by his colleagues in the life sciences.

It therefore follows that the life science institutions should accept the responsibility for programs of technological and mathematical research as an integral part of their own research, to be conceived as of equal merit with their fields of special interest in the life sciences. This is indeed a substantial responsibility, but without it, there is little reason to suppose that adventitious association with limited facets of the mathematical and engineering sciences can provide the basis for a full and vigorous pursuit of those areas which are at the very core of future life science research. It also follows that there is little reason to expect a satisfactory degree of continuing success from programs in these areas of mathematical biology conducted as an appendage to the research programs of institutions whose prime interest is in the mathematical and physical sciences. The substantial nature of the life science requirement in these areas demands that they assume the full responsibility for their development, and that those whom they seek to attract as colleagues from mathematical sciences should feel that these fields are indeed an appropriate vehicle for their life's work. Without such a frame of reference, our early efforts may face a difficult and even perilous future (1).

ACKNOWLEDGMENTS

Studies described in this paper were conducted with the assistance of grants from the U. S. Air Force Office of Scientific Research, the National Aeronautics and Space Administration and the National Institutes of Health.

REFERENCES

1. ADEY, W. R. Data acquisition and analysis in a brain research institute. In Symposium, "Use of computers in biology in medicine". Ann. N. Y. Acad. Sci. In press, 1963.
2. ADEY, W. R., BELL, F. R., and DENNIS, B. J. Effects of LSD, Psilocybin and Psilocin on temporal lobe EEG patterns and learned behavior in the cat. *Neurology, 12*: 591-602.
3. ADEY, W. R., KADO, R. T., and DIDIO, J. Impedance measurements in the brain tissue of chronic animals, using microvolt signals. *Exper. Neurol., 5*: 47-66, 1962.
4. ADEY, W. R., KADO, R. T., DIDIO, J., and SCHINDLER, W. J. Impedance changes in cerebral tissue accompanying a learned dis-

criminative performance in the cat. *Exper. Neurol., 7*: 259-281, 1963.

5. ADEY, W. R., and WALTER, D. O. Application of phase detection and averaging techniques in computer analysis of EEG records in the cat. *Exper. Neurol., 7*: 186-209, 1963.
6. ADEY, W. R., WALTER, D. O., and HENDRIX, C. E. Computer techniques in correlation and spectral analyses of cerebral slow waves during discriminative behavior. *Exper. Neurol., 3*: 501-524, 1961.
7. ADEY, W. R., WALTER, D. O., and LINDSLEY, D. F. Effects of subthalamic lesions on learned behavior and correlated hippocampal and subcortical slow-wave activity. *A.M.A. Arch. Neurol., 6*: 194-207.
8. RHODES, J. S., BROWN, D., REITE, M., and ADEY, W. R. Computer analysis of chimpanzee sleep records. In Symposium, "Sleep Mechanisms", Lyons, France, 1963.
9. WALTER, D. O. Spectral analysis for electroencephalograms: mathematical determination of neurophysiological relationships from records of limited duration. *Exper. Neurol., 8*: 155-181, 1963.
10. WALTER, D. O., and ADEY, W. R. Spectral analysis of electroencephalograms recorded during learning in the cat, before and after subthalamic lesions. *Exper. Neurol., 7*: 481-501, 1963.
11. WALTER, D. O., and BROWN, *D.* Mutual information of two physiological records. *The Physiologist, 6*: 293, 1963.

COMPUTERS AND THE ADMINISTRATION OF JUSTICE

Richard F. C. Hayden, Judge
Superior Court for Los Angeles County, California
Los Angeles 12, California

Lawyers and judges have been thinking about using computers to help in their work at least since 1950 and active research is only a little more recent. Today there are several working research projects going on across the country and at least one of them has been productive of an extremely useful working product as well as theoretical information. There is a Special Committee on Electronic Data Retrieval of the American Bar Association and it is no longer accurate to say that most lawyers have never heard of the idea. On the other hand it would be misleading to suggest that active participants in this field constitute more than a tiny fraction of even that minority of lawyers and judges who are inclined toward theoretical speculation. The momentum is such however that even in the most depressive swings of my personality cycle I no longer suspect that it is all a fantasy. If Western industrial civilization does not destroy itself, computer technology with its concomitant theory will become an integrated instrument in the American legal system and will make significant contributions to the administration of justice.

This is not to say there are no problems. My qualification to speak about them is solely that I have talked about them with dozens of experts in your field and have never yet had one fail to tell me that the problems of applying computers to legal problems were closely analogous to those of most other fields of application. So you all can supply the portion of my talk that should be devoted to "problems" from your experiences in other fields.

The principal focus of my remarks will be on possible applications of computers to the management of records of trial courts and to the use of computers in the making of management judgments in the operation of trial courts. These are relatively undeveloped fields of thought or application, however, and before taking them up I should like to note briefly some of the other areas in which lawyers have felt or demonstrated that computers could be of help.

COMPUTER OUTPUT AS EVIDENCE

Obviously computers, their operations and their products are all facts. In the trial of a case relevant facts are usually admissible. Therefore in an appropriate case the output of a computer operation could be admissible as evidence to prove any relevant fact. If this is too obvious I apologize, but it is often overlooked by lawyers and, when it is considered, it frequently is seen as a very difficult procedure. Back in 1952 I had no difficulty in persuading a United States District Judge in San Diego that he should permit me to prove the results of a statistical analysis by an IBM electromechanical installation. It involved the processing of 10,000 Veterans Administration loan guarantee applications and without machine processing I am quite sure we would never have attempted to make the analysis or prove its results. It was very effective in winning the trial, incidentally.

COMPUTERS TO PREDICT LEGAL CONDUCT

The behavior of judges as judges, jurors as jurors and parolees as parolees are all instances, also, of human behavior and as such are susceptible to the types of analysis available elsewhere in the behavioral sciences. When the computations become complex, and they very frequently do, it is manifestly useful to put the data into computers. Very useful results have been achieved by the California Adult Authority in predicting the conduct of parolees by this means. These have shown a significant improvement over informal intuition. A number of studies have developed quite promising methods for predicting the voting conduct of Supreme Court justices. As yet I know of no attempt to predict the in-court conduct of trial court judges, such as myself, or of jurors, such as any of you might be, but both of these have been suggested and in all probability are ultimately susceptible of as accurate estimates as either appellate justices or parolees.

LOGICAL ANALYSIS AND SIMULATION

If "the life of law has not been logic but experience" nevertheless even Justice Holmes who said this, would have conceded that logic has always had a place in that life. Digital computers operate on a system of logic into which at least some legal concepts can be translated and then be analyzed for internal consistency. A model for such a procedure was worked out some years ago in connection with the drafting of an insurance contract. It is said of insurance contracts that, what the large print giveth the small print taketh away. Perhaps computer analysis would make more explicit the real nature of the agreement.

Substantive fields of law are extremely complicated systems. An amendment of one portion of the Internal Revenue Code, for instance, can have quite unpredicted effects elsewhere in the revenue laws or even in the field of contracts or of copyright. It is possible to translate the concepts of a legal system into machine manipulatable form and simulate the effects of amendments throughout the system. This is a wholly feasible procedure. I do not know that it has been applied.

MACHINE PREPARATION OF DOCUMENTS

Once again perhaps I should apologize for presenting a quite trivial example; I do so however because it is a case where several law firms are currently using components of computer systems to do work-a-day tasks because they have tried them out and found them economically advantageous. The preparation of many legal documents such as wills, trusts, contracts and so forth is usually done by dictating appropriate collections of form language. I will not take your time to argue, as I could, that when this is well done the assemblying and modifying of such documents requires a high skill. Whether creative work or hack work, it involves the typing of collections of largely standard text and then the retyping of largely the same text material with intermittent corrections. Obviously, the tape driven and tape producing flexo-writer typewriters used as input components for some computer systems can be used as a cheap and reliable substitute for scissors and paste and in the redrafting of the documents can reduce the drudgery of proofreading enormously. In the two offices about which I have first-hand knowledge, the first such typewriter was followed in less than a year by the purchase of at least one more.

LAW LIBRARY RESEARCH—INFORMATION RETRIEVAL

By far the greatest amount of speculation, experimentation, and actual development work on the use of computers in law has been in the field of information retrieval. When a lawyer speaks of "research" he means library research. It is in this area that most of us first considered the likelihood of using computers. Our problems are not different in principle than those of any large information field. When we read the articles in *American Documentation,* for instance, they seem quite relevant to our problems.

Our existing library tools seem very defective to us as we work with them but when our discontent makes us study the matter we find that we have a better collection of existing library research tools than any other learned discipline. This is doubtless the result of certain characteristics of our library that are of interest in considering how we might use computers to help ourselves. First of all our collection of information has a very large number of inquiries made of it. Second it is a very large collection which has a high, though not exponential acquisition rate and a very slow rate of obsolescence. Storing and indexing of only current and future acquisitions would probably be of too little use to justify a commercial venture so limited. For this reason the principal obstacle to any extensive testing has been the high cost of putting material into the system. But in spite of these problems a number of trial systems are in existence. There are in service already at least two indexing services using the KWIC program. The most impressive venture to date is the collection in machine readable form of the language of the entire text of the Pennsylvania statutes and of the statutes pertaining to health law of fifteen states. This material is currently being searched on a computer for statutes by calling out appropriate words from the unedited text. It has been of practical help not only for library research, but in legislative drafting.

It may seem like a crude and unpromising procedure but there are many of us who think that it is very likely that in the end the most satisfactory system of searching legal materials will be just in this fashion of searching through the entire body of unedited text for appropriate natural language words or phrases in promising combinations. We are not unaware of many theoretical difficulties that stand in the path of making such a system effective, not the least of which is the intentionally non-uniform vocabulary that characterizes our largest and perhaps most important corpus, the appellate court decisions. Nevertheless the great advantage of having indexing designed by the information requester according to his actual needs rather then predesigned by an editor who must guess what the requester will want, seems to us to justify valiant efforts to develop strategies for searching the raw text.

COMPUTERS TO INDEX THE EVIDENCE FOR A TRIAL

In routine litigation the lawyer's notes and memory are generally adequate to enable him to collect, marshall, present and argue from the evidence. But when a case gets up to say two weeks in length and most particularly when it involves the use of many documents the indexing problem becomes most burdensome. In the truly large anti-trust case, it is a fact that there is no lawyer on either side who has actually studied each bit of evidence available to his side. Indexing then becomes a necessity. Though not widely used, systems for indexing the evidence of large cases by both computer and by the analogous punched card have been developed and there is little question that they will be increasingly used.

Having outlined some of the more important areas where lawyers have believed or proved that computers could help them, I turn to my principal topic, the use of computers in the administration of justice in the trial court.

COMPUTERS TO STORE AND SEARCH TRIAL COURT RECORDS

The trial courts of our country receive for storage and internally generate and store staggering numbers of words each business day. Take for example the Superior Court of the State of California in Los Angeles County, where I participate in this massive pack-rattery.

A recent civil case number in the Los Angeles Superior Court was 800,000. This numbering system does not include divorce, probate, criminal, and a number of other types of cases.

An examination of a not untypical civil case file on my desk at the time I wrote this talk showed that it contained 156 32-line pages of typed material on 8½″ × 13″ paper. When the cover sheets, copies of the court's minutes, affidavits of mailing and primarily printed court forms are added to these it measures 1⅛″ thick at the binding. This case file did not yet include the findings of fact, conclusions of law and judgment.

In addition to the case file of court *documents* there is maintained:

(1) An index to all cases by the names of the parties;

(2) A docket which is a sort of ledger of all court transactions; and

(3) A minute book which is comparable to the journal in an accounting system.

I would not venture to guess as to the number of millions of words that are preserved by our County Clerk nor as to the number of new words added to his store every day.

But these words have certain characteristics in addition to their numbers. First of all they are often very repetitive and are made up largely from a restricted vocabulary and are ordered, generally, by a somewhat special syntax. Moreover, they are capable without loss of meaning of being made more orderly. Secondly, the collection is consulted very frequently not only by court personnel, but by the public at large. Thirdly, the rate of obsolescence is theoretically zero and for practical purposes is indeed very slow.

Because the persons who use the indexes are accustomed to a pretty blunt instrument, and because the search is typically for one special case file only, a purely prospective program for storing and indexing the materials would probably be quite acceptable so that input could be integrated into daily operation of the system.

The users of the existing index are not accustomed to paying anything for the service they now get, and since the court is not expected to make a profit, the rationalization of a computer storage system will doubtless have to be reduction in operating costs or significant improvement in performance by the court.

