

T H E B E L L S Y S T E M

# *Technical Journal*

DEVOTED TO THE SCIENTIFIC AND ENGINEERING  
ASPECTS OF ELECTRICAL COMMUNICATION

---

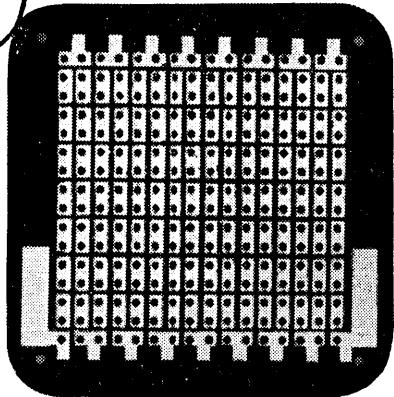
VOLUME XLIII

SEPTEMBER 1964

NUMBER 5, PART 1

---

***NO. 1  
ELECTRONIC  
SWITCHING  
SYSTEM***



# THE BELL SYSTEM TECHNICAL JOURNAL

## ADVISORY BOARD

P. A. GORMAN, *President, Western Electric Company*

J. B. FISK, *President, Bell Telephone Laboratories*

J. E. DINGMAN, *Executive Vice President  
American Telephone and Telegraph Company*

## EDITORIAL COMMITTEE

A. C. DICKIESON, *Chairman*

F. T. ANDREWS, JR.

K. E. GOULD

A. J. BUSCH

W. C. HITTINGER

R. P. CROSS

J. R. PIERCE

R. L. DIETZOLD

D. E. PROCKNOW

R. W. EHRLICH

E. C. READ

## EDITORIAL STAFF

G. E. SCHINDLER, JR., *Editor*

L. M. COLE, JR., *Assistant Editor*

J. T. MYSAK, *Production and Illustrations*

T. N. POPE, *Circulation Manager*

THE BELL SYSTEM TECHNICAL JOURNAL is published six times a year by the American Telephone and Telegraph Company, E. J. McNeely, President, C. E. Wampler, Vice President and Secretary, J. J. Scanlon, Vice President and Treasurer, 195 Broadway, New York, N. Y. 10007. Subscriptions are accepted at \$5.00 per year. Single copies \$1.25 each. Foreign postage is \$1.08 per year or 18 cents per copy. Printed in U.S.A.

# THE BELL SYSTEM TECHNICAL JOURNAL

---

VOLUME XLIII

SEPTEMBER, 1964\*

NUMBER 5, PART 1

---

*Copyright 1964, American Telephone and Telegraph Company*

## Contents, Part 1

- No. 1 ESS: System Organization and Objectives  
W. KEISTER, R. W. KETCHLEDGE AND H. E. VAUGHAN 1831
- Organization of No. 1 ESS Central Processor  
J. A. HARR, F. F. TAYLOR AND W. ULRICH 1845
- Organization of the No. 1 ESS Stored Program  
J. A. HARR, MRS. E. S. HOOVER AND R. B. SMITH 1923
- No. 1 ESS Maintenance Plan  
R. W. DOWNING, J. S. NOWAK AND L. S. TUOMENOKSA 1961
- No. 1 ESS Bus System  
J. B. CONNELL, L. W. HUSSEY AND R. W. KETCHLEDGE 2021
- No. 1 ESS Logic Circuits and Their Application to the Design of  
the Central Control  
W. B. CAGLE, R. S. MENNE,  
R. S. SKINNER, R. E. STAEBLER AND M. D. UNDERWOOD 2055
- No. 1 ESS Program Store  
C. F. AULT, L. E. GALLAHER,  
T. S. GREENWOOD AND D. C. KOEHLER 2097
- No. 1 ESS Call Store—A 0.2-Megabit Ferrite Sheet Memory  
R. M. GENKE, P. A. HARDING AND R. E. STAEBLER 2147

\* This issue of the Bell System Technical Journal was not published until after October 15, 1964.



# No. 1 ESS: System Organization and Objectives

By W. KEISTER, R. W. KETCHLEDGE and H. E. VAUGHAN

(Manuscript received January 22, 1964)

*This paper is an introduction to the No. 1 electronic switching system, a new general-purpose switching system developed for use in the Bell System. Organization and objectives of the system are outlined to provide a background for the detailed technical papers which follow.*

## I. INTRODUCTION

### 1.1 General

The continuing expansion of present-day services, the demand for additional types of service, and the anticipation of new service offerings in the future all indicate a need for a new general-purpose switching system for the Bell System. The No. 1 electronic switching system (ESS) has been developed to meet this need. It is a general-purpose telephone switching machine capable of providing either two-wire or four-wire switching for local, toll, or tandem applications. No. 1 ESS provides today's services as well as several new ones, and is sufficiently flexible to provide for the ever-changing need for future services. It is economically competitive for present applications and will provide new services at relatively low cost.

The purpose of this paper is to outline the objectives of the system and to provide a brief description as background for the several more

technical and more detailed papers which follow. Switching systems are, in general, complex and not easy to describe. No. 1 ESS not only entails many new concepts of switching system organization, but the equipment and apparatus used are also different from those of previous switching systems. As a consequence of this, No. 1 ESS is the largest development project ever undertaken by Bell Laboratories for the Bell System. The papers in this issue should give an adequate description of all the major concepts and techniques used in the system. Other papers, particularly those treating programming and testing aspects, will be published in forthcoming issues of this and other journals.

### 1.2 *History*

Since 1945, Bell Laboratories has conducted an active program of research and exploratory development in electronic switching. The Morris, Illinois, trial<sup>1</sup> of the world's first electronic switching office demonstrated the value of many of the basic concepts used in No. 1 ESS, in particular the use of a stored program and the basic maintenance philosophy. The system at Morris provided customers regular commercial-grade service and some additional new services. The trial established valuable guides for production design in hardware, data formats and programming. In addition, the trial also provided excellent training for many young engineers who later had a major part in the design of the No. 1 ESS. Although all of the hardware units and data processing programs for No. 1 ESS are new, their design relied heavily on the experience gained at Morris. The development of No. 1 ESS has now relegated the Morris system to history.

## II. OBJECTIVES

### 2.1 *Economics*

Essential objectives for a new switching system are that it be at least the equivalent in service features of existing systems and be economically competitive with these systems for some significant segment of the market. These objectives must be attained on the basis of today's market rather than some hypothetical market of the future. The new system must be introduced in an environment of a very large number of older systems, and the general capability of the over-all switching network will be determined largely by the capability of the majority of the offices. The process of achieving improvements in the over-all system must be one of evolution over a period of years. However, the new

system must have greater capability than the older systems in order that this improvement can be realized as the percentage of new offices increases. This points up another important essential. A new office must be compatible with the older offices with which it must operate, since modification of older units in service is inconceivable if we would attain the economic advantages of a new office design.

The economic balance over a range of sizes in the No. 1 ESS plan is quite different from that in electromechanical common control offices such as crossbar. The electronic system is based on a single high-speed central processor which is essentially the same in both large and small offices. In a system like No. 5 crossbar, a multiplicity of control units must be used due to their slower speed. This allows the amount of control equipment — as, for instance, the number of registers and markers in the crossbar system — to be increased as the office grows, so that the full burden of a control capable of handling a large office need not be borne by a small initial installation.

The new system must meet the standards of reliability that have been established by electromechanical systems. The operation of a complete office by a single central processor is a definite problem when reliability is considered. In a multi-marker office of No. 5 crossbar, the failure of a single marker merely reduces the traffic capacity of the office. However, the failure of the controller in an office depending on a single central processor would make the office completely inoperative. The solution followed in No. 1 ESS is to duplicate all units essential to proper operation of the office.

The economic objectives of No. 1 ESS are being realized through a basic design which has been optimized in its details and will be economical in production and in operation over a period of years. The economics of the equipment design have been based on quantity manufacture of its component parts. By the use of the stored program, it has been possible to plan a system which requires no wired options during manufacture and therefore leads to efficiency in production. The system units are designed for a minimum of interconnections between frames, so that most of the wiring and much of the detailed testing can be done at the factory rather than on location during the process of installation. Trouble detection and fault location have been highly automated through the use of stored program. Since most of the system logic involving telephone service features has been placed in the stored program, the introduction of new service features and modifications of existing features will be greatly simplified. In many cases, modifications will be made by changing programs rather than by wiring changes.

It is believed that important savings will be realized in the current development effort required to maintain the system design over a period of years during which its service features and operational environment change.

## 2.2 Size

An important item in planning the system was to determine the range of sizes over which the system would be applicable. Two major parts of the system are greatly influenced by size. One is the network, which must be planned to serve the entire range, with systematic growth from the smallest to the largest size encompassed by the design. The other is the central processor, whose call handling capability must be sufficient to care for the largest office while also being economical when used at only a fraction of its total capacity in smaller offices.

The appropriate size range was determined by making a survey of the range of office sizes. Information for this survey was obtained from the operating telephone companies, who provided information on the number of wire centers and the size of each wire center in each company. This information, covering a total of over thirty million lines in the Bell System, was analyzed by a digital computer to provide statistical information on the makeup of the potential market. The general nature of the results is indicated in Tables I and II. From these it is apparent that there are a very large number of very small wire centers, but that there is a large volume of business in large wire centers. Fifty per cent of the total lines are served from wire centers of 19,000 lines or greater. It was decided that it was not necessary to design an office large enough to accommodate the largest wire centers, because the differential cost per line becomes negligibly small in the higher office sizes. For example, the cost of two 50,000-line units will not be significantly more than the cost of a single 100,000-line unit. For this reason, a decision was made and verified by a number of studies that an upper size limit of 65,000 lines would be reasonable. The lower size limit is determined purely by

TABLE I — TOTAL BELL SYSTEM LINES IN SERVICE VS  
CENTRAL OFFICE SIZE\*

Per Cent of Total No. of Lines	Office Size
75	over 7,500
50	over 19,000
25	over 32,000

\* Data as of January 1, 1960 (includes community dial offices).



TABLE II — CENTRAL OFFICE BUILDING TOTALS VS  
NUMBER OF LINES SERVED\*

Per Cent of Total No. CO Buildings	Lines Served
25	less than 230
50	less than 750
75	less than 3000

\* Data as of January 1, 1960 (includes community dial offices).

economics, but should extend down to at least the 4,000- to 5,000-line size, since the survey showed that a large number of offices are initially installed in this size range.

From a traffic standpoint, the maximum size of the system is set by the capacity of the central processor for handling calls in real time. The maximum capacity of the No. 1 ESS has been set at 100,000 calls in the busy hour. This figure was determined through studies of the cost and complexity of central processors of various designs, weighed against the traffic needs in the wire centers in the Bell System. The capacity of the central processor is determined by the basic speeds of the electronic circuits, its basic clock cycle time, the complexity of its individual logic operations and the amount of processing done in parallel.

### 2.3 Flexibility

The system has been planned to make maximum use of the flexibility inherent in a stored program. Central control is designed so that its wired logic represents basic logic operations which are related to telephone switching functions only through the sequences of instructions in the program. By means of the stored program, most of the logic decisions in call processing have been converted to basic logic operations. This philosophy is extended to items such as trunk circuits. The physical equipment in a trunk circuit is in most cases limited to that necessary for detecting and generating the signals required on these trunk circuits and performing basic switching operations such as loop closure or loop reversal. All sequencing of these operations, including timing of the duration of signals, is performed by central control under instructions from the program. This has the effect of minimizing the variety of trunk circuits required and reducing their cost, since changes in timing or sequence of operation can be made through changes in the program.