For the last two years the Los Angeles Superior Court, under the direction of our Presiding Judge, McIntyre Faries, has been cooperating with System Development Corporation and the Law Science Research Center at UCLA in a study of our system to determine what kinds of economic or social payoff might be expected from the use of computers on our records or in our management. Mr. Eldridge Adams has conducted the study for SDC and Professor Edgar A. Jones, Jr., has represented UCLA on the project. I have been the chairman of the committee of judges who have been directly involved. The study is now being completed. It would not be appropriate for me to comment on it in detail but my reaction at the moment is quite consistent with the generally optimistic tone of this talk.

I have said our present indexing is unsatisfactory. This is how it works:

If I know the name and number of a case I can in about 20 minutes of walking, searching through docket volumes and waiting at file counters, get access to most of the information about that case, although to be certain of each court action I might need to spend twice or three times this amount of time hunting through minute books.

This is a typical access time to the records of a particular case when I know its full address, that is, the case number. It may be half again as long if I know only the name of a party.

This is the nature of the problem of searching our Superior Court records if I know the name of one or more of the parties. If I do not know this or the case number there is *no* way to get access to the file. It does not help me to know the names of both attorneys (assuming their private records are not available), the judge, every witness, and even the names of the bailiff, the clerk and each juror. It does not help me to know where the accident happened or where the parties were married or what the lawsuit was about. Unless I find someone who can say, "Oh yes, that was Smith vs. Jones" I will never find the file or its contents.

I cannot by any means find out from the court records who has sued or been sued for a swimming pool accident or what lawyers end up representing most of the defendants who are bailed out by a particular bail bond broker.

The probation department, both adult and juvenile, the conciliation counselors and the mental health counselors of the court collect a great deal of detailed information about individual cases. At the present time this is not being correlated in any significant fashion and the task in Los Angeles County is probably beyond the capabilities of any hand-operated system.

Let us assume that we could index the material presently being collected by the courts on a variety of different dimensions, for instance, in addition to the individual parties to litigation we could index to the names of witnesses, attorneys, jurors, the judges, receivers, executors, and any other name which appeared in the file. Similarly we could index by types of persons. It would be interesting to know, for instance, some of the characteristics of the parties to litigation, and certainly of jurors, and there might well be similar areas of interest relating to witnesses, lawyers, and judges. Indexing by the aspects of the litigation other than the parties is at least conceivable, for instance, by subject matter, by procedural steps, even by minutes of proceedings and by judgments either in civil, criminal, or other proceedings both as to type of judgment, and amount of judgment. It is not at all difficult to envision the procedure for indexing the names of individuals involved, or of assigning to them the characteristics in which one is interested, such as, occupation, race, or similar classifications. The apparent difficulties in the way of indexing by procedural aspects are not as great as they would appear. The Probate Court of St. Louis County, Missouri, is at the present time making all minute entries by pre-punched IBM cards which are so coded as to punch out, not only the text of the minute order, but the fee involved in that proceeding. The purpose for which they have adopted this practice is primarily an accounting one, and it is not entirely clear to me what complications it has created in the functions of the clerk's office, but a cursory examination of the forms which have been provided to me suggest that it probably simplifies the clerk's function. Manifestly, if this aggregate detail can be placed in machine readable form, virtually every step of the court's activities could be made accessible.

Some of the possible products of such accessibility that occur to me, are the following:

At the present time in the Los Angeles Superior Court there are two privately operated services, one of which gives information about the individual members of the jury panel to plaintiff's attorneys, and the other which gives similar information to defense attorneys. There have been some complaints about the existence of these services, but they are now well established and have their duplicates in the District Attorney's office and most of the busier prosecutive agencies. There is no similar service to the best of my knowledge for defense counsel in criminal matters. Manifestly, the court itself could collect this kind of information quite automatically if it recorded in machine readable form the names of trial jurors on cases, and if the service is available to one side of criminal litigation, there would seem to be some arguments for making it available to both sides. These jury "books" typically contain a small amount of background information on the individual juror, and a record of the cases on which he have acted, and occasionally specific items of information which have come to the attention of the lawyers which tend to disclose a specific bias.

If you assume the desirability of a jury "book" is there not an equal need for a judge's "book"? It is manifestly impossible for any attorney to have detailed knowledge from first hand experience about each of the 120 judges who regularly sit in our Superior Court, and certainly some estimation of how a judge would be likely to perform in a given case can be made with equal reliability based on years of recorded performance as can be done for a juror based on only a handful of trials.

In this latter connection, there is a great deal of current interest in the bar and among judges in the problem of the disparity of sentencing. We are all conscious of the fact that individual judges vary extremely in various kinds of cases as to the sentences which they impose, and while I have never heard any lawyer argue for absolute uniformity of sentences, really gross disparities suggest that injustice must have been done. It would be extremely interesting to any judge sitting in a criminal department to be able to compare his sentencing patterns with those of his colleagues. Similarly, it would be extremely interesting to qualify the known disparity in sentencing in various parts of the country.

Even more troublesome than the question of the disparity of sentencing between judges, is the fact that the individual judge has virtually no means of determining afterwards what the effect of his sentence was on the individual defendant. He imposes the sentence and only in very rare cases does he thereafter hear of the man again. A similar problem is presented in other types of judgments which have a continuing effect, such as child custody orders. If the sentencing judge periodically had fed back to him the record of what has happened to the defendant after a sentence, it certainly could be expected that the judge could be benefited by the "feed-back." This is particularly true in the case where the judge experiments in a questionable matter with a probation type sentence.

At the present time, many courts keep records of the productivity of the individual judges. This could be done in much more detailed fashion and with greater ease by data processing procedures. It would be interesting to observe, for instance, how frequently judges resubmit cases in jurisdictions such as California where the payment of the judge's salary is dependent upon the fact that he not have any matters under submission for more than 90 days. Similarly, it might be possible to evaluate the effectiveness of individual judges in particular kinds of proceedings such as pretrial hearings.

If the individual judges are thus to be revealed in that white light of subsequent analysis, is it not equally fair that attorneys be similarly treated? Conceivably, the statistical analysis of the judgments obtained by individual lawyers might blast some now impressive reputations. Likewise, by being able to get access to the files in which an opposing counsel had been involved, an attorney might very well obtain some valuable insights as to the likely strategy of his opponent. There have been instances when it was rumored that a substantial amount of criminal business was obtained by attorneys as a result of illegal arrangements with bail bond brokers. A comparison between identity of bail bond broker and the attorney in a number of cases might show up interesting correlations that could be the basis for further investigation. Likewise, the suspected relationship between attorneys and doctors in personal injury matters, and in adoption matters, might similarly be revealed or refuted.

There are some persons whose occupation takes them into Court as witnesses on many occasions. It would be a favorable aid to an advocate to be able to obtain information as to testimony given by such persons in previous or similar cases. I know of one instance of a well-known forensic scientist whose testimony as to his technical qualifications was completely destroyed in one trial, and yet to the best of my knowledge he has continued to testify in many cases without this issue being raised again.

The discussion earlier in this talk of the prediction of the Supreme Court Justices' rulings suggests the possibility of prediction of the dollar value of trial court cases in advance of trial. Every lawyer, with greater or lesser skill, engages in the practice of placing a value on the cases which he tries, that is, endeavoring to predict the dollar value of judgments either by a court or by a jury. There is a surprising amount of agreement among experienced judges and attorneys on a case-by-case basis of the value of any particular claim, just as attorneys accustomed to consider constitutional law do particularly well in predicting the conduct of the Supreme Court without resorting to statistical procedures. Nevertheless, even the limited success of the decision-predicting techniques developed to date suggests that more sophisticated procedures might result in greater accuracy than is given by the "hunch" of the specialist, and I can picture the court itself using the results for increasing the number of settlements by providing attorneys with a range of values for particular litigation. This would be most likely to be possible in personal injury cases and in domestic relations support orders.

Courts are public agencies and are accountable for the over-all quality of their performance to the public and its representatives. This is not inconsistent with the demand that judges should be independent of pressure or public passion or prejudice in their decisions of cases. Case by case we have a duty to be independent; over all cases we are accountable for the quality of our performance. My colleagues can be a testily independent group of men, I am happy to report to you, and they can respond with vigor to what they feel is uninformed criticism. I have never heard one who suggested that we were not subject to informed criticism, however. Over and over again the criticisms we receive are purported to be based on statistics. We examine the numbers and the logic of the statistics and usually we see that they are only a part of the story. Then we start the laborious task of assembling the meaningful numbers and weeks later we are able to satisfy our critics, or at least the persons to whom we must account that the criticism as first phrased is not helpful. In the meantime the impression has often been created that in some particular where, in fact we are not subject to criticism, we have done wrong.

This is not to suggest that we are never wrong, nor more so to suggest that statistics are not valid instruments by which to demonstrate this fact. It is to say that in a contest between statisticians it is important to have a rejoinder ready at hand if the war is not to be lost in the first battle.

The data required to analyze our performance, or to help us in our administrative decisions, is to be found in our records. Or at least that is where we, and those who analyze our performance, have routinely looked for them. Experience has made us accumulate, on a routine basis, more and more detailed information about our activity. But, this data collection is over and above our routine information storage procedure. In other words we do not record the significant events in an automatically retrievable form, but instead record it one place as a record and at another place as an item for statistical analysis. This has the readily recognized but actually minor disadvantage that it requires extra personnel. The real flaw in this procedure is that it requires you to decide before you collect the data what events you believe will be relevant and significant. This decision must be made, of course, before calling out from a computer-readable store, data which has been stored as a part of record keeping, but the time delay could be enormously reduced and the varieties of choice for analytic procedures would be much greater.

COMPUTERS AS AN AID IN COURT ADMINISTRATION

The use of statistics to criticize or defend existing procedures is only a contentious variation on the much more important function of analyzing and selecting between procedures. Clearly greater and more rapid access to the facts of our operations should improve our understanding and enable us to improve upon our methods.

The possibility of using such simulation procedures as PERT for analyzing the work flow of a large court calendar has been suggested by a number of persons and I believe has some merit. It has not been applied as far as I know There are two possible problems or limitations on its usefulness that should be noted. The first of these is the relation between the cost of obtaining the input time range estimates and the value of the unit being estimated. It is no particular expense item to add one engineering man day per quarter to keep tabs on a million dollar component. It is a very different thing to add a similar expense to the cost of a fifteen thousand dollar law suit.

The second reservation I would have is more fundamental. There is a serious doubt in my mind as to the extent to which the important problems of running our courts lie in the area of queveing or calendar programming. The big payoff, I suggest, would be in increasing the amount of time the judges and the jurors can be sitting in the courtroom and increasing the amount of work they can get out in a unit of courtroom time.

We have the operations research tool and we should see if we can use it profitably, but I suspect it is not a tool designed to solve our biggest problem of delay in court.

PROBLEMS AND PROMISES

It would be presumptuous for me to tell this group what the problems are likely to be in implementing the use of computers for trial court record keeping and management, but I will simply list some that occur to me. First the problems to be solved in order to get a satisfactory system in operation:

(1) Rationalizing the cost.

(2) Deciding whether to convert to machine readable form prospectively only.

(3) Meeting the problem of inconveniences to the inquirer.

(4) Integrating the system with those of related agencies such as Police, Probation, Adult Authority, etc.

(5) Maintaining security of closed files.

(6) Developing effective programs for (a) indexing, (b) statistical analysis, and (c) prediction.

Second the problems created by greater access:

(1) Difficulty of creating useful statistics without having them be used unfairly.

(2) Increasing pressure for undesirable uniformity in sentences or other judgments.

(3) Too much information about individuals eroding away the privacy of a large society.

(4) Unfair comparisons between productivity of individual judges.

What are the likely social benefits of using computers in the trial courts?

Economic savings? Perhaps, although I am a little skeptical. It has been suggested to me that in other computer installations the sale has been made on the basis of potential savings which have seldom been realized and the payoff has been in increased information.

Increase in public confidence? We want our court to be a little conservative but not too much so. If they are able to integrate the computers effectively, it may tend to make us feel that they are still able to deal with the problems of society.

Reduction of the delay in the court's calendar? Maybe a little. But, as I noted above, my suspicion is that the cause of delay is not in the calendar department, but in the trial department. A really good record of how we do our business would, if my suspicions are correct, force us to stop focusing on the calendar flaw and turn our attention to how we try our cases.

Behavioral science insights? This could well be the biggest benefit of all. If access to our records could increase our understanding of why the parties ever came to court in the first place, we might well be able to address the problems of court congestion at its social source. Historically, courts have focused their attentions on the most difficult and pressing then current social problems. This is inevitable given the court's function of determining otherwise unresolvable conflict. The court's business today is primarily: with the automobile and its 40,000 fatal accidents a year; with the disintegrating American family (in Los Angeles County last year there were 44,922 marriage licenses issued and 37,535 Domestic Relations actions filed—I make no guess as to the effect on either of these numbers of having Las Vegas as conveniently close at hand as it is); with crime and its essentially incalculable cost which is only suggested by the permanent presence in our prisons of nearly one-half million men and women and a nearly equal permanent jail population.

Clearly the source of our crowded calendar is in our crowded society. We cannot complain of this. It is our stock in trade. But we could hope that if we understood better the cases we prove, we could reduce them to more manageable numbers. We judges have no fear of automation as threatening our security of employment.

BIBLIOGRAPHY

M.U.L.L.
This quarterly publication of the Special Committee on Electronic Data Retrieval of the American Bar Association in collaboration with Yale Law School is the best current source of information in this field. Available from the American Bar Association, Chicago, Illinois.

Law and Contemporary Problems, Vol. 28, No. 1, Winter, 1963. Published by the School of Law, Duke University. This entire issue (270 pages) entitled "Jurimetrics" is on topics related in part to computers and the law.

LAYMAN E. ALLEN, ROBIN B. S. BROOKS, and PATRICIA A. JAMES, *Automatic Retrieval of Legal Literature:* Why and How (1962).

EDGAR A. JONES, JR. (Ed.), *Law and Electronics: The Challenge of a New Era.* Matthew Bender (1962).

THE COMPUTER IN EDUCATION: MALEFACTOR OR BENEFACTOR

Robert L. Egbert
System Development Corporation
2500 Colorado Boulevard
Santa Monica, California

INTRODUCTION

When "computers" and "education" are mentioned together, one might visualize the computer as a tool for scientific research, as a teaching aid for instructing students in the use of computers, or as a device for helping in the business operations of an educational institution. In these three roles, the computer has amply demonstrated its utility. Few would question that the computer is now an important adjunct of the university research program; that computers in schools are necessary for teaching our future generation of computer engineers, programmers, and computer users; or that a computer can do much to facilitate and to permit integration of the various business activities of a large school system.

Although these have been the first uses of computers in education, investigators more recently have been studying the possibility of employing computers in instructional and organizational-administrative aspects of education. The explorations of these new applications of computer technology have been primarily stimulated by some recent developments in instructional methodology and in organizational patterns. The characteristics of these educational innovations must be understood before the computer applications can be discussed. Therefore, this paper deals with (a) new developments in instruction and organization of schools, (b) the role of computers in implementing these developments, and finally, (c) the implications of using computers in education. Although there is no sharp line of demarcation between the instructional and non-instructional aspects of education, for clarity we will discuss these topics separately.

INSTRUCTIONAL ASPECTS OF EDUCATION

Recent Developments in Instruction

In considering a school, most of us envision a class of 30 to 35 students, the teacher at her desk or the blackboard, and the students at their individual desks. But modern technology is beginning to provide equipment that may well change this configuration. We shall discuss three of these technological innovations that are particularly pertinent to the topics of this paper: programmed learning, educational television, and language laboratories.

Programmed Learning. Programmed learning or, as it is more popularly called, the teaching machine movement, has a very brief history, briefer even than that of the digital computer. As early as the 1920s S. L. Pressey[13, 14] published articles describing test-scoring devices which also performed teaching functions. Relatively little interest was attracted, however, to the field until 1954 when B. F. Skinner[17] published his article "The Science of Learning and the Art of Teaching." In this article Skinner described a device " . . . about the size of a small record player" which presented problems or questions, gave immediate reinforcement for correct answers, recorded errors, and made provision for the individual to progress at his own rate. The programmed learning movement, as it has developed over the past few years, can really be said to have had its beginning with this paper.

By programmed learning, we refer to the autoinstructional procedure wherein the student studies materials (programs) which have been carefully prepared and organized in sequential steps with each step being set off in a separate rectangle called a frame. Some frames contain statements together with questions based on these statements; other frames simply contain statements; and still other frames contain answers to questions. The frames appear one at a time either in a text or in a machine viewer.

Programmed learning involves four features: first, it presents the material in a carefully organized, closely controlled manner; second, it asks questions; third, it provides for student response; and fourth, it provides a way for the student to check his answers. Since the material in programmed learning is autoinstructional, the student proceeds at his own rate. Thus, the bright student who reads and computes quickly is able to move ahead more rapidly than those students who do not. On the other hand, students are not forced to go on to new concepts until they have fully mastered previous ones. The provision for student response forces the student to be an active participant in the learning situation and prevents him from simply daydreaming his way through the material. Providing the student with the answers to questions gives him immediate feedback about the correctness of his response.

Programmed learning is essentially of two types, linear and branching. In the programmed-learning field, linear programs are programs in which each student goes through the same sequence of frames. Branching programs are those in which different students may study different sets of frames depending upon their answers to questions in the programs. Like the tutor, the branching program responds differentially to individual students. Thus one student who learns the material very easily may study only a relatively few frames. Another student who has some difficulty with a particular part of his material may be branched through remedial sequences to give him a fuller understanding of this section. A third student may be able to answer almost all of the questions correctly but when asked how well he thinks he is doing indicates that he would like to have some remedial work on either all or part of the material covered. In this instance, he is branched by his own request to new presentations of the same material. Programmed learning, then, introduces a new concept to education: the student can study material so prepared as to give individual tutoring with a minimum of attention from the teacher.

Educational Television. The term "educational television" refers to a great range of activities. It may be either open or closed circuit. It may be broadcast from a large studio in a metropolitan center and carried over a number of channels in many different cities, or it may be televised from a small room in a local school district and received by only a few receivers in the schools of that district. It may be a yearlong course in some subject such as physics, or it may be a series of relatively unconnected 20- or 30-minute programs. The producer-director may be a person with relatively limited experience in the local district, or he may be a career professional with many years of experience in a metropolitan center. The actors may be teachers whose primary responsibility is classroom teaching with television being a side line, or they may be paid professionals who may or may not have appeared as a teacher in a classroom.

Some local districts own their educational television facilities; others use the programs broadcast from other sources; and still other districts make no use at all of educational television. An example of a small school district that uses television rather extensively is the Weber District in Ogden, Utah. The Weber District has 17¾ hours of television time available every week at the elementary level and would like even more. For any given grade level this amounts to from two to four hours per week. The basic philosophy of this program is to enrich, supplement, and motivate. In very few cases does television become the total teacher. In fact, Weber District personnel feel that the success of even the best television programs depends upon the role that the teacher plays in integrating the program into the course.

In addition to the open circuit which broadcasts elementary and junior high school programs, in each of its two high schools the Weber District has a closed circuit which is used in English, social studies, science, and mathematics classes. In the Weber District and elsewhere, students give indication of liking television instruction, not as a steady diet, but as it is currently used they feel it adds materially to their education. Moreover, studies indicate that children learn well using television.

Language Laboratories. A language laboratory is a facility in which a student can receive individual instruction in both speaking and listening comprehension of a foreign language. The students listen to a foreign language spoken by a model, presented through a series of tape recordings. The student can also speak and compare his speech against a criterion. In some instances, his speech is recorded and he can compare the recording against the criterion. In other instances, he simply speaks and hears his own voice directly against the recorded criterion.

Language laboratories have proved to be very successful in teaching listening and speaking, and they are widely used to supplement the live teacher in foreign-language instruction.

In most schools at the present time the three innovations just described—programmed learning, educational television, and language laboratories—are used in fairly traditional classroom settings. However, revolutionary uses, including computer control of these instructional devices, are being considered. Research projects involving experimentation with such uses are described in the next section of this paper.

Role of Computers in Implementing Instructional Innovations

Although a number of organizations, including International Business Machines,[18] Bolt Beranek and Newman,[12] and Thompson Ramo Wooldridge[3] have been experimenting with the use of computers in instructing students, we shall limit our discussion to the activities of two of these groups—System Development Corporation and Coordinated Science Laboratory of the University of Illinois.

CLASS. CLASS stands for Computer-Based Laboratory for Automated School Systems and is a facility developed and operated by System Development Corporation.[5] The CLASS facility is not considered a prototype of a school system but is devoted rather to educational research in a system context. The laboratory permits the integration of (1) individual student automated instruction, (2) group instruction, both automated and conventional, and (3) centralized data processing for administration, guidance, and planning functions. For flexibility, the central control of the laboratory is exercised by a digital computer, the Philco 2000. Various innovations in instructional load, administrative data processing, and counseling procedure can be provided by computer program modification. The facility itself consists of an administrative area, a counseling and observation area, and an automated classroom area. Individualized automated programmed instruction is made possible through electronic components on each student's desk. Each student has two components, a film viewer and a multiple-choice response device, connected to the computer through a buffer system. Each student response is checked for correctness by the computer, and the appropriate feedback message is transmitted.

A cumulative record is kept of the student's performance on each of the topics. These performance measures include the student's response time, error count, and pattern of errors. If a student's performance falls below an acceptable level for a topic, the student is branched or detoured to a special set of remedial items on that topic.

The student may be taken through the entire remedial routine or may be brought back sooner to the basic series if he performs well on the remedial set. If he makes a good record on any given topic demonstrating by his overt behavior that he understands that topic, he may not be branched at all but may proceed rapidly, skipping some material as he proceeds.

In the CLASS facility, the teacher has four sources of information available to monitor student learning behavior: (1) a teacher's display console, (2) an educational data display, (3) a response device similar to the student's unit, and (4) a film viewer for monitoring the educational program.

The teacher's information sources permit him to locate students who are having difficulty, call up supplementary information about students, and to follow a student step by step through his lesson material. To initiate a period of automated, individual instruction, the instructor inputs the student assignments from punched cards and activates the first action button. This causes the student history tape to be brought into the computer memory indicating the student's IQ, his previous grade record, and where he is in his lessons. Another magnetic tape is read into the computer assigning the day's lessons to each student. With the assigned lesson thus available the student can begin his work with the teacher monitoring via his automated devices.

As has already been stated, CLASS flexibility provides students with the opportunity of working individually at self-paced speeds on different material, in large groups or in mixed groups where some are individually tutored and others receive group instruction. In all cases, however, continuous computer analysis of student performance provides the teacher with cross-diagnosis of student weaknesses. The teacher is thus liberated from most record-keeping and paper-grading, no matter what the mode of instruction.

In the group mode of instruction, the TV screen and speaker serve as a common group output for different kinds of audio-visual presentations. The audio-visual system incorporated in CLASS is modeled after one proposed for the University of California at Los Angeles, which offers remote controls and slide and film projectors. This system enables the teacher to (1) call up films or other displays from the front of the classroom, (2) stop the display wherever appropriate for group discussions, (3) repeat short film strips or sequences of film when needed for repetitive learning, and (4) select a particular slide on command. It also provides the important capability of scheduling and calling up displays at the moment they are appropriate to the topic being taught. In short, a laboratory such as CLASS enables us to try out educational ideas and developments before implementing them on a large scale. Techniques and procedures can be tested frequently enough to determine which procedures will be most applicable to the actual school setting.

PLATO. PLATO, which stands for Programmed Logic for Automatic Teaching Operations, is another laboratory for educational research.[1] It uses an ILLIAC computer-controlled system of slides, TV displays and student response panels for simultaneously instructing a number of students. (Initial work has been with only two student stations.) It is part of the Coordinated Science Laboratory of the University of Illinois.

In the PLATO system, the computer accepts each student's request in sequence, but because of its high speed students can be served without noticeable delay. Each student has a key set and a television set. The key set, used by the student to communicate to the central computer, has keys of two types: character keys, such as numerals, letters and other symbols; and logic keys such as "continue," "reverse," "help," "judge," and "erase." These logic keys are used by the students to proceed through the instructional material. From the student's response or from the position he has reached in the program, information is selected by the computer for display on the student's television screen. This information has two sources: (1) a central slide-selector, and (2) an electronic blackboard. Although a single slide-selector is shared by all students, each student has independent access to any slide. Each student has an electronic blackboard on which the central computer can write characters and draw diagrams. Information unique to a student, such as his answer to a question, is written on his own blackboard by the computer. The images from the slide-selector and the blackboard are superimposed on the student's television screen.

Each lesson is divided into a main sequence and a number of help sequences. The main sequence is designed as a minimum instructional program which is presented to all students. Help sequences are available for students who need assistance in answering questions from the main sequence. These help sequences typically present further information and leading questions which are helpful in finding the correct answer to problems from the main sequence.