The switching network has been designed for flexibility in a variety of situations. The basic pattern consists of line frames and trunk frames connected together by groups of junctors of three types. Junctors are

provided between line frames and trunk frames for connections between lines and trunks. Junctors from line frames back to line frames are used for line-to-line connections, and likewise junctors between trunk frames are used for tandem traffic and other trunk-to-trunk connections. The number of line frames and trunk frames can be varied independently and the junctor group sizes adjusted according to the mix of inter- and intra-office and tandem traffic. The network has been designed so that frames of four-wire ferreed switches can be used without modification of the central processor complex or its basic network control programs. In this way, the single basic system design can be adapted to a variety of local, toll, and tandem applications which in the past have required quite different system designs.

### III. SYSTEM ORGANIZATION

#### 3.1 *Outline of System Plan*

The basic concept of the No. 1 ESS is that of a high-speed electronic central processor operating with a stored program to control the actions of the central office on a time-sharing basis. The general system organization is illustrated in the block diagram (Fig. 1). The switching network provides the means for making interconnections between the lines and trunks to be served by the system and also provides access to the various service circuits required in handling telephone calls. These include tone sources, signaling detectors, ringing sources and the like. All information processing is handled by a central processor consisting of central control and the temporary and semipermanent memories. The temporary memory is used for storage of the transient information required in processing calls, such as the digits dialed by the subscriber or the busy and idle states of lines and trunks. The semipermanent memory contains the stored program and translation information. The contents of this memory need not change during the processing of a call. When the semipermanent information must be changed for any reason, these changes can be made on a manual basis without interfering with the operation of the office. The semipermanent memory also has the advantageous characteristic that its stored information is not erased by circuit malfunctions.

The central control consists of wired logic for performing information processing operations. It is organized on a word basis, with a word length of 24 bits, and operates on a basic cycle time of 5.5 microseconds. It is important to realize that the logic wired in central control is designed for information processing. The telephone switching logic is contained

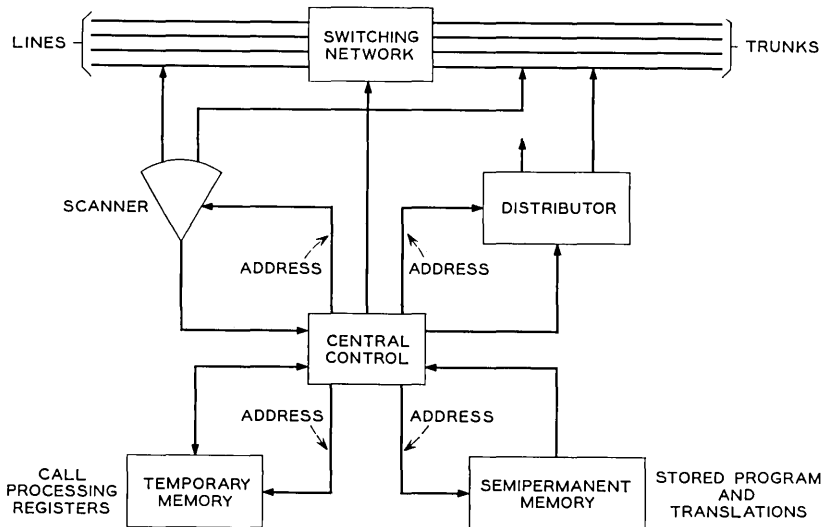


Fig. 1 — Central information processor with stored program (No. 1 ESS).

in the stored program. Thus, the hardware of the control complex is largely independent of the type of telephone service to be provided and the service treatments to be offered to the subscribers. This leads to great flexibility, since the same equipment can be used with different programs to provide a variety of services.

Input information for the central processor is provided by scanners connected to various points in the system where information must be obtained. These include the lines and trunks and signal receivers. The scanners are directed periodically to the lines to detect service requests, to the trunks to detect incoming calls, and to the signal receivers for detecting dialed digits and other control information. The process of recognizing this information involves periodic scanning and recording the results of these scans in temporary memory.

The signal distributor is the inverse of the scanner. It is connected to the various points in the system where actions must be controlled by the central processor. Central control can address the distributor to a particular terminal where a flip-flop or other memory device can be set to start an action. At a later time, central control can address the distributor to terminate this action. In the present design, the system handles this distributor action through two types of units. One is the central pulse distributor, which is all-electronic and is used for high-speed actions. The other, the signal distributor, makes use of a relay tree to perform lower-speed actions such as the control of trunk relays.

Thus the facilities provided in the system can be divided into several categories, such as the switching network with its associated terminal circuits which perform the physical functions required in making connections and detecting and producing signals; the central processor, with its wired logic to perform basic information-processing functions; the scanner and distributor, providing input and output communication for the central processor; and the stored program, which contains instructions for performing all the switching tasks in ordered lists of instruction words.

Particular distinction needs to be made between the program and translation information contained in the semipermanent memory. The program is the lists of instructions for performing all of the service features. It is part of the basic design of the machine and is not influenced by the characteristics of the particular installation. Ordinarily, it will not be changed except for some significant change in features or the sequence in which actions are to be performed, which would correspond to redesign and wiring changes in electromechanical offices. The translation section, on the other hand, contains the specific layout of facilities in the particular office: the association of subscriber directory numbers with the equipment location of their lines, the classes of service to be provided — such as individual and two-party, coin, extended area dialing and so on — and also the specification of trunk routes and their location and alternate routes to be used when available. This information must be changed periodically with the inward and outward movement of subscribers, changes in their class of service, and readjustment of trunk routes. Means are provided for making these changes on a periodic routine basis.

### 3.2 *Hardware*

The equipment and apparatus of No. 1 ESS has been designed for large-volume manufacture at minimum cost. Note that even a volume of only one million lines a year means over 15 million ferreed crosspoints,  $1\frac{1}{2}$  million electronic circuit packages, 5 million transistors, 15 million diodes, etc. Obviously, such production rates must be highly mechanized.

Equipment and apparatus must be of special design to be adaptable to low-cost mechanized manufacture. Further, the high manufacturing volume justifies greater development effort and the creation of special devices and components. Thus No. 1 ESS uses many apparatus items that were developed for this particular application and whose design was tailored for mass manufacture. The ferreed switch is a good example. It is designed to be made as an array, not as an assembly of individual

crosspoints. The control coils are not wound individually and then connected in series. Rather they are wound in simultaneous rows and columns from continuous lengths of wire.

Choice of technology is based on function and cost. In No. 1 ESS, relays are used in substantial quantities. The relay has cost advantages over an electronic approach where speed and function permit. For example, control of ferreed switches involves routing a high-current pulse through a selected path in the matrix of windings. This can be done by means of diodes and PNP triodes, and indeed such a control structure was developed initially. However, by appropriate design of the control circuits one can, in effect, substitute a relay contact for a high-current diode. The cost advantage of the relay approach is substantial.

Standardization and minimization of codes are other important steps to low-cost volume manufacture. In No. 1 ESS strong efforts were made to standardize the hardware and to achieve the necessary variability by program or translation methods. An example of equipment standardization is the network equipment. Only six codes of frames permit assembly of a switching network suitable for any local office. Two additional codes take care of four-wire networks for toll offices. An example of device standardization is the use of only two codes of transistors, the low-power 29A and the relatively higher-power 20D. The 29A is used in the many logic circuits, in audio amplifiers and oscillators, and in broadband feedback amplifiers and regulators. Another effect of standardization has been the virtual elimination of wired options. Most electromechanical switching equipment makes liberal use of wired options to meet the variations of size and features of different installations. Thus the equipment is specially wired and tested at the factory. In No. 1 ESS the corresponding variations are stored in memory. The factory makes a particular frame the same way every time to a fixed set of test requirements.

The choice of seven-foot-high equipment arrangements (instead of 11.5-foot) eliminates the need for special buildings and allows maintenance from the floor, without ladders. This also contributes to the objective of simplified installation and growth. As a part of the general objective of dependability, and to permit reuse of existing buildings, air conditioning is not required. While most offices will doubtless be air conditioned, it will not be required for machine operability.

No. 1 ESS uses unregulated storage battery power fed by silicon-controlled rectifiers. The equipment has therefore been designed to operate over a  $\pm 10$  per cent voltage range. This design eliminates the need for end cells, counter cells and their associated switches.

Transmission has received particular attention. All outgoing trunks include loop compensation networks to improve return loss characteristics. Tones are generated by precision transistor oscillators and are fed to lines and trunks out of precisely balanced terminations. These and other similar measures, including careful control of noise, were taken in recognition of the role of the switching center in the maintenance and improvement of transmission objectives.

### 3.3 *Programs*

Approximately 90 programs totaling about 100,000 words are used to control the operations required for telephone service and to control the maintenance of the system. These programs, each an ordered set of instructions to provide a particular function, are stored in the program store. The call programs provide the solution to any problem a customer can present to the system, either directly or through some other switching system. An assembly of call programs must tailor-make a connection according to the demands of the customer.

Several approaches toward providing programs for a large number of different offices could be used. A generic program, which is the same for each office, with detailed differences listed in a parameter table, is the approach used in No. 1 ESS. The generic program includes all features for a large number of offices, covering sizes from 2,000 to 65,000 lines and means for handling growth and changing traffic conditions. This approach simplifies record keeping, because only the parameter tables which specify present size and operating conditions are unique to each office. Additional data which characterize a particular office are found in translation tables also in the program store. Typically, 18 different sets of translations are required in each office. These include directory number to equipment number translations for both lines and trunks, class of service, and special treatment for lines and trunks.

In the future, economics may dictate the need for several generic programs—for instance, one for small offices, one for large offices, one for four-wire offices, and perhaps some combinations of these.

The development and preparation of programs for the system require the use of several utility programs written for a commercial computer, an IBM 7094 in this case. These programs are used to convert the language of the programmer to the language of the machine, to assemble and compile the individual pieces of call and maintenance programs, and to load information onto a tape which finally controls the writing of the magnets on the twistor cards. Additional programs are used to assemble, compile, and load parameters and translations. Utility pro-

grams will be used by the Western Electric Co. as tools for manufacturing the programs put into commercial installations.

#### IV. DEPENDABILITY

##### 4.1 *Objectives*

The dependability requirement on a telephone office is limited only by what the state of the technology can provide. Certainly a new system must at least be comparable to existing offices. This means outages of no more than a few minutes in 40 years. Heretofore no large electronic machines have been able to approach this kind of dependability. In fact, the dependability objective represented one of the major challenges of the No. 1 ESS development. Since No. 1 ESS is a large digital information processor, it is a cousin to the general-purpose digital computer. However, the dependability requirement requires that No. 1 ESS be a very different kind of system with a much higher level of redundancy.

##### 4.2 *The Large Immortal Machine*

The large size of No. 1 ESS and the unique dependability requirement lead to the term "large immortal machine." It also implies that No. 1 ESS is a new kind of information processor, a kind that has never been developed before.

Another way of contrasting No. 1 ESS with a computer is to compare the relative hazards of a machine data processing error and a total machine failure. In a general-purpose computation center a machine stoppage is a nuisance: the problem must be rerun. But a data processing error could be called a "disaster" because the results come out wrong. In a telephone office it is the other way around. A data processing error may cause a particular call to be mishandled. This is a nuisance, particularly to the customer whose call must be redialed. A total machine failure in a telephone office, however, means no telephone service during the outage, and the magnitude of such a disaster need not be argued. The key point is, of course, that the dependability requirement on No. 1 ESS is both remarkably severe and quite different from that of general-purpose computers.