The student may use the "continue" and "reverse" buttons to move forward and backward through the material. However, when a question is asked by the computer, the student must answer that question correctly before he is allowed to proceed. After inserting the answer to a question, the student pushes the "judge" button and the computer writes "OK" or "NO" next to the answer. If the answer is correct, the student may proceed. If the answer is incorrect, the student may erase his wrong answer and try again. Or he can ask to see the help sequence by pushing the button labeled "help."

While the student is proceeding through the lesson material, the computer is keeping detailed records which are essential for evaluating instructional material and for providing information for future system design.

ORGANIZATIONAL AND ADMINISTRATIVE ASPECTS OF EDUCATION

As has been indicated in the preceding section, such innovations as educational television, language laboratories, and programmed teaching have opened the way to alternative instruction approaches. Unfortunately, the traditional school cannot be readily adjusted to accommodate these innovations. With its students divided into classes all the same size, lock-stepping their way through the curriculum, the typical public school has maintained a structure in which only minor modifications are possible. Furthermore, even minor changes in an ongoing program often face major obstacles which prevent their implementation.

Problems associated with innovations are not unique to education; large industrial and military organizations have faced similar difficulties.[7,8,9] In such fields, studies attempting to resolve implementation obstacles and related organizational problems have resulted in development of extremely effective techniques for analyzing systems and for introducing changes to these systems. These recently developed techniques appear to have application to a wide range of systems, including schools.[10,19]

Recent Developments in School Organization

Through the innovations that have become available to education in recent years, a single teacher in a central studio can give a lecture or demonstration visible and audible to hundreds of students in numerous classrooms in each of many schools; or the teacher in a given classroom may work with one student while others are proceeding at their own pace with programmed materials; or teaching teams can be formed so that while one teacher is lecturing to a large group of students, others can be preparing for future lessons.

Bolstered by such innovations, some school administrators have been experimenting with the structure of schools. One such variation in organization, the Trump Plan, with its emphasis on team teaching, on multisized groupings, and on use of clerks and teacher aides, has been introduced in a number of schools. One of the more promising directions being attempted is the experimentation which some schools are conducting to produce greater individualization of instruction.

Experimenting with the organization of a school presents many difficulties and can be expensive; hence, even in those districts where plans are given long and careful consideration knotty problems are faced. Description of three schools experimenting with their organizations will illustrate types of innovations and some of the problems that are developing.

School 1. An eastern high school which recently received national publicity is permitting students to progress at different rates through the curriculum. This has been accomplished by describing each course as a series of study units; dividing students according to ability and then further subdividing each class homogeneously, much as an elementary school teacher does; helping each subgroup to move at its own rate through the units of study in a course; and defining completion of the course as mastery of all the units and passing of a final examination. Thus, one student group may complete a course in six months while another requires fifteen months. A student who moves at a rate different from the rest of his subgroup is placed in a more appropriate group.

Student progress is reported in terms of (a) extent of progress and (b) quality of work. Thus, each parent is informed how far his child has progressed in each subject and also how thoroughly he has mastered the work covered.

School 2. The personnel of a junior high school in southern Los Angeles County are attempting to provide individualized instruction at the ninth-grade level in a manner quite different from that of School 1. Children all spend the same amount of calendar time on a course, but the daily time units may vary, thus providing opportunity for such elective activities as special science projects, typing, or remedial or speed reading. To effect this plan, the ninth grade is rescheduled each day in the following manner:

a. The school day has been divided into fourteen 20-minute time modules with five minutes between modules.
b. Every day each teacher submits a time request for four days later, indicating the number of modules needed for each class. (The day the teacher turns in the request will be referred to here as Day 1 and all subsequent days will be numbered from there.)
c. On Day 1, team leaders, using the teacher time requests, assemble a schedule unique to Day 4. This schedule is then reproduced.
d. During the first time module of Day 2, the students meet in groups of 20 with a teacher-counselor and each student prepares his personal schedule for Day 4 in quadruplicate, using pressure-sensitive paper. The student first registers for modules which his teachers have requested and then registers in the remaining modules for activities of his own choosing.
e. During Days 2 and 3, the teacher-counselor checks the schedule for every student to determine whether he is properly registered for Day 4.
f. On Day 4, the student carries his schedule with him so that each teacher may stamp it to indicate attendance.
g. The stamped schedules are collected centrally and checked to insure that each student has been present for each selected activity.

School 3. At the present time, a strikingly different plan is being prepared for a high school in eastern Los Angeles County. While this school will not be operational until 1964, education and building plans are both being advanced rapidly. The Continuous Progress Plan, described by Read and Crnkovic,[15] forms the basis for the educational plan being developed. This plan is designed to permit students to move at their own rate through the curriculum. To achieve this goal, instruction is oriented around individuals and small groups. The physical plant will have a large central room with individual study stations (carrels) where it is estimated each student will spend from 50 to 70 per cent of his time.

While instructional procedures will vary from course to course, a typical academic course will proceed somewhat as follows: Those students ready to begin the course at a particular time will go to a small lecture room where a teacher will introduce the course, outlining content, procedures, etc., answering questions and leading discussion. At the conclusion of this introduction, appropriate materials will be issued to each student and he will proceed to individual study in his carrel using programmed materials, library resources, etc. As a student completes each instructional unit in a course, his work will be evaluated and, if it is judged satisfactory, the student will have the next unit introduced to him. In addition to the individual and lecture-discussion work of the course, small groups of students will also complete appropriate group projects and will participate in seminars and other group activities.

Student registration will be an individual matter and may occur at any time during the year. Registration will take place most frequently at the time of completion of a course, but it may also occur at other times.

Each of the three schools just described is attempting to make better provision for individual differences. The solutions developed differ greatly in some respects but are parallel in others. Schools 1 and 3 provide for a difference in course calendar time, while School 2 requires that students spend the same calendar time in a course but permits the daily period to vary. Schools 1 and 2 retain traditional class groupings, but School 3 considers the individual to be the unit. Traditional instructional materials and media are used in Schools 1 and 2, but School 3 will employ programmed instructional materials, film strips and other innovations.

The three schools described share some interesting and perplexing problems. Two of particular interest to us are:

(1) Scheduling, typically a relatively simple matter occurring only twice a year, becomes much more complex. How can this problem be solved so that a minimum of teacher-administrator time is required?

(2) Student progress, once controlled by the student's schedule and by his teachers, now becomes an open problem. Especially in School 3, how can there be assurance that students are progressing at rates and in directions commensurate with their abilities and interests?

Role of Computers in Implementing Organizational Innovations

Although computer uses in education have been limited in the past, even a cursory consideration of the needs of the three schools just described indicates that a computer could be of real value to them. Of the various needs which a computer could fill in a flexibly organized school such as School 3, using the Continuous Progress Plan, we shall briefly discuss two: (a) command and control, and (b) guidance surveillance and detection.

Command and Control. "Command and control" is a phrase which is rarely used in education. In fact, the two separate nouns carry a negative affect value for most educators. However, if we view much of what takes place in the administrative aspect of education, we must certainly come to the conclusion that the command-and-control function is extremely important. For example, the whole processes of hiring, purchasing, and scheduling involve command-and-control functions. For our purposes here, we shall limit our discussion of command-and-control processes to scheduling.

In a school utilizing the Continuous Progress Plan, scheduling will be a much more complex problem than in a traditional school. Under this plan, scheduling will exist at a number of different levels.

(1) The first level of scheduling, that of assigning students to courses, will be somewhat parallel to scheduling in the traditional schools. However, under the Continuous Progress Plan, the procedure should be somewhat easier since the space limitation will not be as rigid as in a traditional school.

(2) The second level of scheduling under the Continuous Progress Plan will be assigning specific facilities to students on a day-by-day basis. For example, there will be a limited number of stations in the language laboratory. If students are to have some freedom in when they use the language laboratory, scheduling of the stations must take place. The same is true of various other items of equipment such as bunsen burners in the chemistry laboratory and duplicating equipment in the business-education laboratory. In a traditional school, this individual - student - matched - with - a - piece - of-equipment type of scheduling is handled by the teacher concerned. Under the Continuous Progress Plan, this kind of scheduling must be much more flexible and hence could probably be handled better on a day-to-day basis with some sort of central control such as could be provided efficiently with data-processing equipment. Students might turn in their specific requests each day; these could then be scheduled and each student's printed schedule handed back to him the next morning on his arrival at school.

(3) The third level of scheduling is substantially more complex than either of the other two and is the type of scheduling that occurs with relative infrequency in a traditional school. This level involves forming subgroups of students with similar problems and with other characteristics which the teacher may desire. For example, let us suppose that a teacher of algebra wants to form groups of

students to discuss quadratic equations. Suppose also that in a given period of time 20 students need the teacher's help on quadratics. This is more than the maximum number that the teacher wants to work with at one time and also, incidentally, more than the number that can be accommodated in one of the teaching studios. The teacher might earlier have given the instruction that if more than a particular number of students need a discussion of a specific topic on a given day, these students will be subdivided according to their immediate past history of success in the subject matter. When this general rule has been specifically stated, the computer, of course, can make the assignments as needed.

This type of subgroup formation would be impossible for the individual teacher to accomplish, and real coordination across subject matter areas would be impossible without the facilities of a high-speed computer.

Guidance Surveillance and Detection. The counseling function, as it operates in the traditional secondary school, does not appear to maximize the use and processing of student information. More efficient operation would be possible if the following factors were taken into consideration.[6]

The Continuous Progress Plan seems to provide a kind of environment in which the student's awareness of his responsibility is greatly heightened. The lock-step system, on the other hand, provides a learning environment in which the student can more readily project the responsibility for learning onto the teacher; by keeping students close together in progress, it reduces the wide differences that will emerge when motivation is allowed a free rein. The lock-step system can help to engender an attitude of "I'll do as little as I have to." However, early experimental work with the Continuous Progress Plan indicates that many students express a wish to return to the old system in which the teacher continually told them what to do. This heightened awareness of their responsibility is indicated by expressions of guilt about not applying themselves. There seems to be a consciousness of the fact that it is they themselves and not the teachers who are suffering the greatest loss when this responsibility is ignored.

In short, it seems evident that the learning conditions created by the Continuous Progress Plan will result in an increased need for counseling. As student awareness of the responsibility for learning increases in the presence of poor learning habits and attitudes, students will experience greater conflict and psychological tension. If the procedures do not provide for counseling where it is needed, the increase in disturbance may result in the development of maladaptive ways of solving the problem. Thus, if the system is not sensitive to the performance of students and appropriate action is not taken early, the Continuous Progress Plan may represent a negative learning experience for some students.

A guidance surveillance and detection system should serve a range of functions all the way from (1) checking on a student to insure that he carries out a task he has been instructed to do, to (2) accepting student requests for help, to (3) frequently sensing or evaluating whether a student is meeting normal expectations in his academic achievement. If this third function is to be performed, one of the more significant steps of the school should be an establishment of a reasonable expectancy level for the student. This expectancy level can then serve as a guide to the student, to his teachers, and to the guidance surveillance and detection system.

In essence, the guidance surveillance and detection system should facilitate the functions of *surveillance* and *detection* of trouble. It should maintain an accurate and up-to-date record of each student in relation to such critical questions as whether the student is achieving at his expectancy level. In a very general sense, the system should function in accordance with the flow diagram in Figure 1.* First, the information pertinent to a question is collected (1). Then the information is analyzed (2), and a decision is made about whether the student needs help or not (3). If help is needed, the appropriate personnel are alerted (4), and the information describing the problem is provided (5). If no help is needed in relation to the question, the system carries out the same steps

* Figure 1 was developed by Dr. John F. Cogswell of System Development Corporation.