##### 4.3 *Duplication*

Since some failures of individual components are bound to occur over decades of system service, duplication is essential. Every system unit required to maintain service must be provided in duplicate. Not only

must there be duplication, but troubles must be found and corrected quickly to minimize exposure to system failure due to multiple troubles. This implies that all units must be continually monitored so that trouble in the standby can be found just as quickly as in the unit giving service. This further implies automatic detection of troubles and automatic switching of service when units fail. These processes are covered in detail in other papers.

#### 4.4 *Repair*

When a trouble occurs, the telephone actions are interrupted as briefly as possible to reestablish an operational machine. Then, on a less urgent basis, the defective unit is diagnosed by the system itself and the results printed on the maintenance teletypewriter.

Where offices are unattended during at least part of the day, alarms and a remote teletypewriter are provided at some other location where 24-hour attendance is available. For minor troubles, not affecting service, the repair can be deferred. A defective trunk circuit can be taken out of service with a teletypewriter message. Major troubles may, on the other hand, need prompt attention to insure service continuity.

#### 4.5 *Maintenance Programs*

More than half of the stored instructions are used for maintenance programs. Some of these programs, in conjunction with logic wired into the hardware, detect and report faults and troubles; others control routine tests and diagnose troubles and control emergency actions to insure a satisfactorily operating system, either by eliminating faulty subsystems or by reorganizing usable subsystems into a new operating combination. The classes of maintenance programs are arranged in a hierarchy of interrupt levels, so that when an error is detected the control system can be forced to stop what it is doing, keep a record of where it is, and after proper maintenance action is taken, restore itself to proceed with normal operation. In the hierarchy, a higher class can interrupt any lower class of maintenance or operating program.

### V. TESTING

The separate development of major sections of hardware as individual subsystems, and of programs as other subsystems, and their subsequent successful combination into an operable system present major problems. First, the operation of all hardware subsystems working together to



form a single system must be verified. Each of the hardware subsystems is factory tested by a combination of manual and automatic tests. The over-all system operation is checked by a special set of programs known as X-ray programs, which check step-by-step the sequences of operation. Such programs will also be used by Western Electric installation people for checking new installations.

This procedure was also used on the Holmdel, New Jersey, No. 1 ESS as the first over-all check of the system design, and it was here that many interface problems were solved. The solution of these required changes in hardware, program, or both. After the hardware was checked and operating satisfactorily, testing of the call and maintenance programs was begun on the actual system. However, some simulation was usually done on the IBM 7094 computer before the program was submitted to the No. 1 ESS laboratory system. When all programs operate satisfactorily, the system will be tested under simulated traffic conditions and finally under actual traffic conditions.

#### VI. STATUS

At the time of writing of this article, all of the hardware has been tested by means of the test programs and proved to operate satisfactorily. Many of the call and maintenance programs are now operating on both the laboratory and Succasunna, New Jersey, systems. Other programs are still in preparation. Maintenance programs are usually the last prepared because they must be designed after the operating peculiarities of the hardware have been identified. Publication of untested plans, in general, is dangerous. However, some of the authors of the following articles, on programs as yet only partially tested, have decided to present their plans so that a relatively complete story of No. 1 ESS can be presented under a single cover. When the plans have been fully tested, additional details will be reported.

#### VII. SUMMARY

This paper has attempted to present a general outline of the organization and objectives of No. 1 ESS. It also serves as an introduction to the papers that follow.

#### VIII. ACKNOWLEDGMENTS

Because No. 1 ESS is such a large undertaking the efforts of many hundreds of people are involved. Even listing the numerous organiza-

tions within Bell Telephone Laboratories would be difficult and this would exclude the many people in the Western Electric Company, the American Telephone and Telegraph Company, and the operating companies who have contributed in important ways.

REFERENCE

1. Keister, W., Ketchledge, R. W., and Lovell, C. A., Proc. IEE, **107**, Part B, Suppl. No. 20, 1960.

# Organization of No. 1 ESS Central Processor

By J. A. HARR, F. F. TAYLOR and W. ULRICH

(Manuscript received January 28, 1964)

*The central processor controls the operation of the No. 1 electronic switching system by executing sequences of program instructions. The logical organization of the central processor is described by the simultaneous evolution of:*

*(1) an instruction repertoire to carry out the required telephone and system maintenance tasks efficiently, and*

*(2) a circuit logic design to provide the necessary circuits (flip-flop registers, accumulators, etc.) to execute the instructions at a high data processing rate.*

*The design aims, order structure, timing, internal sequencing, and communications with the peripheral equipment of the system are described.*

## I. INTRODUCTION

Telephone central offices must cover a wide range of sizes and provide a large variety of services to customers; in addition, they must be compatible with existing systems, adaptable to varied and changing operating conditions, dependable, reliable, and economical. The development of an electronic switching system capable of satisfying these requirements presented many new problems to the designers. As a result, many techniques new to the telephone switching field were introduced in the system.<sup>1</sup> One of the most important new techniques is the control philosophy, which utilizes a stored program.

A system employing a stored program is one which consists of memories for storing both instructions and data, and a logic unit which monitors and controls peripheral equipment by performing a set of operations dictated by a sequence of program instructions. The stored program philosophy permitted the designers to use centralized logic circuitry and large-capacity memory units as a means of attaining flexibility and over-all economy in the system.

In this paper some of the design characteristics of the central processor are described, followed by a step-by-step development of the central control order structure and the corresponding logic circuitry needed for its implementation.

II. DESIGN CHARACTERISTICS

A simplified diagram of the electronic switching system (ESS) is shown in Fig. 1. The outer circle represents the entire No. 1 ESS, having primary inputs from lines and trunks<sup>2</sup> via scanners,<sup>3</sup> and outputs to the network<sup>4</sup> and signal distributor,<sup>3</sup> with teletypewriters as administrative input-output devices and with a magnetic tape for automatic message accounting (AMA)<sup>5</sup> output. The inner circle, the central processor, contains a central control<sup>6</sup> unit which executes program instructions and memory units used for storing program instructions and data.

In order for the central processor to handle traffic submitted by offices

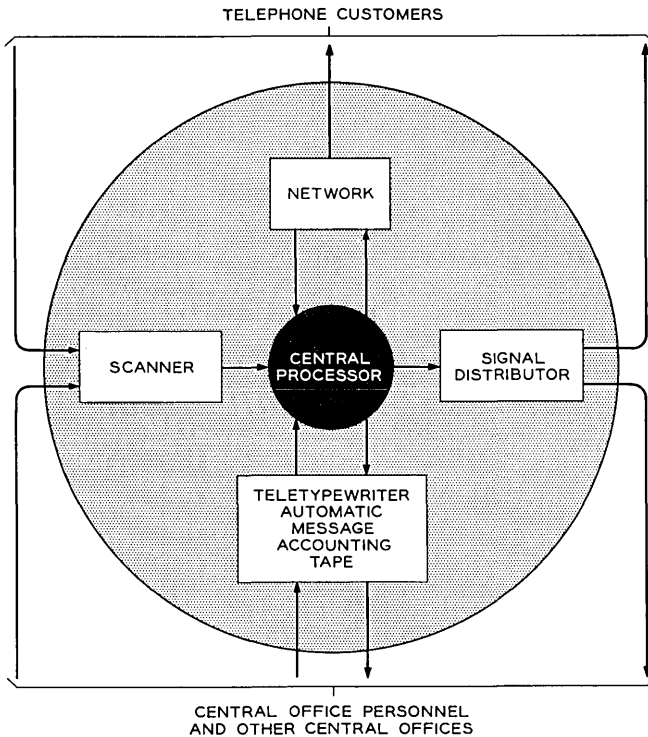


Fig. 1 — Simplified diagram of the No. 1 ESS.

serving 5000 to 60,000 customers, the number of memory units in the system must be expandable over a wide range. Therefore the central processor address registers, memory word size, and address buses are designed to accommodate the largest office. The system design must not only meet initial office size requirements, but must also include growth capability.

Since the data processing operations required for performing telephone functions are accomplished by executing a stored program<sup>7,8</sup> which must be error-free and remain error-free\* at all times to insure the proper behavior of the system and good service for the customers, the memory chosen for storing the program is semipermanent and requires off-line operations to change it. (This avoids the risk of an error in operation introducing an error in the program.) Therefore the central processor contains two types of memory: a semipermanent memory system (program store)<sup>9</sup> for storing programs and a high-speed readable and writable memory (call store)<sup>10</sup> for storing call progress data.

An address of at least 21 binary bits is needed to gain access to the total number of words required in both the program and temporary memories. To meet this requirement, along with memory store design considerations, the designers decided upon a word length in the call store of 24 bits, comprising a parity check bit and 23 information bits. For convenience, the same word length is used throughout the central control. Most instructions which operate on memory words require a 21-bit address field. Instructions which contain a data field must allow for 23 bits in the data field to be compatible with the length of temporary memory words. Accordingly, the operation fields of each of these types of instructions are 16 and 14 bits in length, respectively, to accommodate the many types of instructions needed in each category. Therefore the instruction word length is 37 bits. To check and correct program instructions with single errors and to determine whether a word read has a double error in it, 7 check bits are also needed for each program instruction, making a total of 44 bits stored for each entry in the program store.

Engineering studies of the number and kind of telephone functions which the system must perform dictated a need for an efficient instruction repertoire with specific attributes. Some of these attributes will now be described.

To perform the functions required to handle the busy-hour traffic submitted by the largest office, the system can spend only approximately

---

\* As will be seen later, facilities are available for detecting and correcting single errors.

5000 machine cycles per call. To meet this requirement, the instruction repertoire must include efficient multifunctional program instructions.

Since the repertoire must include the storage of data of variable bit length in the temporary memory and retrieval of these data, both read-from-memory and write-to-memory instructions include masking facilities. As used here, the masking of words read from memory means changing all of the bits of the word to zero except those which specify the item of interest; these remain in the same state as they were in memory. Masking of words written into memory is a facility for inserting an item of variable length into a word already in memory; the remaining bits of the word are unchanged.

To assist in achieving program word efficiency, in most cases the same functional program is used for all calls in progress requiring the execution of a given function.<sup>7,8</sup> For reliability, instructions in this system are not changeable at run-time (i.e., while the machine is actually processing calls). Indexing facilities provide means by which the programmer can change the addresses of memory words acted upon by the same program. Also, the indexing facilities can be used to vary the sequence of programs to be read and executed.

To carry out data processing of call information, the repertoire includes the arithmetic operations of addition, subtraction, comparison, shifting, and rotation, and the following logic operations: AND, OR, EXCLUSIVE-OR, and COMPLEMENT. Since the functions which the system must perform do not require multiplication and division operations, it was not necessary to include these in the central processor. To perform the logic functions required to carry out the many telephone functions, a variety of decision instructions are required to transfer control to specific program sequences based on the condition of internal central control registers after data manipulations have been performed on them. For example, a program of three instructions capable of performing the following three operations illustrates the primary way the central processor can vary the sequence of its operations according to input data it has received.

(1) Read, at an address specified by an index register, the word from memory containing the first dialed digit of a call; mask out all of the bits in the word other than the four bit positions used for the first digit; and load the word into an accumulator register.

(2) Compare the word just loaded into the accumulator with the value 10 (i.e., the number of pulses counted when a customer dials zero) specified by the data field of the instruction.

(3) Transfer to the program specified in the address field of the in-

struction if the two compared quantities are equal (this program will cause the customer to be connected to an operator); otherwise, continue the present program sequence.

For efficient operation, the central control should be capable of executing instructions which combine a number of the operations listed above. For example, the repertoire includes an instruction which reads the word at a temporary memory address specified by an index register, masks the word read, complements it, and then AND's the result with the accumulator register in the central control during one operational cycle.

To make efficient use of data processing time the repertoire must include a class of special instructions designed to facilitate the reading and making of logical decisions on input data and special orders for delivering output data to both the network controllers and trunk control circuitry. Therefore the central control can be described as an input-output processor superimposed on a general-purpose data processor.

Since the fundamental task of the central processor is continuous monitoring and controlling of its rapidly changing environment, consisting of wide variations of traffic submitted by customer lines and trunks, real time must be carefully considered when planning the system and writing the program. Programs to monitor and gather input data and to deliver output signals and data must be especially efficient in their use of central processor cycle time.

Input-output programs must be executed on a strict schedule. For example, the program to detect and receive dial pulses must be performed every 10 milliseconds. In order to accomplish this, an interrupt mechanism is included in the central processor. The interrupt mechanism, when activated by a source such as a clock or check circuitry, generates a new program address, thus transferring control of the system to an interrupt program sequence. When this occurs, the address of the next instruction which normally would have been executed is automatically stored. When the interrupt program completes its task, control is returned to the interrupted program.

To make the system capable of continuous operation during malfunction of circuit components, the central processor is duplicated. In order to detect and ultimately pinpoint hardware malfunctions, the central processors include:

- (1) circuitry which compares the execution of instructions in both central processors,
- (2) circuitry for generating a parity bit for each word stored in the temporary memory,

(3) circuitry for checking the parity of words read from temporary memory,

(4) circuitry for detecting and correcting single-bit errors in instructions read from the program memory,

(5) circuitry within the peripheral equipment for checking data received and for notifying the central processor when either data it receives or its own check circuitry indicates trouble,

(6) circuitry within the central control for verifying signals sent back by peripheral equipment, and

(7) circuitry for special-purpose instructions for controlling and interogating stores and peripheral equipment.

### III. BASIC DESCRIPTION OF CENTRAL CONTROL

In this section an order structure, including the symbolic names of the instructions, and a central control block diagram will be concurrently explained.

#### 3.1 *Basic Facilities*

In the simplest form (see Fig. 2), the central processor consists of a central control, a program store which receives an address from central control and returns the corresponding instruction, and a call store, which receives an address and either receives data to be recorded at that location or returns the data previously stored at that location.

As a starting point, central control must contain registers for receiving program store instructions and call store data, and facilities for generating and transmitting addresses to both stores and data to the call store (see Fig. 3). The buffer order word register (BOWR) receives instructions from the program store, and the data buffer register (BR) receives data from the call store. The program store address register (PAR) is

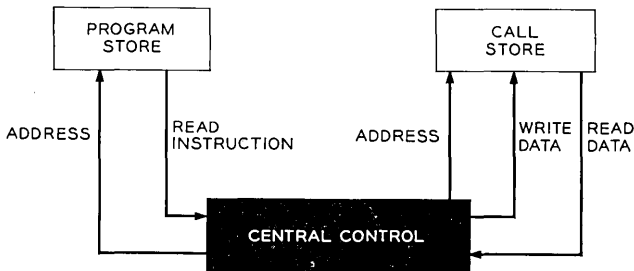


Fig. 2 — Block diagram of the data processor.



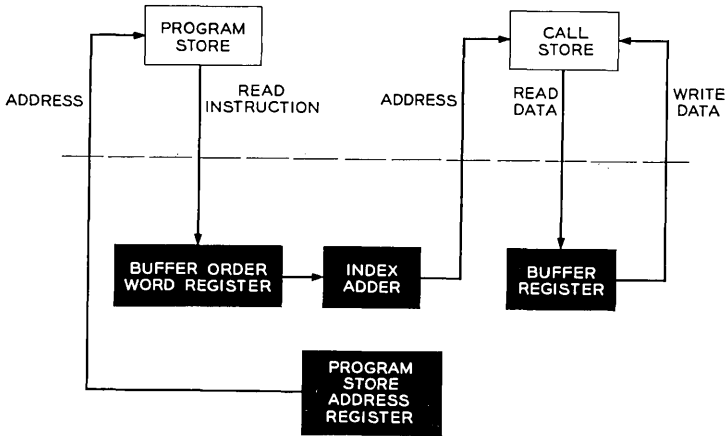


Fig. 3 — Derivation of the central control (1).

used as the source of addresses of instructions to the program store, and the index adder is used to generate addresses for the call store.

These facilities must be augmented by an instruction decoder, the buffer order word decoder (BOWD) attached to the buffer order word register (see Fig. 4). This decoder is used to control the gating of information inside central control.

Also needed to control such gating and to synchronize the decoding with the reading of information from the stores is a clock. In the No. 1 ESS central control, the clock is a synchronous 5.5-microsecond clock, with 22 distinct phases separated by 0.25 microsecond. Arbitrary-length clocking pulses are derived by setting a flip-flop circuit with one arbitrary phase, resetting it with another, and using the output of the flip-flop as the clocking pulse. Such pulses are repeated once every 5.5 microseconds.

Gates are therefore controlled by decoding the output of the buffer order word register and combining the decoded output with a suitable clock pulse.

### 3.2 Index Registers

In addition to the buffer register, there are a number of general-purpose flip-flop index registers, F, X, Y, and Z (see Fig. 4). The index registers are 23 bits long, the basic word length of the central control and the call store. (The 24th bit of the call store, a parity check bit, does not store useful information; this bit need not be carried along in central control data processing, although it must be generated anew when-

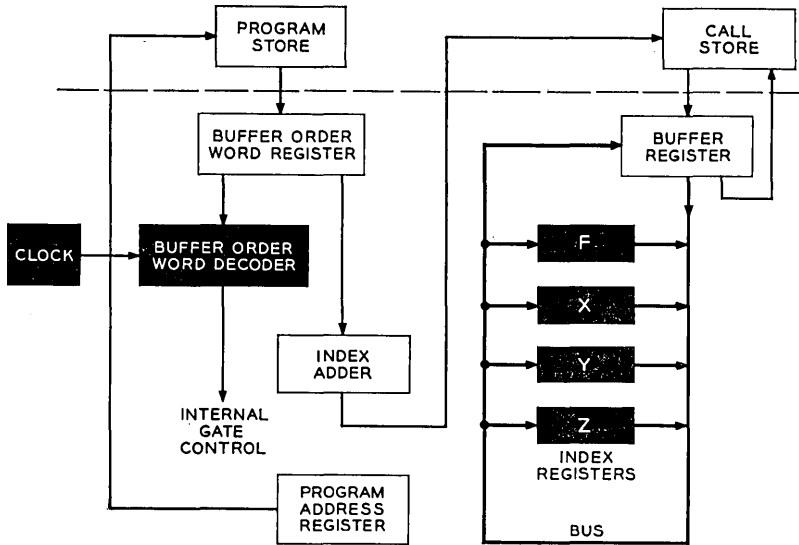


Fig. 4 — Derivation of the central control (2).

ever information is stored in the call store and checked whenever information is read from the call store.)

Indexing is useful for developing a program which is general for any telephone central office. It is the process of deriving a memory address by adding a constant from the instruction to a variable previously derived and stored in an index register. For example, the index register might contain the starting address of a block of call store words used for accumulating dialed information; the constant might be the location of a word within such a block, containing information known to be needed at a certain stage of a call. Such an instruction may be described as follows: fetch the call store reading in the third word of a block whose starting address is stored in the Y index register, and store the reading in the X index register. The description of the operation is divided into three parts (see Fig. 5): the basic operation (fetch data from memory

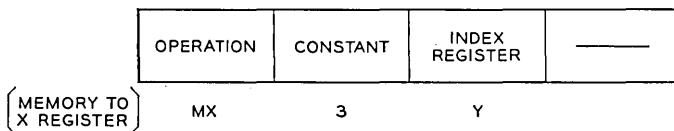


Fig. 5 — Basic instruction format 1.

and store in the X register), the constant of the instruction (3), and the index register used in indexing (Y). In the basic mnemonics of the system, memory to X register is written as MX; therefore this instruction is written as MX,3,Y. Instructions which read or write in memory contain M in their mnemonic representation and are collectively designated M instructions.

3.3 Bus

Information is transmitted among the index registers via a bus (Fig. 4) consisting of 23 parallel information paths. The F, X, Y, and Z registers plus the buffer register, which can also be treated as an index register, all have output gates to and input gates from the bus.

3.4 Index Adder

In order to perform indexing, an adder (see Fig. 6) is required. The index adder receives one input (the constant of the instruction) from the buffer order word register, and the other input (the variable, i.e., the contents of the specified index register) from the bus. The output

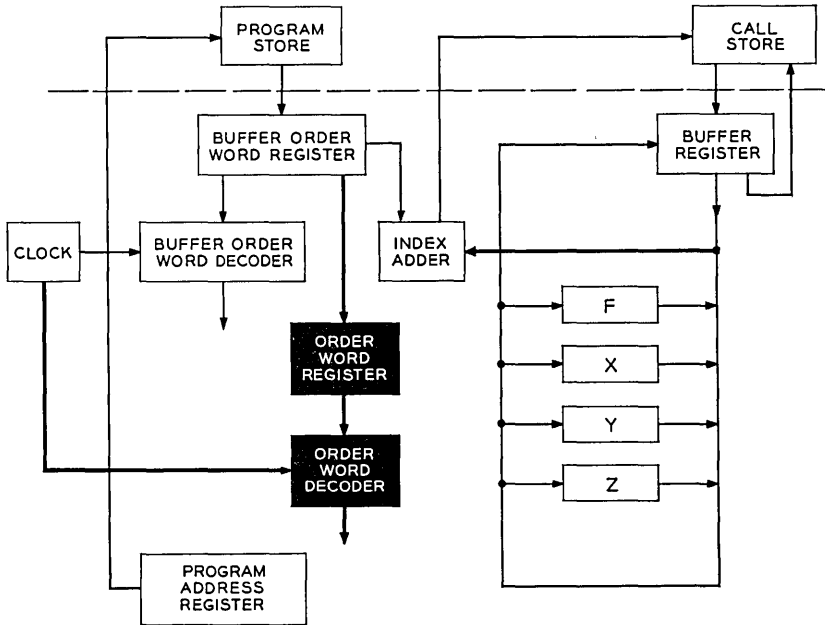


Fig. 6 — Derivation of the central control (3).

of the index adder, as previously indicated, is the source of addresses to the call store.

### 3.5 *Order Word Register*

The basic cycle time of the program store is 5.5 microseconds, and the maximum time from the reception of an instruction such as MX,3,Y until the specified reading from a call store is in the data buffer register is about 6.0 microseconds. When one considers the additional data processing of a call store reading after it has been received by central control, it can be seen that the execution time of an instruction occupies major fractions of two machine cycles. Thus there is overlap in the execution of two consecutive instructions. This overlap is described in Section VIII. An order word register and decoder are therefore provided to control part of the execution of an instruction. The buffer order word decoder and the order word decoder simultaneously control the execution of two consecutive instructions.

The buffer order word decoder controls the addressing of the call store. On a reading instruction, the order word decoder controls the gating of information from the call store to the data buffer register and thence, via the bus, to the destination register; on a writing instruction, the order word decoder controls the gating of data from some source register to the data buffer register and thence to the call store. The actions of the two decoders are sufficiently independent that the division of decoders into a buffer order word decoder and an order word decoder does not cost very much compared to the use of a single decoder.