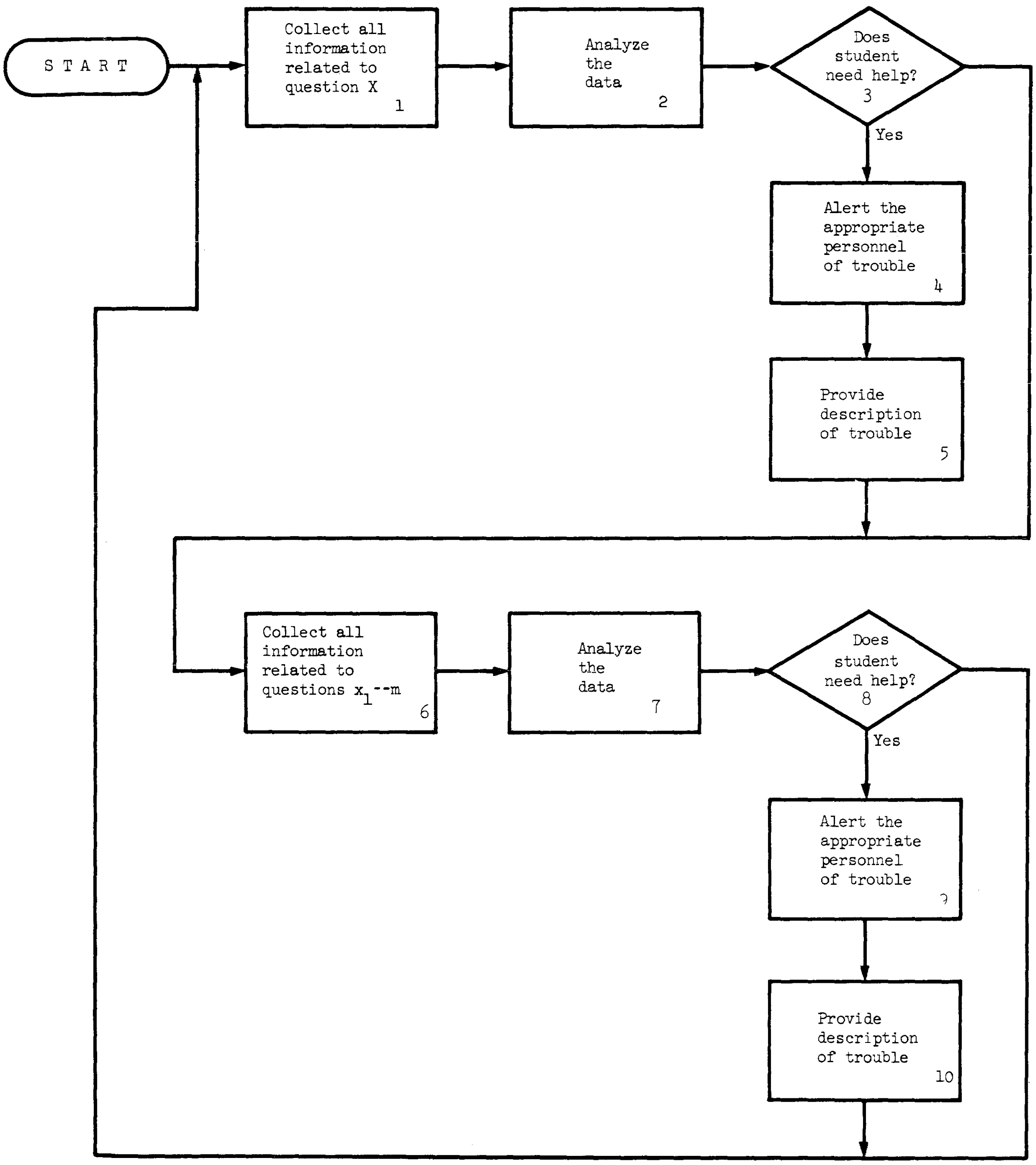

START
Collect all information related to question X 1
Analyze the data 2
Does student need help? 3
Yes
Alert the appropriate personnel of trouble 4
Provide description of trouble 5
Collect all information related to questions x_1--m 6
Analyze the data 7
Does student need help? 8
Yes
Alert the appropriate personnel of trouble 9
Provide description of trouble 10

Figure 1. General Flow of Guidance Surveillance and Detection Function.

in terms of all other questions that need to be asked (6-10). When all the questions have been answered, the cycle is repeated to keep the function updated in relation to new information that may reflect dynamic changes.

Role of Computers in Designing New Educational Systems

At the present time the typical procedure for designing a new educational system is for an educator who is dissatisfied with the outcomes of the system with which he is working to try to think through a better system. He then checks this new system with his faculty and various other educators, secures community support, and installs the new system. For a system simpler than that of a school, these procedures might work effectively. In fact, in some instances educators are relatively successful in redesigning new systems for their schools. Typically, however, many unanticipated problems arise. Because these unanticipated problems can result in costly revisions and poor results, some groups are now experimenting with computer simulation of schools to enable an inexpensive testing or study of the new system before it is actually implemented.[2, 4, 16]

If a school is simulated, the model should probably involve computer representation of students, teachers and other school personnel, curriculum, space, and equipment. A simulation model of a school should be sufficiently flexible to accomplish many purposes.[11] For example, representation in the model should be possible for each of the three schools described earlier in this paper. In addition to representing a number of different school organization patterns, a generalized model of a secondary school should permit the investigator to employ and study a wide range of theories, theories about students and their behavior, about instruction, and about school organization and management. If the goals are to be attained, the model must be able to construct and manipulate student groups varying greatly in size; it must be able to depict students being registered individually as the need arises, all being reregistered each day, or all being reregistered each semester; it must have the capacity to cope with a school day consisting of short modules, long modules, or periods of varying lengths; it must be able to represent students always studying under the direct guidance of a teacher and students studying alone with the capability of requesting help as needed; it must able to show both degree of achievement and speed of progress for each student; it must have very few rules governing its operation as a school and it must be able to accommodate many different rules as desired by the investigator; it must be able to represent students with no predetermined characteristics which influence outcomes and those having a variety of traits affecting their school progress; and it must be able to show the changes in the student as he progresses through the curriculum.

If an educator, in attempting to design a new system, had access to a computer model such as is mentioned here, the chances of his producing an effective and efficient shool system would be greatly improved.

IMPLICATIONS OF THE USE OF COMPUTERS IN EDUCATION

In this paper we have attempted to outline a number of developments that are occurring in education, developments that are becoming available to those schools interested in considering change. We have also briefly examined some of the experimental programs that have implications for the future in education. Both in the new technology that is now available and in that which appears possible in the foreseeable future, problems exist that are of concern to educators and noneducators alike. Several of these problems might be mentioned briefly.

(1) To what extent should teaching be directed toward producing uniformity? In programmed learning, the assumption is inherent that students should all arrive at the same goal at the end of the lesson. The computer provides a particularly effective means for insuring that the needed material is presented to each student in a sequence that insures that all will arrive at this same goal. Depending upon subject matter and on one's point of view, this may or may not be good. If, for example, we consider certain elementary aspects of mathematics, we might all agree that we want students to arrive at the same answer, and a program of teaching that enables each student to

secure this answer would be judged good. On the other hand, we may not want all students in a social studies class to arrive at the same conclusion. Diversity of opinion may be desired. Is there some danger, then, in our providing detailed assurance through programmed, computer-controlled materials that all students will eventually reach the same goal?

(2) At any given time, how much information should be available to one person about another? This question, of course, refers to the guidance surveillance and detection system. In our earlier description of such a system, the positive results were given primary consideration. Along with the positive outcomes, however, perhaps we should consider the "big brother is watching" attitude which might develop both among school personnel and among students. In an idealized guidance surveillance and detection system, the person who has a need to know may be able to secure, with very little effort, extremely detailed information about a student. The student concerned may not want this person to have that much detailed information. At the present time, relatively limited amounts of information are available and this information is difficult to acquire. With a highly organized guidance surveillance and detection system such information might be acquired much more readily. The degree to which this is desirable is open to question.

(3) In a school making extensive use of technological innovations, with computer assistance in its operation, will there be a change in the role of the teacher? This is one of the questions asked most frequently in discussions of technology in education. Usually this question carries two implications: first, that a reduction in pupil-teacher interaction will follow introduction of "machines," and second, that there will be a decreased need for teachers. With the information now available a definitive answer to this question is not possible. Certainly the teacher's role will be modified, but the extent and nature of the change are uncertain. In general, repetitive activities ranging from roll-taking and attendance-reporting to presenting routine lectures and demonstrations will be most amenable and attractive to technological replacement. Activities which require constant thought and adaptability on the part of the teacher will be much less susceptible to new technological substitutions.

(4) If teacher roles undergo metamorphosis because of computer-aided innovations, what training and retraining problems will result? As technological innovations are introduced, teacher-training programs will need to increase emphasis on how to make use of the new developments and on how to work with individuals and small groups. There will be a corresponding decrease in emphasis on control functions of the teacher.

(5) What will be the implications of new educational technology for school architecture? For a number of years foresighted school building planners have sought to construct schools with flexibility such that they could accommodate a variety of innovations. This flexibility has stressed interior walls that could be moved readily. In addition to even more readily adaptable space, future schools will certainly need greater provision for electronic equipment for administration, instruction, counseling, and for the various library services.

(6) Will the educational community accept computer-aided innovations? The answer here is "yes," but slowly and with reservations. In fact, the reservations will be so extensive and the progress so slow that some skeptics will feel that acceptance is totally lacking. However, as has been true with other developments, educators will gradually accept computer-aided innovations. Continued pressures of population increases, budget problems, and demands for more effective and efficient education will provide the needed impetus.

Computers have contributed greatly in fields other than education and seem to hold similar promise as an aid both in developing and implementing educational procedures and technology. However, problems such as those mentioned raise serious questions on specific computer applications in education.

SUMMARY

In our paper we have outlined some recent developments in instruction and we have described two programs in which computers are being used in work with instructional innovations. We also have mentioned some recent developments in school organization and have described possible roles for computers in implementing organizational innovations. Finally, we have raised some representative questions which we think merit serious consideration concerning implications of certain educational innovations which are either facilitated through the use of computers or which are possible only through using computers.

BIBLIOGRAPHY

1. BITZER, D. L., BRAUNFELD, P. G., and LICHTENBERGER, W. W. Plato II: A Multiple-Student, Computer-Controlled, Automatic Teaching Device. In J. Coulson (Ed.), *Programmed Learning and Computer-Based Instruction.* New York: John Wiley and Sons, 1962. 205-216.
2. BOYER, R. A. Simulation and Models. Paper prepared for Educational Data Processing Workshop. February, 1963. 3 pp.
3. CHAPMAN, R. L., and CARPENTER, JANETH T. Computer Techniques in Instruction. In J. Coulson (Ed.), *Programmed Learning and Computer-Based Instruction.* New York: John Wiley and Sons, 1962. 240-253.
4. COGSWELL, J. F., EGBERT, R. L., MARSH, D. G., and YETT, F. A. Construction of a School Simulation Vehicle. System Development Corporation document, TM-1409. August, 1963. 47 pp.
5. COULSON, J. E. A Computer-Based Laboratory for Research and Development in Education. In J. Coulson (Ed.), *Programmed Learning and Computer-Based Instruction.* New York: John Wiley and Sons, 1962. 191-204.
6. EGBERT, R. L., and COGSWELL, J. F. System Design in the Bassett High School. System Development Corporation document, TM-1147. April 1, 1963. 22 pp.
7. HEYNE, J. B. On the Empirical Design of Management Control Systems. System Development Corporation document, TM-585. February 15, 1961. 48 pp.
8. HOPKINS, R. C. A Systematic Approach to System Development. System Development Corporation document, FN-4176. August 11, 1960. 28 pp.
9. JAFFE, J. Information-Processing System Design: General Principles, Part I. System Development Corporation document, TM-743/001/00. August 9, 1962.
10. KERSHAW, J. A., and MCKEAN, R. N. System Analysis and Education. RAND Corportation document, RM-2473-FF. October 30, 1959. 64 pp.
11. LACKNER, M. R. Toward a General Simulation Capability. *Proceedings, 1962 Spring Joint Computer Conference.* 1-4.
12. LICKLIDER, J. C. R. Preliminary Experiments in Computer-Aided Teaching. In J. Coulson (Ed.), *Programmed Learning and Computer-Based Instruction.* New York: John Wiley and Sons, 1962. 217-239.
13. PRESSEY, S. L. A Simple Apparatus Which Gives Tests and Scores—and Teaches. *School and Society.* Vol. 25, March 20, 1926. 549-576.
14. PRESSEY, S. L. A Machine for the Automatic Teaching of Drill Material. *School and Society.* Vol. 25, May 7, 1927. 549-552.
15. READ, E. A., and CRNKOVIC, J. K. *The Continuous Progress Plan.* Provo, Utah: Brigham Young University Press, 1963.
16. ROME, B. K., and ROME, S. C. Automated Learning Process (ALP). System Development Corporation document, SP-785. April 13, 1962. 52 pp.
17. SKINNER, B. F. The Science of Learning and the Art of Teaching. *Harvard Educational Review.* Vol. 24, Spring 1954. 86-97.
18. UTTAL, W. R. On Conversational Interaction. In J. Coulson (Ed.), *Programmed Learning and Computer-Based Instruction.* New York: John Wiley and Sons, 1962. 171-190.
19. WILLIS, B. C. *Total Information Service for the Board of Education, City of Chicago.* Chicago: Board of Education, Chicago, 1962.