### 3.6 *Masking: Logic Register, Unmasked Bus, Masked Bus, and Mask Circuit*

The 23-bit word length is much longer than many of the basic quantities of data. A long useful quantity of data is a 21-bit memory address. A typical short quantity is a single binary-coded decimal digit, 4 bits long; several such short quantities may be packed in a single word. In order to treat partial words efficiently, the central control has masking facilities (see Fig. 7). Since most data words pass over the bus, a single mask circuit on the bus accomplishes most of the masking functions. The mask circuit has two inputs, the unmasked bus, which is connected to the gates at the outputs of index registers, and the output of a logic register whose chief function is to control the masking function. The output of the mask circuit is called the "masked bus" and is connected to the input gates of index registers.

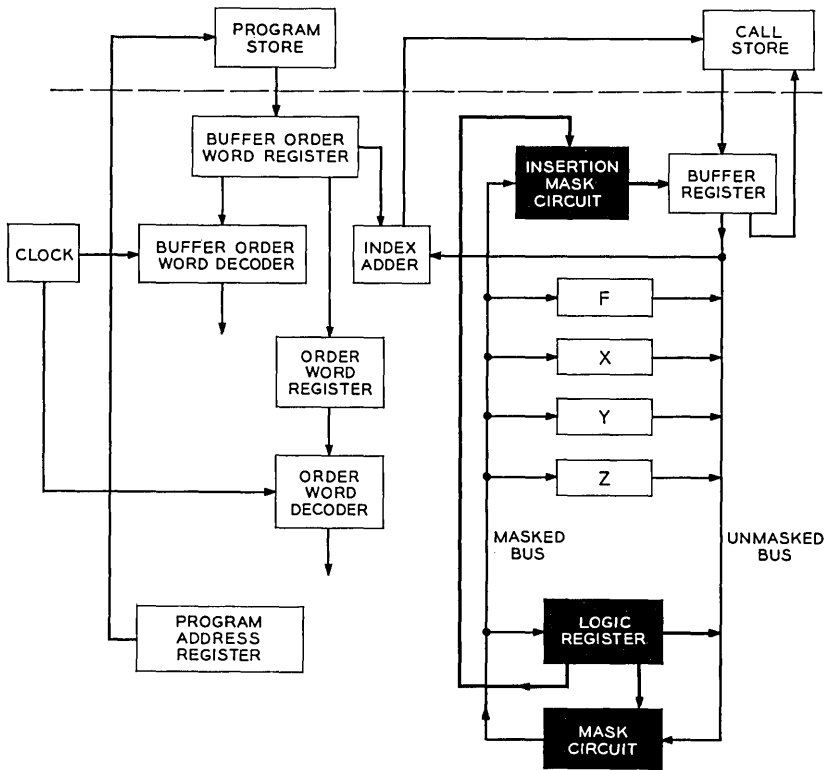


Fig. 7 — Derivation of the central control (4).

The logic register is 23 bits long; each bit controls the masking of 1 bit on the bus. The logic register is itself connected to the unmasked and masked bus so that it may be controlled and read as easily as any index register.

Masking is an option of most instructions. If in the previously described instruction (MX,3,Y) only the four least significant bits were of interest and the logic register were already set up with these four bits equal to one and the rest of the bits equal to zero, then by specifying masking only the four least significant bits would be transmitted to the X register (see Fig. 8). The rest of the bits would be transmitted as zero. This form of masking is called PL masking (P = product, L = state of the logic register; therefore, product with the state of the logic register). This instruction would then be written as MX,3,Y,PL.

	OPERATION	CONSTANT	INDEX REGISTER	MASK AND INSERTION
	MX	3	Y	PL (MASK)
(X REGISTER TO MEMORY)	XM	3	Y	EL (INSERTION)

PL OR EL WILL USE THE CURRENT CONTENTS OF THE LOGIC REGISTER L WHICH WAS SET BY A PREVIOUS INSTRUCTION

Fig. 8 — Basic instruction format 2.

### 3.7 Insertion Masking

Another form of masking that is used frequently is insertion masking. Insertion masking permits all but a selected group of the bits of a certain register to remain unchanged. The selected group of bits is then set up according to the instruction. Because of the requirement that certain bits remain intact, it is convenient to associate the insertion mask circuit with only one of the registers. The most logical choice is the buffer register, since insertion masking is most frequently used when only a portion of a word in the call store is to be altered. The insertion mask circuit is also controlled by the logic register, since in most cases the bits to be inserted and the position associated with these bits have been set up in the logic register for some previous masking (PL) operation. Insertion masking is indicated by specifying EL masking. (E = insertion, and L = the state of the logic register.) If, for example, the four least significant bits of the X register are to be stored in the address  $Y + 3^*$  while leaving the other bits of that memory location intact, this action could be performed with the following two-step program (provided that the logic register is already set up to 1's in the four least significant bits and 0's elsewhere): MB,3,Y (read the contents of memory at the address  $Y + 3$  into the data buffer register); XM,3,Y,EL (insert the contents of X into the BR for all bit positions of the logic register equal to one, leaving the rest of the bits of the BR intact; then write the buffer register into memory at address  $Y + 3$ ). If PL, instead of EL, masking had been specified, the contents of memory would be all 0's except in the four least significant bits; by specifying EL the upper bits remain the same as they appear in the BR.

There is no circuitry available at the call store for performing the equivalent of insertion. Therefore, insertion into memory must always

\* For simplicity, the following convention is used in this paper: the contents of a register, such as Y, plus a constant, such as 3, are represented by an unbracketed expression, such as  $Y + 3$ .

be a two-step operation: the first step to read the word at which information is to be inserted; the second step to insert this information and then write a complete word back. Insertion is entirely a central control function, not a store function.

### 3.8 *Transfer Facilities*

So far the details of addressing a program store have not been shown except for a program store address register (PAR) which transmits such an address. In the program, one of two things can happen. Normally, the program advances from one instruction to the next, so the contents of the program store address register are simply incremented by one. This is accomplished by attaching an increment circuit to the program store address register (see Fig. 9). (The program stores themselves do not have any incrementing facilities. A program store is always addressed with a complete address.) However, sometimes in a program a transfer is necessary. A transfer instruction is an instruction which causes the program to go to another set of instructions, not the immediately following instruction. The most convenient source of the address to which the program would transfer is the output of the index adder, since this is the place where the contents of the instruction are combinable with the contents of registers and thus indirectly with memory readings. A connection from the index adder to the program address register is therefore provided (see Fig. 9).

Direct transfers are transfers to an indexed address. Indirect transfers — i.e., transfers to an address stored in memory, the memory being read through the use of an indexed address — are also possible in No. 1 ESS. An indirect transfer is indicated by an M suffix in the index register field. An input (for simplicity, left out of the diagram) from the call store to the buffer order word register transmits the transfer address to the index adder, thence to the program store address register.

### 3.9 *Complement Option*

Another option existing in the system is the complement option. When numbers are considered numerical rather than logical data, a negative number is stored as the complement of the positive number whose absolute value is the same. The most significant bit, 22, is the sign bit of the entire quantity. This means that both a positive and a negative quantity 0 exist in the system, since the complement of all 0's (+0) is all 1's (-0). Such a system has the advantage of having very simple adder circuits, even though it does introduce occasional programming

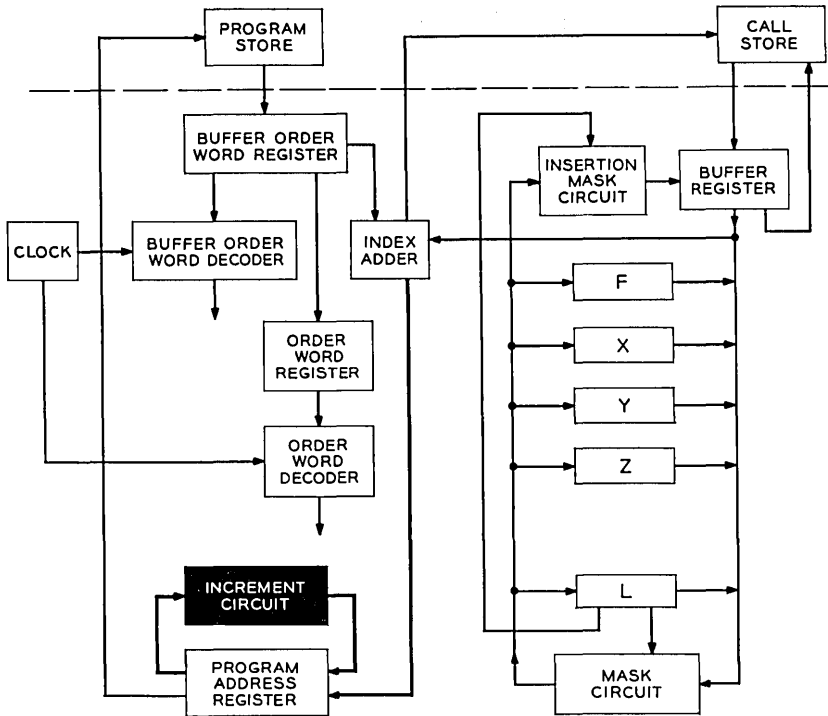


Fig. 9 — Derivation of the central control (5).

problems. The complement circuit is in series with the mask circuit (see Fig. 10); masking takes place before complementing. The complementing is specified as part of the mask field. Thus if we wish to load into X the masked and complemented contents of memory found at the address  $Y + 3$  we may specify this instruction (see Fig. 11) by writing  $MX,3,Y,PLC$ .

### 3.10 Data Instructions

So far only instructions which deal with memory readings have been considered. There is another large class of instructions which deal with internal data manipulations and with the setting up or altering of registers by some constant (data word) within the instruction. These instructions are defined as W (for word) instructions. (It is important to bear in mind the fundamental property that instructions in this machine are not variable. A constant in an instruction is truly a constant until such a time as the program itself is altered, which can be done only by



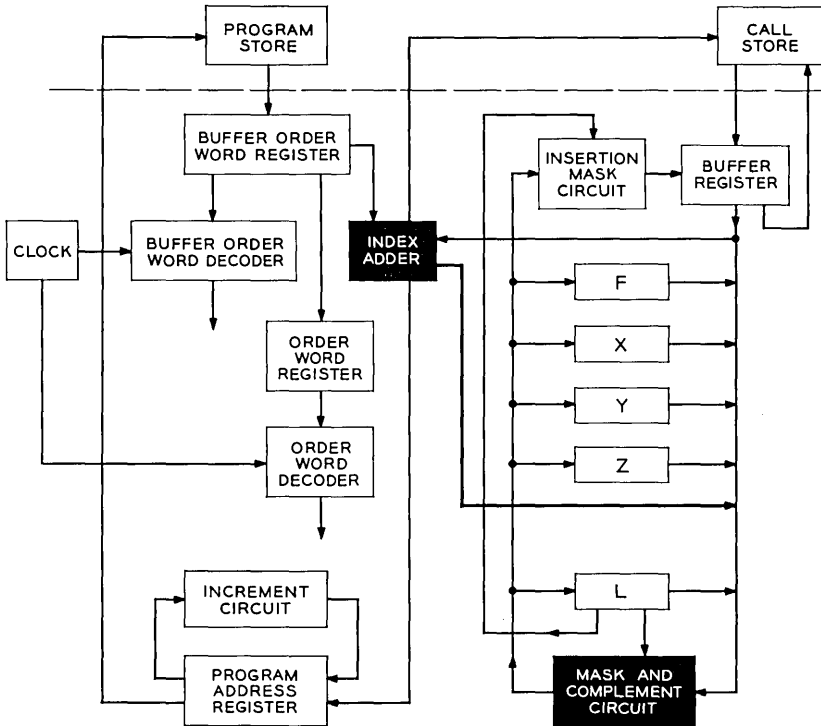


Fig. 10 — Derivation of the central control (6).

writing new permanent magnet twistor cards.) A natural channel for performing such data manipulations is via the index adder. Therefore the index adder has an output onto the unmasked bus (see Fig. 10). Thus, for example, we can increment register X by a constant by gating the X register to the index adder, adding the constant of the instruction and gating the output of the index adder via the bus back into the X register. All these operations are performed by the instruction WX (W equals indexed word, i.e., output of the index adder). This instruction is executed by circuit actions equivalent to generating a mem-

OPERATION	CONSTANT	INDEX REGISTER	MASK INSERTION AND COMPLEMENT
MX	3	Y	PLC
			( MASK AND COMPLEMENT )

Fig. 11 — Basic instruction format 3.

ory address, except that the address is gated to the unmasked bus instead of the memory.