COMPUTER-ORIENTED PEACE RESEARCH*

Dr. Louis Fein, Consultant
Palo Alto, California

INTRODUCTION

Imagine a management consulting and research type of organization called, say, the Universal Study Center for the Salvage and Reorganization of Institutions in Imminent Danger of Destruction Applying Computers Wherever Feasible (USCSRIIDDACWF), located on a planet in our solar system. Its services are available for hire to planetary social institutions. One of its newly formed divisions called, say, the *P*eace on *E*arth *R*esearch *C*enter (PERC), has on its staff historians, social scientists, physical scientists, artists, and members of the computer profession. Suppose, furthermore, that Earth, a planetary social institution in imminent danger of self destruction, has called upon USCSRIIDDACWF to produce considered and documented alternative advisory opinions on how Earth might prevent or indefinitely postpone its demise, and on how to establish conditions favorable to future Earth peace, health and prosperity. The contract is assigned to PERC—the division of USCSRIIDDACWF specializing in Earth's problems.

In the light of the combined experience of PERC's staff with the process of analyzing problems of institutions of many kinds and sometimes applying computers, to a wide variety of problems, how would we in PERC proceed on this assignment? What tasks and schedules would we allocate to individual PERC researchers? What kind of technical, political, and psychological problems should we expect them to encounter? What should we caution them about? What validated procedures and tested hypotheses may they use with a high confidence level? What kind of problems should we expect to be amenable to solution with present computers and programs? What kinds not amenable? What kinds of problems can people alone solve? What kind of problems are so far imponderable?

Social and physical scientists have had a good deal of experience and some success in trying to solve moderately complex problems with the aid of computers. Regardless of how much more complex are the problems of attaining and maintaining world peace, the distinctive analytic approach to practical problems and computing instruments of computer-oriented analysts may be fruitful in finding solutions to some of the problems of peace.

Relevant Experience of Computer-Oriented Analysts to Peace Research†

The combined experience of computer-oriented problem analysts has been gained primarily within the last decade in many fields, including banking, insurance, combinatorial mathematics, chemical engineering, and war gaming. We are quite familiar with the role

*This is a revision of the author's paper "A Proposal for a Scientific Computer-Oriented Project on World Peace Research" that appeared in BACKGROUND; Journal of the International Studies Association, Vol. 7, No. 2, August 1963. Excerpts from that paper are published with permission of the copyright owner: Institute for Research on International Behavior, 1600 Halloway Avenue, San Francisco 27, California.

† Expressions in parentheses in this section highlight the relevance of this experience to peace research.

of computers as problem solvers, as calculators and as simulators, emulators, and imitators. But computers play their most significant role as a socratic goad to analysis and problem formulation. The mere attempt to decide how computers may be helpful in a situation (e.g., identifying causes and preventives of war), stimulates analysis of that situation often resulting in insights into solutions of problems not necessarily requiring subsequent use of computers.

Ascertaining the specific tasks for which computers may be helpful has turned out to be one of the last steps of long and tortuous sequences of events in company salvage and reorganization programs. Clients (Earth) usually need established facts and solutions to problems obtained with or without the aid of computers. They appreciate computers as means, not as ends.

Management consultants and system analysts have found a certain distribution of opinion among client managers and staff (national leaders and their counsellors) about the desirability of calling in help at all. While admitting that their institution (Earth) is in trouble, some (statesmen and politicians) feel that they themselves could salvage the situation. To understate the case, such persons usually do not cooperate with outside helpers. Others accept outside suggestions with reservations only incidentally related to the good of the institution (Earth). If the outsiders' (PERC's) recommended reorganization, policies, and practices serve to maintain or enhance a man's (nation's) empire, position, prestige, or ego, or if the recommendations do not conflict with strongly held beliefs (ethos) then the suggestion is acceptable; otherwise not. The attitude of such individuals (nations) will be somewhat cautious.

In gathering background information, we should expect to find objectors among those (leaders, statesmen, politicians) who for the very first time will be asked to state clearly what they do, and why they do it in the way they do. Many conscientious and hard working persons have learned to cope satisfactorily with their tasks more or less empirically, making such fundamental inquiries unnecessary. As a critical analysis proceeds, and as rationales (or lack of them) are revealed, people feel threatened by the possibility of a further uncovering of their inability to give orderly and specific explanations of what they do, how they do it, and why. Since practitioners (leaders, politicians, statesmen) have learned to do their jobs with a considerable investment of study, practice, money, and self-involvement, one should expect them to resent and reject what seem like attempts by outsiders to undermine their situation. Furthermore, a man who fears computers will not respond in the same way as one who expects computers to make his work "right" and "efficient." These feelings will determine whether computers are acceptable at all, or whether they will be used exclusively, and will affect the credibility of results obtained with their aid.

It would be wrong, of course, to hold that if only there weren't vested interests and fearful people, all problems arising in the salvage and reorganization of institutions (Earth) would be soluble, and that for most of these, computers and programs would be helpful. Even if we were unencumbered by conscious or unconscious sabotage, there are many tasks that can neither be satisfactorily performed by computers alone, by people alone, nor by both in combination. For example, while trying to identify the current problems of an institution, one often finds that although clients can identify problem *areas,* they are hard put to state clearly *specific* problems within these areas. Problems (of the institution Earth) that can't be formulated cannot be attacked—let alone solved. But, even having well-formulated problems is no guarantee that they are soluble or, if soluble in principle, that a solution is practical. Many institutions are obsolete, i.e., their organizations and procedures are no longer equal to satisfying present needs for growth and prosperity. Solving even well-formulated problems in the context of an outmoded institution (social structure) is useless to the client (Earth); it can only show him how to solve yesterday's problems.

Furthermore, well-formulated problems even of up-to-date organizations may be unsolvable, if no efficient method of solution is known or if the input data for what would be an efficient solution are irrelevant, inaccurate, or incomplete. For example, a highly placed administrator in one of our large companies is said to have complained that he cannot perform his well-formulated task of planning for the next five years because most of his department managers lie to him about their present and anticipated situation. In international affairs, the head of a government would be hard-pressed to make appropriate decisions if the reports on local situations from his ambassadors were falsified.

Conditions for Solution of Problems

Some conditions under which problems can be solved by human beings are:

(1) The phenomena, behavior, or situations to be controlled, predicted, and understood must be "lawful" rather than accidental events; the "laws" may be empirical, deterministic, or probabilistic.

(2) Human beings must somehow "know" the laws, having ascertained them by intuition, by experience, by test, or by accepting the untested hypotheses or revelations of oracles, prophets, historians, logicians, or philosophers.

(3) The particular problem to be solved must be well-formulated.

(4) An efficient method of solution must be known in a form that can be implemented by human beings.

(5) Relevant input data, sufficiently accurate and complete, must be available or inferred as needed.

If we supplement the human problem solvers with computers (or substitute for them altogether) then similar conditions must obtain:

(1) The phenomena, behavior, or situations to be understood, predicted, and controlled must be computer modellable (or simulable).

(2) Analysts and programmers must somehow know the model, having ascertained it by intuition, by experience, or by test.

(3) The particular problem to be solved must be computer formulable.

(4) An efficient program for solution must be known and the computer must be capable of carrying it out.

(5) Relevant input data, sufficiently accurate and complete, must be available or inferred as needed.

A Hypothetical Peace-Research Program—PERC

A typical salvage and reorganization program undertaken by management consultants who already know the nature (the "lawfulness") of their client's business consists in general of six overlapping phases:

(1) Identifying and accumulating relevant, substantiated, and accurate background knowledge unique to the institution under study, to provide the basis for the subsequent diagnosing of the ills and the prescribing of treatments,

(2) Diagnosing the reasons for imminent collapse of the institution,

(3) Identifying conditions under which longer-range success may be expected,

(4) Inventing alternative strategies and tactics for the salvage operation, and estimating their probable success,

(5) Inventing alternative strategies and tactics for the longer-term reorganization operation, and estimating their probable success,

(6) Inventing and evaluating feasible routes of getting from the present condition to a desired set of future conditions.

If the nature of the client's business is not known at the outset, then gaining this knowledge is the researcher's first concern.

Let us (PERC) plan a six-phase program whose results would be well-considered, documented and tested theses on the prevention of nuclear war and the establishment of the basis for future peace on Earth. In the first phase, background data is gathered. This might include ascertaining the past, present, and near future values, mores, aspirations, needs, motivations, beliefs, interests, attitudes, abilities, and physical and spiritual resources of the nations of our client, Earth, of the groupings within the nations, and of the individuals within the groupings. We may also have to ascertain past and present policies, strategies, tactics, organizational structures and procedures, and the rationales of all five of these; first for nations, then for groupings of individuals, and finally for individuals. We may want to characterize these items still further by economic, political, sociological, technological, psychological, cultural, military, moral, ethical, ideological, legal, semantic and any other aspect required by the diagnosticians trying to identify and trace the causes of the present dire straits of Earth and required by the prescribers and administrators of the palliatives and of the cure.

The statement that diagnosticians and the prescribers of palliatives and cures determine the sort of data to be gathered, implies that Earth's illnesses are diagnosable and treatable, which in turn implies that world events and international and individual behavior are "lawful"; that we somehow know the "laws" or we can find them; that we can thus make valid inferences and derive useful conclusions with the aid of these "laws." Furthermore, if computers are to be used as aids in diagnosing and prescribing, then the implication is that world events and international and individual behavior can be modeled and that computers can be programmed efficiently to investigate the properties of the model and thus to solve specific problems of war and peace.

Unfortunately, it is by no means clear which, if any, of the phenomena and behavior patterns observed and studied in "peace research," do follow "laws" or which can be modeled. Nor do we know how to ascertain beforehand that they are either "lawful" or modellable. Those who would diagnose and prescribe treatment for the ills of the world must have an abiding faith in the "lawfulness" of world events and a conviction that, by intuition, experience, experiment, or revelation, they have or can attain both a sufficient knowledge of those laws and of all necessary data to allow them to make accurate diagnoses and to prescribe adequate cures.

Indeed, there have been a large number and wide variety of theses purporting to specify the causes of war as well as the conditions for world peace and prosperity. National and world movements, and cults and cultures have sprung from some of these theses, advanced by Plato, Marx, Freud, Darwin, Buddha, Christ, Malthus, Thoreau, Moses, Ghandi, Nietzsche, Jefferson, and others. Inasmuch as it devolves on PERC to invent and to evaluate short-term and long-term strategies and tactics for the prevention of war and the establishment of peace on Earth, PERC will have to obtain by experiment, experience, or otherwise, a causal (deterministic, probabilistic, or empirical) model of the world, which would be used to indicate what relevant data to gather. This model, together with the processed data, would later be used to deduce and evaluate alternative strategies and tactics. Without a suitable model, PERC cannot get started just as a management consultant, without a knowledge of the nature of the business of his client, cannot plan a program. The adopted model will determine what data are relevant.

Let us examine just two of these theses that might influence PERC modellers and "law" seekers or from which they may even borrow. Some of these theses include, explicitly or implicitly, a formula for saving the world now and assurance of future peace and prosperity; they provide both a diagnosis and treatment. How would a six-phase research program be affected by the adoption of such theses?

If Christ's teaching on universal love were accepted as a necessary and sufficient condition for world peace, then data-gatherers may be directed to ascertain how the characteristics and behavior patterns of nations, groups within nations, and individuals correlate with the principles of universal love. These data would be analyzed with a view toward recommending strategy and tactics for getting from the present world condition to the world condition accommodating universal love. Computers would certainly be useful in the vast data-processing chore of sorting, calculating, correlating, etc. They might also be useful in prescribing an optimum Earth reorganization, if a suitable model were devised and if programs for attaining and maintaining the conditions of universal love could be tried on this computer model.

If PERC diagnosticians and prescribers of treatment were to accept the Marxian hypothesis that wherever one finds capitalism (primarily corporate and individual capitalism as in the U.S., or state capitalism as in the USSR) war is inevitable, then data-gatherers would be instructed to ascertain the character and behavior patterns of nations, groups within nations, and individuals that affect and reflect its capitalistic nature. (These data would probably be different from those requested with the adoption of Christ's hypothesis.) These data would be used in determining the optimum route for going from a capitalist society to a so-called democratic socialist society where the Augustinian slogan "from each according to his ability and to each according to his need" would be the guiding policy for the prevention of war and the establishment of peace and prosperity. Here, too, computers might be useful in data-processing and in trying alternatives on models.

The reader can imagine for himself how the magnitude and the course of each of the six phases of the PERC investigation would be influenced by the adoption of other world views, and also how computers may be helpful. But on further contemplation, it is logical for you to insist that any world view that is adopted must be correct; if not, then diagnoses and treatments based on it will be wrong and might result in our destruction. Perhaps PERC is not at the moment in any position to adopt with confidence a world view or model, despite the plethora of views available to it. Perhaps PERC has a seven-phase program rather than a six-phase program with the first phase being to ascertain the "laws" which explain national, group, and individual behavior, or to find models which simulate this behavior. I myself think so. In this new first phase, existing views, hypotheses, and formulae should be tested, evaluated, and new ones should be invented, tested and appraised. Of course, we shouldn't expect to find "the" explanation. But the remaining six phases could then be carried out in the light of the clearly exhibited character of, and evidence for, the adopted world view.

To pick assumptions to pieces till the stuff they are made of is exposed to plain view is a commonplace procedure used by mathematicians in order to understand the foundations on which theorems are built. Is it not just plain good sense (and I daresay more important to society than for mathematical theorems) to pick our assumptions about human behavior and "laws" to pieces till the stuff they are made of is exposed to plain view in order for us to understand the basis on which our actions are guided? Indeed, it may be ridiculous to expect us to be able to find a world view or model in which we could have sufficient confidence to use it as a safe guide for prescribing Earth palliatives and cures. Even if we did find one, the data-gathering phase would be a mammoth task, especially if it turns out that we must ascertain past, present, and near-future values, mores, aspirations, needs, motivations, beliefs, interests, attitudes, abilities, and resources of nations, groupings within nations, and individuals; also ascertain strategies, tactics, structures of organizations, procedures, and their rationales; and also classify these data possibly by tens of aspects including economic, political, sociological, psychological, cultural, military, technological, moral, ethical, ideological, legal, and semantic. The next five phases of diagnosing the reasons for imminent destruction of Earth and of inventing and evaluating alternative palliatives and cures may seem impossible tasks. All these fears may turn out to be well-founded. But we won't know until we have tried to carry out such an applied peace-research program. It is premature to give up now on our intellectual resources to figure out ways to our immediate survival and ultimate salvation. Computer-oriented analytic approaches may be an important resource. Computers may be key instruments.

Organization for World Peace Research

I therefore propose that a PERC type of organization be established with international sponsorship, support, and participation. Its charter will be to plan and carry out the kind of integrated and coordinated seven-phase peace research program already outlined. PERC administrators and researchers, representing most of the disciplines, will consider that the peace problem is technical—not political. They will proceed as scientists do; they will insist on valid procedures and critical testing of hypotheses concerning "lawfulness," problem solution methods or algorithms, diagnoses, and treatments. If they find that on occasion they have no valid procedures or ways to test hypotheses, then they will apply their efforts to trying to find valid procedures and hypotheses-testing methods. They will not adopt an untested procedure or hypothesis because they do not yet have a way to test its validity. To do all this, their minds must be unintimidated, unconstrained, and uncoerced. The criterion for professional participation in PERC will be technical competence and disciplined objectivity.

International backing for PERC is needed for two reasons. The seven-phase program is extensive, time consuming, and expensive; such a job may require financial, material, and intellectual resources from every part of the world. PERC's suggested diagnoses and treatments must be as "scientifically" valid as possible; this can be assured only if practitioners are able to work in an unconstrained environment. Imagine the reaction of both the governments and peoples of the USSR, the U.S., or the People's Republic of China to such possible PERC recommendations as: there ought to be a world government and a communist economy; there ought to be a world government and a capitalist economy; both the USSR and the U.S. should give aid to China; there should be a world common market; every nation should be given a nuclear deterrent. International backing is most likely to provide appropriate conditions of freedom of thought for PERC practitioners.

One integrated and coordinated peace research project directed by a central organization is recommended, rather than several specialized programs, both because this is of necessity a crash project and also because the problems to be dealt with in each of the seven phases are too closely interwoven and interdependent to allow for effective operation otherwise. In the U.S. alone, there are hundreds of scattered and uncoordinated peace research studies in progress, attempting attitude surveys, peace gaming, conflict resolution, economic modelling, etc., at universities, volunteer research groups, government agencies, industrial establishments, think factories, and research institutes. None of these research groups or individuals who claim to be doing peace-research seem to be engaged in a program which has all the required elements of a technically competent, orderly, and goal-oriented piece of applied research. These elements include at least:

- clearly stated and well-formulated specific goals and objectives of a total peace research program,
- clearly stated and well-formulated specific goals and objectives of the particular applied peace research in progress,
- tested hypotheses, validated procedures competently carried out, and substantiated relevant data.

Finally, the element which is universally missing in proposed and on-going peace research is the knowledge or even concern of how the results of a particular peace research project might contribute to the attainment of the goals and objectives of a total peace research program. In short, if every single one of these isolated projects succeeded at this very instant, we would not be one step ahead in our pursuit of peace; i.e., in learning under what conditions the people of the world will live at peace with each other, or in determining feasible routes for getting from our present condition to these conditions. Indeed, we might be behind because of the mistaken notion held by most "peace-researchers" that their results are relevant.

Results of peace research have been disappointing to sponsors perhaps because they (the sponsors) failed to comprehend the essential ingredients and implications of genuine and effective applied (as opposed to pure) research. This is not a task for one professor, or for one "five-man" research team, working on a problem of its own choosing in the best traditions of pure research. A lone genius or principal investigator on a government or foundation or dedicated philanthropist's grant, will not emerge some Tuesday and present the world with realizable and the least-risky means of getting from our present condition to one congenial with peace. The sucessful completion of the Manhattan (Atomic Bomb) Project was due to a vast research team of scientists working under the administrative direction of General Groves. Despite his $E = mc^2$, Einstein alone couldn't possibly have accomplished it. Project PERC would be fundamentally different in spirit, size, direction, organization, and competence from current fragmented and miniscule research. This is not to imply that scholars outside Project PERC should stop working on peace research and development problems, but their contribution to an integrated and coordinated effort could be fortuitous at best.

Opposition to PERC

Obtaining moral and financial support for PERC may be extremely difficult. The opposition is all-pervasive, including lay "peace-seekers" and "war-mongers"; journalists; professors and other individual researchers working in isolation on problems of peace; their sponsors, their institutions, and their governments. The grounds for opposition are ideological, tactical, organizational, jurisdictional, pragmatic, moral, economic, emotional, intellectual, and scientific.

In the West, peace is not respectable; by some it is considered a dirty word. Many feel that PERC would be infiltrated or used by insincere, non-peace-loving organizations and nations for nefarious ends. A great number think that the determination of the causes and treatment of a disease like polio, for example, is clearly the job of a well-directed, integrated, and coordinated team of qualified, professional, medical researchers and clinicians to whom the lay public must give only moral and financial support rather than amateurish medical advice. Astonishingly, at the same time, they believe that the determination of the causes and treatment of the social diseases of world nuclear war is just as clearly *not* a job for a well-directed, integrated and coordinated team of qualified, professional researchers and practioners in international relations to whom must be given only moral support and financial support rather than amateurish advice on how to prevent world nuclear war. They think that current problems of war and peace are primarily of a moral and empirical nature and can be coped with by national leaders if only they would use "common decency" and "common sense" as Leo Szilard said, or as Bertrand Russell wrote, "if only they (the leaders) could be stopped from acting insanely." These are but two representatives of the belief that the orderly scientific method in the pursuit of world peace is irrelevant; long-haired theoreticians whether in a library, a laboratory, or a vast research organization, such as PERC, are not needed to solve these problems. Others, especially among participants and supporters of peace-action groups, feel that research takes too long and we just do not have the time. They have been saying this for the last eighteen years.

Furthermore, there is the vast majority of us who know the answers and who obviously couldn't see any advantage whatever in supporting research which would seek solutions that we already know. Among us we believe that:

(1) If we arm, then peace would result because arms are deterrents,

(2) If we disarm, then peace would result, because without arms nations would have to find alternatives to nuclear violence for resolution of conflict,

(3) If we strengthen the UN, then peace would result, because a strong UN would mediate any aggravating situations among nations,

(4) If we dissolve the UN, then peace would result, because it is organized to favor one group of nations,

(5) If we abolish capitalism, then peace would result, because capitalism leads to war,

(6) If we abolish communism, then peace would result, because Marx, Lenin, and Stalin said that the overthrow of capitalist nations by violent means is inevitable,

(7) If we had a world government under world law, then peace would result, because . . . .

That these views, however sincerely and dearly held, and on whatever empirical evidence and moral or religious conviction they are based, are nevertheless untested hypotheses in the scientific sense, is largely unrealized by their adherents. PERC would insist on testing all of these hypotheses in its Peace Hypothesis Testing Laboratories. Adherents may fear that such tests might serve to invalidate their opinions.

The remainder of those that would oppose PERC believe that even if PERC came up with a set of well-considered recommendations, arrived at in an orderly, scientific manner—sometimes using computers—the world would reject them anyway, so why bother?

There will be opposition to PERC even from among those who believe in the necessity and relevance of what they consider to be peace research.

Slowly but surely, peace research (usually coupled with arms control) is taking on the character of some military-sponsored and some other government-sponsored research. The pursuit of grants by universities mainly for support of graduate students and the pursuit of contracts by industry, institutes, and think-factories mainly for fees, or notoriety, or just to remain in business, is concomitant with mediocrity and indifference to accomplishment. A few grantees and contractors will oppose PERC as a threat to their way of life.

Those who believe that the appropriate formulation of the problems of peace and war and the pursuit of their solutions are matters for scattered and uncoordinated grantees and contractors with lots of national and international conferences to increase communication and integration, will oppose PERC on organizational grounds.

There are many who deplore the fact that the U.S. military budget for 1962 was $52 billion; for the U.S. Arms Control and Disarmament Agency, it was $6 million. They believe that technically and organizationally we are doing peace research in an adequate manner. They merely think that what is being done should be given much more financial support. They will be PERC opponents.

Actually, most, if not all, of the work proposed and proceeding under the rubric of peace research is, in fact, peace development. In an applied scientific research program, an objective (not requiring validation) is *first* agreed to and *then* hypotheses (requiring validation) on means for attaining this objective are proposed and tested. In the succeeding *development* program, validated theoretical means that survived the tests are made realizable within the constraints imposed by the environment, economics, etc. The process of searching for the most practical of these means is often described as research. In many peace research programs a favorite hypothesis of the grantee or principal investigator is first adopted without subjecting it to a test; *then* they proceed to *develop* this untested opinion. (This has also been called axe-grinding research.) A commonplace example of this appears in those programs, and in the writings of supporters of the scientific method in peace research, where the adoption of peace as an agreed goal (not requiring validation) is confused with the adoption of the untested hypothesis: "If we disarm, then peace would result" as the agreed goal. Not appreciating this crucial distinction, they proceed to investigate scientifically and to do research in realizable ways to disarm. This is disarmament research, i.e., the agreed *goal* is disarmament, whether or not it may turn out that in the process of its attainment war would result! Disarmament research is not the equivalent of peace research until we establish the validity of the hypothesis: "If we disarm, then peace would result." Armament researchers, world-government researchers, and others fall into the same trap. When one makes his hypothesis his objective, then he has disavowed a key step in the scientific method. Such persons will oppose PERC ideologically.

It has already been mentioned that supporters of "peace research" have been disappointed. If no difference is seen between PERC and what they supported as "peace research," they will (at best) be indifferent to PERC.

There are those who will point out, and rightly so, that specialists in international relations, apparently the very ones on whom PERC would have to depend most heavily, admit that their data is anecdotal and their theory untested; that their models and procedures are inadequate. In short, our intellectual cupboard for pursuing peace research is almost bare. Perhaps most important, we have no ways, and we may never be able to find ways to test hypotheses, such as "If we disarm, then peace would result." Persons appreciating this situation will urge caution and deliberate slowness in activating PERC.

But it would be unwise to allow these difficulties to deter us. We would be unfair to ourselves. It would be foolish to disregard the advantages of such an insurance policy. PERC should fulfill its contract. It should try to develop dependable hypothesis-testing methods, procedures, and models, which would give Earth the opportunity to consider well-considered, well-documented, and evaluated hypotheses on alternative solutions to the problems of war and peace. These may have a higher probability of success than the untested hypotheses Earth now considers and acts on. Even if PERC failed to develop the required techniques and was thus unable to say which combination of conditions and actions were most likely to lead to peace, it would have been worth the try (the premium on the insurance policy).