W instructions are also maskable, since the output of the index adder has to go through the mask circuit before it arrives at the destination register. Thus the instruction WX,3,Y,PL takes the Y register, increments it by 3, and places the result in the X register after first masking it according to the present contents of the logic register.

### 3.11 Accumulator

The central control must perform many additions and logical combinations of two quantities. It is convenient to use one register as an accumulator (K) and to permit this register to be combined readily with masked memory and W-type data. The accumulator adder (see Fig. 12)

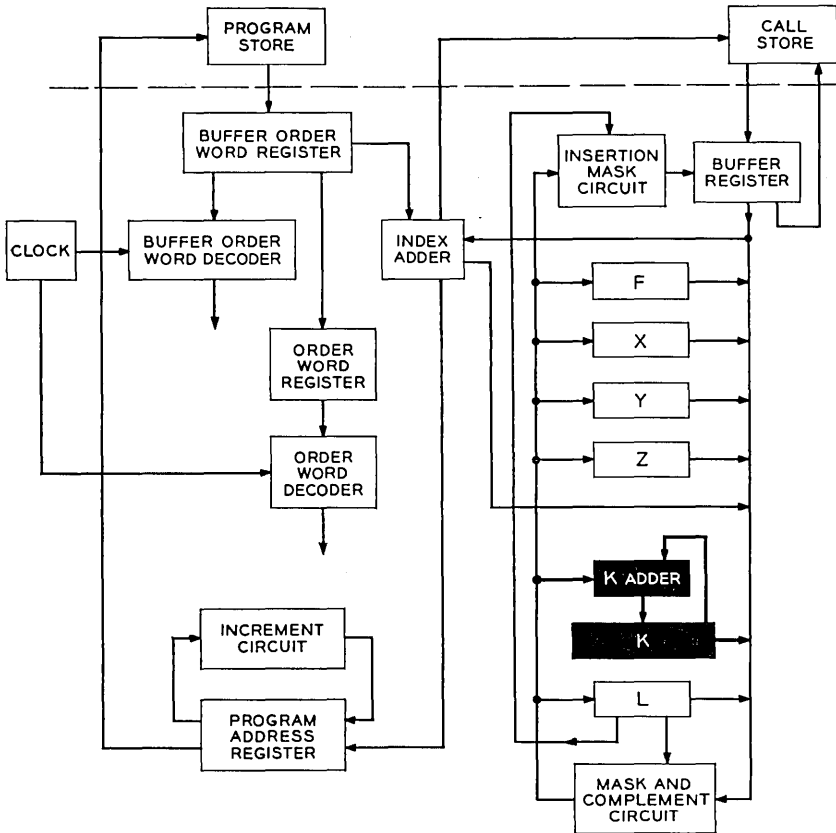


Fig. 12 — Derivation of the central control (7).

is capable of combining the present contents of the accumulator with indexed data or memory readings, both optionally masked, by adding, ANDing, ORing, or EXCLUSIVE-ORing the two. The timing problems are sufficiently severe that a single adder system cannot serve both as an index adder and an accumulator adder for combining two data operands. Accordingly, central control contains two adder systems to handle both operations concurrently.

Data in the accumulator can also be shifted and rotated. The shifting is not usually done for multiplication purposes, but to line up two items of information found in different positions within two data words to a position where the two items may be logically combined or treated in some other standard manner. For example, a pulse count for a decimal digit may always be accumulated in bit positions 0 through 3 (see Fig. 13). However, it may have to be stored in positions 4 through 7, or 8 through 11, or 12 through 15, according to which digit of a number it represents. To get data accumulated in positions 0 through 3 to positions 4 through 7, a shifting operation is necessary.

The rotation operation is similar to the shift operation except that for a left rotate the contents of the most significant bit, instead of being shifted out, are shifted back into the least significant bit, and vice versa for a right rotate. A special-purpose rotation within 16 positions of K is also available in central control. This rotation is extensively used in the network path hunt program.

Shifting is also performed very frequently when a number is composed of two parts, the first part indicating the location of an appropriate table of information and the second part indicating the location within that table.<sup>11</sup> For example, a line equipment number consists of a line link network and line switch frame indication, which is used to find a table, each table containing line translation information for one line switch frame and a position within that line switch frame which is used to find the line translation information within such a table. Without the ability to split such numbers into parts, it would be very diffi-

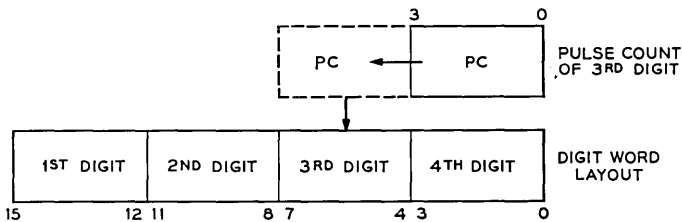


Fig. 13 — Use of shifting.

cult to organize the memory layout of the system for both rapid access and compact storage.

### 3.12 *Data from the Program Store*

So far the simplifying assumption has been made that data always come from a call store and instructions always come from a program store. In practice, however, much of the data is stored in the program store — specifically, the translation data. The process of reading a program store for data is a complicated one, especially in view of the overlap operation that is used.

The same types of instructions are used to read data from the program store and from the call store. This helps programming, since it does not fix a memory location at the time the program is written and helps to relieve the programmer from the burden of considering two different types of memories. A memory address decoder (see Fig. 14) connected to the output of the index adder recognizes when the output of this adder specifies an address that is not in the call store but is in program store.\* It triggers a sequencer (see Fig. 14) which takes care of a special group of operations to be described below.

A sequencer is necessary to prevent the data that are coming from the program store from being incorrectly interpreted as an instruction. This sequencer must cause the program store to be read at the address specified by the output of the index adder and must then go on to the next instruction.

Fig. 15 shows the contents of the buffer order word register, order word register and program store address register during the processing of an instruction for reading data from the program store. The instruction is  $MX, BB, Y. \dagger Y + BB$  specifies an address which happens to be in the program store. This instruction is located at the address  $AA$ . At time 1 the buffer order word register contains the instruction at address  $AA$ , the order word register contains the instruction at address  $AA - 1$  and the program store address register contains the number  $AA + 1$  in preparation for the reading of the next instruction. At time 2 the buffer order word register has the instruction at address  $AA + 1$  but it cannot execute this instruction because the data word called for by the previous instruction has not yet been fetched. The order word register, in the

\* Blocks of memory addresses assigned to program stores and call stores are fixed for all No. 1 ESS installations; the wiring pattern of the memory address decoder is therefore the same in all installations.

†  $BB$  is a symbolic representation of some constant;  $AA$  represents a program address.

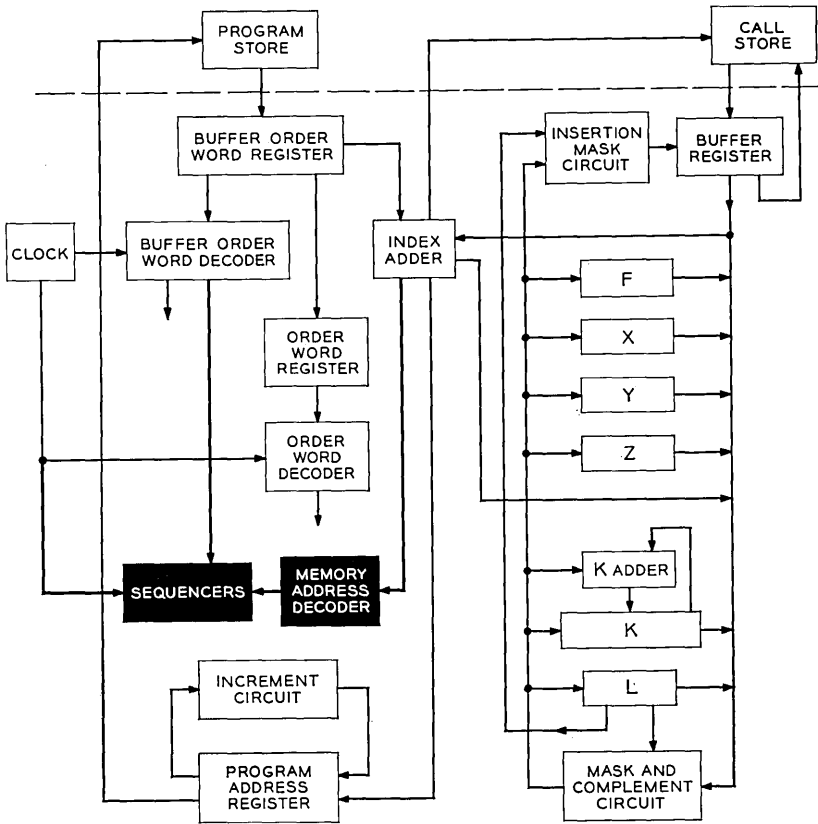


Fig. 14 — Derivation of the central control (8).

meantime, has the instruction at address  $AA$ , while the program store address register has received a data address from the index adder. This data address  $Y + BB$  is now being used to read the program store. At time 3 the buffer order word register contains the information located at address  $Y + BB$ , and the order word register continues to hold the instruction at address  $AA$ , while the program store address register has now been incremented to the value  $AA + 1$  to prepare for the reading of the next instruction. At time slot 4 this instruction has been read into the buffer order word register, the output of the buffer order word register has gone via the index adder to the appropriate destination (the  $X$  register) under the control of the order word register, and the program address register is preparing to read the instruction at address

<u>TIME SLOT</u>	<u>BUFFER ORDER WORD REGISTER</u>	<u>ORDER WORD REGISTER</u>	<u>PROGRAM ADDRESS REGISTER</u>
1	(AA)	(AA - 1)	AA + 1
2*	[AA + 1]	(AA)	BB + Y
3*	(BB + Y)	(AA)	AA + 1
4	(AA + 1)	(AA)	AA + 2
5	(AA + 2)	(AA + 1)	AA + 3

(—) → SYMBOL MEANING WORD STORED AT THIS ADDRESS OR IN THIS REGISTER.

INSTRUCTION AT ADDRESS AA IS MX, BB, Y;  
BB + Y IS AN ADDRESS OF DATA IN THE PROGRAM STORE.