Possible Ways of Planning and Starting PERC

The United Nations probably has the resources and the motivation to undertake such a program. It may take the form of a U.N. World Peace Organization perhaps patterned after, and in the spirit of, the U.N. World Health Organization.

Another possibility is that thousands of people all over the world might, as individuals, support a PERC organization financially and some might participate in its technical work. Researchers often receive letters from physicists, engineers, behavioral scientists, etc., asking how they could participate professionally in peace research. For every letter writer, there must be many who feel this way but don't write. If so, perhaps, grass roots financial support might be obtained from such individuals who would also feel a sense of professional participation if PERC sent them reports of work in their specialty so that they could advise, criticize, and comment at their leisure.

Laymen the world over, especially women, may be willing to support an orderly, unintimidated investigation of the conditions under which the people of the world will live at peace with each other, even though the outcome may conflict with their most precious convictions that the solution lies in disarming or arming, in or out of the United Nations, in capitalism or communism, in world government or nation states, in religion or atheism. The March of Dimes may be a model of the organization for financing PERC.

Other possible sources of moral and financial support include scientific and literary organizations and publications; international labor organizations; churches; individuals and foundations; and world trade organizations.

The estimates given in this essay on PERC organization, financing, and personnel are necessarily tentative. PERC, which might cost a billion dollars in the course of fifteen or twenty years, should be competently and carefully planned in detail, well in advance of its starting, as should any billion dollar enterprise. During two years, ten experienced senior men of high caliber (indeed ten of the world's best planners), playing the role of the corporate planning staff of a billion dollar organization, should plan PERC finances, personnel, facilities, equipment, sites, research and priorities, organization, administration, policies, etc. This planning and starting phase would cost about one million dollars. To identify and recruit these ten key men would take about one year and might cost about $100,000.

Thus, if in 1963, the planning staff were recruited, PERC could be planned and started during 1964 and 1965. By 1966, PERC would have gained good momentum.

REVIEWERS, PANELISTS, AND SESSION CHAIRMEN

AFIPS and the 1963 Fall Joint Computer Conference Committee would like to express their sincere appreciation to those listed below for their contribution toward the formulation and execution of the technical program.

REVIEWERS

L. AMDAHL
G. W. ARMERDING
R. BARTON
W. BAUER
G. A. BEKEY
R. BENNETT
C. BLOCK
J. BLOOM
D. G. BOBROW
H. BORKO
J. R. BROWN
A. CARROLL
G. CHAPIN
R. B. CONN
R. D'EVELYN
E. A. EMERSON
M. FRISHBERG
A. FAULKNER
C. FISCHER
M. J. FRIEDENTHAL
H. P. GATES
E. GLASER
R. GOODMAN
M. GRAHAM
W. F. GUNNING
G. H. HANSEN
E. HIGGINS
J. HLEDIK
L. C. HOBBS
G. L. HOLLANDER
P. INGERMAN
R. W. JACK
A. JORGENSEN
P. D. JOSEPH
B. P. KERFOOT
L. KAPLAN
E. F. KLEIN
S. KLEIN
J. L. KOORY
G. A. KORN
S. KOVNAT
P. KRIBS
J. L. KUHNS
M. R. LACKNER
H. T. LARSON
R. LEUSCHNER
A. LEVINE
C. MACON
F. N. MARZOCCO
W. C. MCGEE
R. MCREYNOLDS
J. MURTHA
E. NELSON
A. OPLER
M. OSTROFSKI
J. J. PARISER
R. L. PATRICK
E. R. PETTO
J. A. POSTLEY
N. PRYWES
J. RAJCHMAN
L. C. RAY
A. REICHENTHAL
M. ROSENBERG
G. F. RYCKMAN
F. H. SCAIFE
E. J. SCHUBERT
R. K. SIERING
R. SIMMONS
H. K. SKRAMSTAD
J. R. SMITH
R. SWANSON
J. WEIDENKOPF
R. S. WEISZ
G. WEST
I. L. WIESELMAN
W. O. WOOTAN

PANELISTS

W. BRUNNER
M. E. CONNELLY
E. C. DE LAND
J. EGAN
M. C. GILLILAND
D. G. HAYS
W. J. KARPLUS
S. LAMB
G. H. MEALY
V. C. RIDEOUT
G. F. RYCKMAN
F. H. SCAIFE
T. B. STEEL, JR.
F. TONGE
K. WRIGHT

SESSION CHAIRMEN

W. F. BAUER
G. A. BEKEY
H. GATES
G. L. HOLLANDER
G. A. KORN
F. N. MARZOCCO
A. OPLER
R. L. PATRICK
J. A. POSTLEY
M. ROSENBERG
D. L. SLOTNICK
D. L. STEVENS
R. I. TANAKA
E. O. THORP
F. WAGNER
I. L. WIESELMAN

AMERICAN FEDERATION OF INFORMATION PROCESSING SOCIETIES (AFIPS)

Standing Committee Chairmen

Admissions
MR. BRUCE GILCHRIST
Service Bureau Corporation
425 Park Avenue
New York 22, New York

Awards
MR. SAMUEL LEVINE
Teleregister Corporation
445 Fairfield Avenue
Stamford, Connecticut

Conference
MR. KEITH UNCAPHER
The RAND Corporation
1700 Main Street
Santa Monica, California

Constitution & ByLaws
MR. EUGENE JACOBS
System Development Corporation
2500 Colorado Avenue
Santa Monica, California

Finance
DR. R. R. JOHNSON
General Electric Company
Computer Division
13430 North Black Canyon Highway
Phoenix, Arizona

International Relations
MR. I. L. AUERBACH
Auerbach Corporation
1634 Arch Street
Philadelphia 3, Pennsylvania

Planning
MR. G. L. HOLLANDER
Hollander Associates
P. O. Box 2276
Fullerton, California

Publications
MR. J. D. CHAPLINE
Philco Corporation
3900 Welsh Road
Willow Grove, Pennsylvania

Public Relations
MR. ISAAC SELIGSOHN
IBM Corporation
T. J. Watson Research Center
P. O. Box 218
Yorktown Heights, New York

Social Implications of Information Processing Technology
DR. MORRIS RUBINOFF
Moore School of Electrical Engineering
University of Pennsylvania
Philadelphia 4, Pennsylvania

1963 FALL JOINT COMPUTER CONFERENCE COMMITTEES

Chairman
JAMES D. TUPAC, The RAND Corporation

Vice Chairman
ROBERT A. KUDLICH, AC Spark Plug Division, General Motors Corporation

Conference Administrator
MARVIN HOWARD, Thompson-Ramo-Wooldridge, Inc.

Finance Committee
HERBERT JACOBSOHN, North American Aviation, Inc., Chairman
RAYMOND J. MASON, The RAND Corporation
JOHN SKOWRONSKI, System Development Corporation
O. T. GATTO, The RAND Corporation

Technical Program Committee
PAUL M. DAVIES, Abacus, Inc., Chairman
THOMAS B. STEEL, JR., System Development Corporation, Vice Chairman
J. REESE BROWN, JR., Burroughs Corporation
SAMUEL NISSIM, Thompson-Ramo-Wooldridge, Inc.
HAROLD K. SKRAMSTAD, Naval Ordnance Laboratory
V. J. BRAUN, System Development Corporation

Local Arrangements Committee
EUGENE H. JACOBS, System Development Corporation, Chairman
AL DEUTSCH, Litton Industries
JERRY KOORY, Planning Research Corporation
FRANK B. MEEK, Space Technology Laboratories, Inc.
FRANK O'NEILL, Desert Inn Hotel
ALBERT H. ROSENTHAL, The RAND Corporation
JACK A. STRONG, Computer Sciences Corporation

Exhibits Committee
DAVID F. WEINBERG, Space Technology Laboratories, Inc., Chairman
RICHARD B. BLUE, SR., Space Technology Laboratories, Inc.

Printing and Mailing Committee
LOUIS H. KURKJIAN, Hughes Aircraft Company, Chairman
WILLIAM C. NICKELL, Packard Bell Computer Division, Vice Chairman
SEI SHOHARA, Hughes Aircraft Company, Vice Chairman

Publications Committee
BURT P. WHIPPLE, IBM Corporation, Chairman
DON BREHEIM, IBM Corporation

Registration Committee
EUGENE S. GORDON, System Development Corporation, Chairman
EMANUEL HAYES, System Development Corporation, Vice Chairman
A. V. GRANT, System Development Corporation

Public Relations Committee
PHYLLIS HUGGINS, AFIPS Public Information Director, Chairman
HAL BERGSTEIN, Computer Sciences Corporation
WILLIAM ORR, Thompson-Ramo-Wooldridge, Inc.
CHARLES ELKIND, IBM Corporation, San Jose
WILLIAM MCGUCKIN, General Electric-Phoenix
IRWIN SCHORR, System Development Corporation

Special Events Committee
MARJORIE F. HILL, Control Data Corporation, Chairman
CAROLLEE MATTHEWS, Space Technology Laboratories, Inc.
DAVID KEY, System Development Corporation

Exhibits Management
JOHN L. WHITLOCK

EXHIBITORS
1963 FALL JOINT COMPUTER CONFERENCE

ADAGE, INCORPORATED
ADDRESSOGRAPH-MULTIGRAPH CORP.
AERO MAYFLOWER TRANSIT COMPANY, INC.
AERONEUTRONIC DIV., PHILCO CORP.
AMERICAN DATA PROCESSING, INC.
AMERICAN TELEPHONE & TELEGRAPH CO.
AMPEX CORPORATION
ANELEX CORPORATION
APPLIED DYNAMICS, INC.
AULT MAGNETICS, INC.
BECKMAN INSTRUMENTS, INC.
BENSON-LEHNER CORPORATION
BRUSH INSTRUMENTS DIVISION
BRYANT COMPUTER PRODUCTS
BURROUGHS CORPORATION
CALIFORNIA COMPUTER PRODUCTS
C-E-I-R, INC.
COLLINS RADIO COMPANY
COMCOR, INC.
COMMERCE CLEARING HOUSE, INC.
COMPUTER CONTROL COMPANY, INC.
COMPUTER DESIGN PUB. CORP.
COMPUTER SCIENCES CORPORATION
COMPUTER SYSTEMS, INC.
CONTROL DATA CORPORATION
CORNING GLASS WORKS
DATAMATION MAGAZINE
DATAMEC CORPORATION
DATA-STOR DIVISION
DATA SYSTEMS DEVICES OF BOSTON
DIGITAL EQUIPMENT CORPORATION
DIGITRONICS CORPORATION
DYMEC DIVISION
ELECTRONIC ASSOCIATES, INC.
ELECTRONIC MEMORIES, INC.
ENGINEERED ELECTRONICS COMPANY
FABRI-TEK, INC.
FERRANTI ELECTRIC, INC.
FERROXCUBE CORPORATION OF AMERICA
GENERAL COMPUTERS, INC.
GENERAL DYNAMICS/ELECTRONICS
GENERAL ELECTRIC COMPUTER DEPT.
GENERAL PRECISION, INC.
HITACHI LTD.
HONEYWELL EDP DIVISION
IBM CORPORATION
LFE ELECTRONICS
INDIANA GENERAL CORPORATION
ITT DATA PROCESSING CENTER
LITTON INDUSTRIES, INC.
LOCKHEED ELECTRONICS COMPANY
McGRAW-HILL BOOK CO., INC.
MEMOREX CORPORATION
MIDWESTERN INSTRUMENTS, INC.
MOXON ELECTRONICS CORPORATION
NATIONAL CASH REGISTER COMPANY
NATIONAL CONNECTOR CORPORATION
PHILCO CORPORATION, C. AND E. DIV.
PHOTOCIRCUITS CORPORATION
POTTER INSTRUMENTS COMPANY, INC.
PRENTICE-HALL, INC.
RCA SEMICONDUCTOR & MATERIALS DIV.
RECORDAK CORPORATION
ROTRON MANUFACTURING COMPANY, INC.
ROYAL McBEE CORPORATION
SCIENTIFIC DATA SYSTEMS, INC.
SOROBAN ENGINEERING, INC.
SPARTAN BOOKS, INC.
SYSTRON-DONNER CORP.
TALLY CORPORATION
TELETYPE CORPORATION
THOMPSON RAMO-WOOLDRIDGE, INC.
UGC INSTRUMENTS
UPTIME CORPORATION
WESTINGHOUSE ELECTRIC CORPORATION
JOHN WILEY AND SONS, INC.
WYLE LABORATORIES, INC.

AUTHOR INDEX