\*THESE ACTIONS ARE CONTROLLED BY CENTRAL CONTROL INTERNAL SEQUENCE CIRCUITS SINCE THE OPERATION COVERS MORE THAN ONE CENTRAL CONTROL CYCLE TIME.

Fig. 15 — Time sequence of words passing through BOWR, OWR, and PAR when reading data from program store.

AA + 2. At time 5 this instruction is in the buffer order word register and the order word register has the instruction located at address AA + 1, while the program store address register has been incremented to AA + 3. Since the order word decoder is strictly a combinational circuit, the sequencer must be used to control the actions of fetching the data; otherwise the order word register would simply control the execution of the instruction at address AA three consecutive times. Note that the instruction for reading data from the program store consumes three cycles: one basic cycle, one cycle to read the data from the program store, and one cycle to reread the next instruction.

### 3.13 Conditional Transfers

A very important part of any data processing machine instruction repertoire is the set of conditional transfer instructions. These instructions cause a transfer of program control to a specified address if some data word or bit of data appearing in central control is some predetermined value. If the word or bit does not have that value, the transfer is not made, and the instruction immediately following the transfer instruction is executed.

Eight transfer instructions are provided to interrogate the contents of the accumulator for the following values: positive, negative, arithmetic zero,\* all but arithmetic zero, logical zero (+0 only), all but logical zero, less than or equal to arithmetic zero, and greater than or equal to arithmetic zero.

\* Arithmetic zero includes +0 (all zeros) and -0 (all ones). In both cases all 23 bits are alike, and arithmetic zero is therefore also referred to as "homogeneity."

A pair of control (C) flip-flops connected to the masked bus (see Fig. 16) store the homogeneity and sign of data words read from memory as the words appear on the masked bus. Another eight transfer orders test the C flip-flops for the same combinations of values available for testing the accumulator register.

Normally, the conditional transfer instructions follow immediately after the condition is registered, either in the accumulator or the C flip-flops. The usual instructions for gating information into registers may set the accumulator or the C flip-flops. In addition, there is a set of compare instructions (see Fig. 17) which do not alter any register but

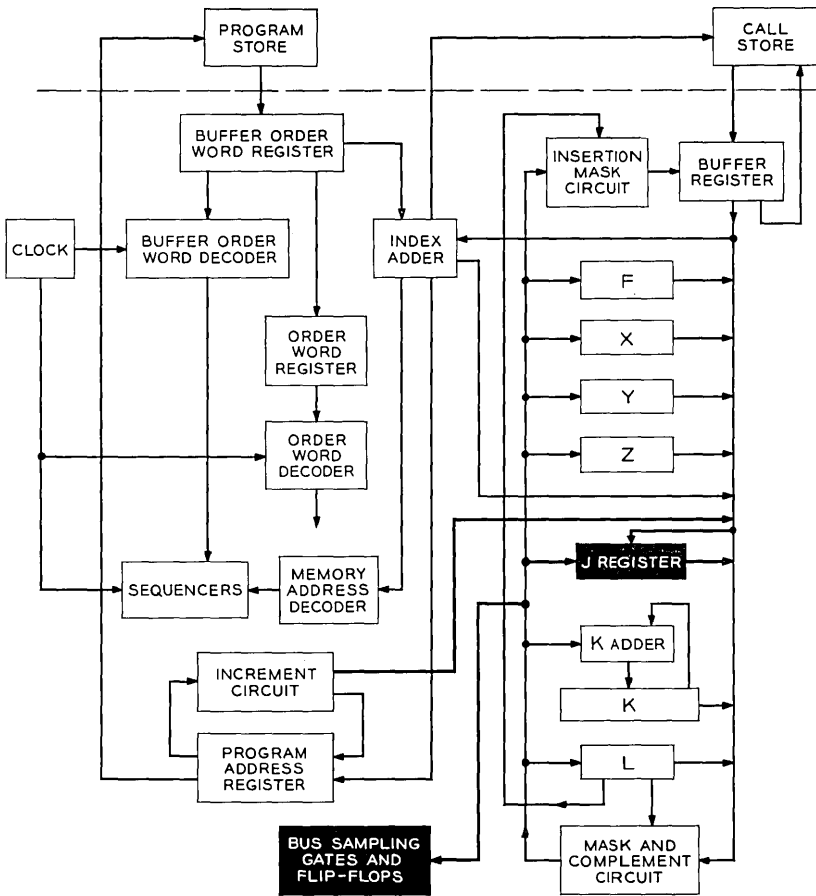


Fig. 16 — Derivation of central control (9).

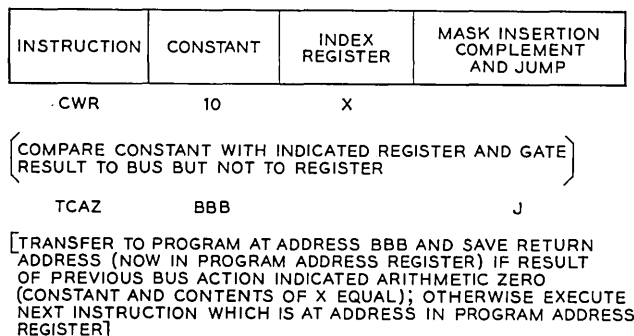


Fig. 17 — Basic instruction format 4.

which set the C flip-flops according to the result of the comparison. For example, the instruction CWR,10,X compares 10 with X and places the result in the C flip-flops. The comparison is performed by subtracting 10 from X, placing the result on the bus, and gating it only to the C flip-flops. One way to check whether the X register was actually equal to 10 is to follow the first instruction with an instruction TCAZ, BBB: transfer to the address BBB if the C flip-flops are equal to arithmetic 0; if not, advance to the next instruction.

### 3.14 *J Option and Register*

Associated with transfer instructions is a return address (jump) option (see Fig. 17). Unconditional or conditional transfer instructions occur frequently in the middle of a program and are used to transfer to a subroutine to do a common task; subsequently the subroutine returns control to the original program. Since the subroutine must know where to return, a J register (see Fig. 16) has been provided which may be set up at the discretion of the programmer whenever a transfer is executed. If the transfer is actually executed, the J register is set to the address of the instruction immediately following the transfer instruction. To set up the J register, a path must be provided from the output of the increment circuit to the unmasked bus and thence to the J register (see Fig. 16).

### 3.15 *Index Register Modification Options*

Index register modification options are available in the No. 1 ESS order structure. If some task is being performed on a number of successive memory locations, it is sometimes convenient to set an index



register to the value of the first memory address, then to increment the register by +1 as successive words are read from memory. This incrementing can be performed as an option on most reading instructions. For example, the instruction  $MX,3,Y$  simply gates the contents of memory at the address  $Y + 3$  into the X register. The instruction  $MX,3,YA$  (see Fig. 18) gates the contents of memory at the address  $Y + 3$  into the X register and increments Y by 1. The incrementing is performed by connecting the increment circuit input to the unmasked bus (see Fig. 19); this allows the index register to be gated into the increment circuit; the output of the increment circuit may later be gated to the index register via an output connection to the masked bus.

Two other index register modification options exist which change the indicated index register to the indexed quantity. Thus, the instruction  $MX,3,YW$  reads the memory at address  $Y + 3$  and gates this into the X register and also changes Y to the value W, the indexed quantity, which is  $Y + 3$ . Another index register modification option is the setup index register modification. For example,  $MX,30000,SY$  would read the contents of memory at address 30,000 into the X register and would place the quantity 30,000 into the Y register ( $SY = \text{set up } Y$ ).

### 3.16 Logic Register Setup Options

The logic register is changed very frequently in the course of a typical program. Furthermore, many of the programs that are encountered include effectively indirectly addressed readings,\* i.e., one instruction is used to read a quantity, and the reading is then used as the address of the quantity which is actually desired. In such a case, the register being used for the second reading usually contains the complete address so that no additional data are required as the constant of the instruction. Consequently, it is desirable to be able to use the constant of the second instruction to set up the logic register instead of performing unnecessary indexing. A direct path has therefore been provided between the buffer order word register and the logic register (see Fig. 19). If setup masking is specified, then the constant of the instruction is used to set up the logic register and is not used in indexing.

The interplay of index register modification and setup masking options can be illustrated by the following programming problem: complement the 3 least significant bits of memory at the address  $Y + 3$ .

---

\* Indirect addressing was not considered worthwhile for data operations, because it never saves time and rarely saves instructions in this system.



Using straightforward programming, this can be done by the following three-step program: MX,3,Y (read memory into the X register); WL,7 (set up the logic register to the value seven, i.e., 1's in the 3 least significant bits, 0's elsewhere); XM,3,Y,ELC (store X in memory using insertion masking and complementing at the address  $Y + 3$ ). This program will result in 3 least significant bits of X — i.e., the bits selected by the logic register — being complemented (C option) and inserted (EL option) into, first the buffer register, thence the memory, at the location  $Y + 3$ . A two-step program for performing the same task is the following: MX,3,YW (read from memory at address  $Y + 3$  into the X register and set up Y to the value  $Y + 3$ ); XM,7,Y,ESC (set up the logic register to the quantity 7 as indicated by the S in the ESC mask option; complement, as indicated by C, the X register, and insert, as indicated by E, into the buffer register the 3 least significant bits as selected by the L register; and gate this information to memory at the address which is now in Y, and which is 3 greater than the original value of Y). Note that the first program repeated the constant 3 twice, whereas the second program used it only once. The index register modification option permitted the constant of the instruction to be remembered for subsequent instructions without using any extra steps. Similarly, the constant 7 for setting up the mask was useful in the second instruction of the modified program because, since no constant was necessary for addressing the memory, a constant could be used for setting up the logic register.

It is important to remember that these options not only conserve memory space for instructions but save the time necessary to execute additional instructions. In this system, each instruction takes one cycle whether it be a memory instruction (M) or a register setup instruction (W).

### 3.17 *Rightmost One Function*

One function that occurs frequently in the type of data processing work that constitutes the call processing program of the No. 1 ESS is that of detecting and identifying a one in the midst of a group of zeros in a word. The one might signify a request for action, the zeros, inactivity; the position of the one would represent which member of the group requires the action. By concentrating always on the least significant one, successively all requests will eventually be handled.

It is important to have some instruction which identifies the position within a word of the rightmost one, because this operation is performed often and is awkward to do using more conventional instructions. The

word to be examined is placed in the accumulator; a rightmost one detect circuit connected to the accumulator gates the binary position of the rightmost one onto the unmasked bus. This information is then transmitted via the buses to the F (for "first-one") register. A circuit to reset this bit in the accumulator then receives its selection information from the F register.

Two instructions exist for the first-one function: TZRFZ transfers if the accumulator is zero; otherwise, it gates the position of the rightmost one to the F register and resets that bit; TZRFU performs the same actions, except that the bit in K is not reset. The programmer specifies the transfer address information in the same way that it is specified for any conditional transfer instruction.

### 3.18 *Buffer Bus Registers*

In addition to the index registers described above, a number of other flip-flops in the central control are under the control of a programmer. They include the bulk of the maintenance control and match flip-flops. These flip-flops are in groups and are set up and read by buses connected to the B register (see Fig. 20). The flip-flop groups are then examined and controlled as if they were word locations in memory. Each flip-flop group is assigned a distinct address; when this address is generated, the memory address decoder operates the gates to or from this flip-flop group. Thus the control of registers does not consume instruction code space.

### 3.19 *Interrupt Facilities*

The No. 1 ESS central control has interrupt facilities. These facilities permit a signal to come in at any time and:

- (1) cause the program currently being executed to be interrupted,
- (2) permit the state of central control to be stored in memory,
- (3) allow an interrupt program to be executed,
- (4) allow the state of the central control to be restored to its pre-interrupt state, and
- (5) allow the interrupted program to be resumed.

In effect, the programmer need not be concerned about the possibility of an interrupt occurring at any time, since the interrupt will not interfere with the execution of his program.

The interrupt is used for two purposes: first, dial pulse scanning and similar functions which must be performed every five or ten milliseconds are carried out by an interrupt program, triggered by a five-millisecond



Sometimes, this is unavoidable; for example, the program that detects an incoming trunk service request (an interrupt-level program) and the program that seizes an outgoing trunk on a terminating call (a base-level program) both write into the same busy-idle bits if the trunk is a two-way trunk used for both incoming and outgoing calls. Interaction problems may occur if a bit is being inserted in memory and the interrupt occurs between the reading and writing steps that constitute an insertion into memory. Two instructions have been created to solve this problem: MCII and MKII. These instructions are the normal MC (memory to the buffer and also the C flip-flops) and MK, except that all but the maintenance interrupts are barred until the immediately following instruction has been executed. By using one of these instructions as the first step of an insertion, a programmer guarantees that no interfering interrupt will occur while he is inserting the desired information.

### 3.20 *Mixed Indexing*

Sometimes, a couplet of instructions such as

$$\text{MX},0,\text{Y}$$

$$\text{MZ},0,\text{X}$$

occurs. The second of these instructions uses the value of X that was set up by the first instruction. Since the call store reading of the first instruction comes back at the same time that indexing is performed for the second instruction, a timing problem exists. This is handled by recognizing such situations (mixed indexing) and gating the data on the masked bus, i.e., the data going into the X register, to the index adder (see Fig. 20), instead of gating data from the unmasked bus, i.e., the present contents of the X register. The circuit for recognizing this situation must examine the buffer order word register and the order word register to check for this condition. Note that if an interrupt takes place between these two instructions, then the X register will have been set up to the new value, and no mixed indexing takes place; in effect, the second member of the couplet has been preceded by a vacant or no-op instruction.

### 3.21 *Early Transfer Instructions*

For certain highly repetitive programs, especially those involving scanning, it is desirable to have a conditional transfer instruction which

will consume additional cycles only if the transfer is made. This is accomplished by coupling a normal read or write instruction with an indication that a transfer to a preset address is to be made if the C flip-flops or the K register records a particular state. The instruction is called an "early transfer" instruction because, if the transfer is made, the reading or writing action is inhibited; the transfer decision must be made sufficiently early so that the unwanted action is prevented.

Two of the early transfer instructions are TCMMF and TAULM. TCMMF will transfer to the address (previously set up) stored in the J register if the C sign flip-flop shows a 1 (or minus); otherwise, a normal MF (memory to the F register) instruction is executed. All options normally available for an MF instruction can be specified, since the TCMMF operation itself completely specifies the conditional transfer instruction and does so without permitting any options or any choice of the source of the transfer address. TAULM will transfer to the address (previously set up) in the Z register if the C flip-flops show a nonzero quantity; otherwise, a normal LM (logic register to memory) instruction is executed. Again, all options normally available for an LM instruction can be specified; the TAULM operation completely specifies the conditional transfer instruction and transfer address source.

The advantage of the early transfer instructions is that the transfer address need be set up only once for a large number of loops of a sub-routine, or that the transfer address may have been previously set up in the course of executing another part of the program; if no transfer takes place, no cycles have been wasted on making the decision.

### 3.22 *Logical Combinations of Registers with Memory or Data*

Instructions are available which permit indexed data (W) or the contents of memory found at an indexed location (M) to be logically combined with the contents of the X, Y, or Z register. This is accomplished by first gating the X, Y, or Z register to the L register via the buses, then using the mask and complement circuit to logically combine L with either M or W and gate the result back into X, Y, or Z. AND, OR, AND complemented, and OR complemented are the logical expressions that may be specified; naturally, this means that the mask circuit must be able to OR as well as AND. Since these instructions use the L register, no masking may be specified.

## IV. PERIPHERAL SYSTEM FACILITIES IN CENTRAL CONTROL

In addition to communicating with the stores, the central control also communicates with the peripheral system.<sup>12</sup> This system contains three

main bus systems (Fig. 21). The first of these is used in communicating with the central pulse distributor.<sup>3</sup> This central pulse distributor either operates flip-flops which drive relays directly or is used for selecting the particular unit which is to be addressed via the second bus system, the peripheral address bus. Responses from peripheral units come via the scanner answer bus.

Peripheral actions are not generally performed in the middle of a complicated data processing problem. Therefore, some of the general purpose index registers of central control are used for driving these buses (Fig. 22). The F register is used to drive the translator which controls the central pulse distributor. The logic register is used to receive answers from the scanner response bus. This makes it easy to combine scanner answers with memory information, since the memory information can be read into the buffer register and can be combined logically with the contents of the logic register in the mask and complement circuit.

Ideally, it would be most reasonable to drive the peripheral address bus from the data buffer register. However, the timing of the waves of information leaving central control to address the peripheral system is such that the buffer register would not be available for a reading on the subsequent cycle; peripheral actions start after a considerable amount of preliminary processing, frequently including a call store reading. For this reason the contents of the buffer register are transmitted to the accumulator addend register, which is known to be available at this time. The accumulator addend register is used to drive the translators connected to the peripheral addressing bus.

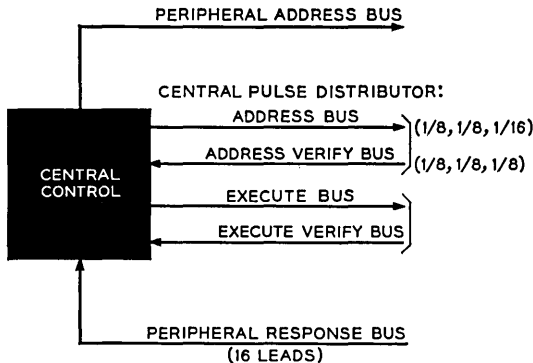


Fig. 21 — Block diagram of central control communication with peripheral equipment.



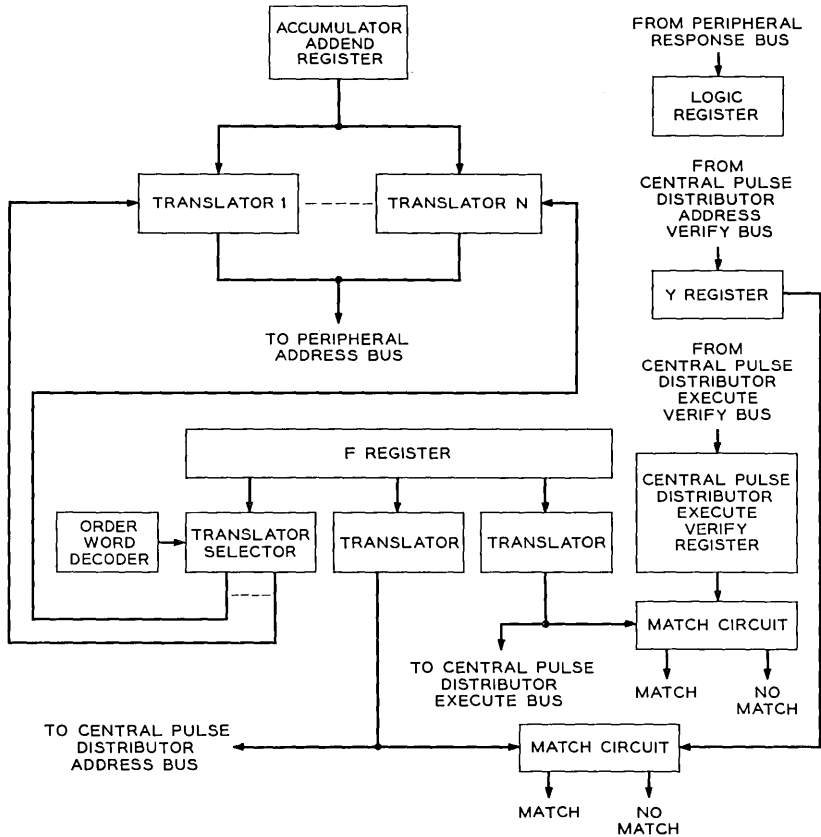


Fig. 22 — Detached internal central control blocks for communication with peripheral equipment.

A large majority of the units driven by the peripheral system are network units,<sup>4</sup> signal distributors,<sup>3</sup> and scanners.<sup>3</sup> For economic design of the many peripheral controllers, a coded address is used for controlling these units. For example, a 1024-point scanner requiring a 1-out-of-64 row selection is addressed by two 1-out-of-8 signals, not by 6 binary signals. Furthermore, the address code for each of the various network frames and the signal distributors is different. Central control has built within it translators to convert the binary information, convenient for data processing, to the particular code used for controlling a specific unit.

Since many peripheral instructions are sent out from a section of call store called the "peripheral order buffer," it is desirable that the actual

peripheral instructions be independent of which particular type of frame is being addressed. Since the enable address for a particular frame is always required along with the peripheral instruction, it is convenient to store the type of frame along with this enable address so that the proper translation of the binary addressing information will be made. A peripheral action is therefore preceded by setting up the F register to the enable address necessary for selecting the particular unit and a code which will select the proper translation option of the binary information to be sent to the peripheral bus. The actual peripheral instruction is then set out via an instruction, such as MA, which assumes that the F register has been preset. MA first sends out an enable signal, then takes the contents of memory and gates them via the buffer register and accumulator addend register to the peripheral address bus, translating the binary information according to the code stored in the F register.

When a peripheral (scanner) response is to be returned to central control, it is sent to the logic register. Therefore the instructions for controlling scanners are different from the instructions for simply sending an instruction to a peripheral unit which does not give a response. The basic general-purpose peripheral instructions are therefore MA and MAS (MAS being used when a response is expected) and the counterparts, WA and WAS. The MAS instruction and WAS instruction also reset the logic register and open gates to this register from the peripheral response bus.

If a central pulse distributor is addressed only for the purpose of operating or resetting a flip-flop, a special instruction, MD or WD, is used. With these instructions it is only necessary to set up the F register, since the peripheral addressing bus is not required.

For addressing the scanner, only 6 bits of information are necessary to select a row. There is enough information in a 23-bit word to set up an enable address and to actually address the scanner. The MSF and WSF instructions, by simultaneously setting up the F register and the accumulator addend and then performing a peripheral operation, accomplish this in one setup cycle. (The actual execution of the instruction overlaps into the next cycle so that two scanner readings cannot be made in consecutive cycles.)

For purposes of verifying peripheral operations, a verify response signal comes back from the central pulse distributor. This response is gated into the Y register and is then matched against the output of the translator that is connected to the F register. (No response is returned when executing MD or WD instructions.)

For driving such units as the tape unit,<sup>5</sup> which will accept straight binary information, and for sending test signals to peripheral units, it is also possible to bypass the translator, simply letting the outputs of the accumulator addend register go directly to the addressing bus. Since the peripheral addressing bus is 36 bits wide, it is possible, if testing a unit which requires more than 23 input leads, to take 13 bits from the accumulator as well as 23 bits from the accumulator addend.

It is important to remember that in the case of network frames the peripheral addressing bus contains instructions as well as addressing data.<sup>4</sup>

Another check made on peripheral operations is that the proper central pulse distributor has been selected. When a central pulse distributor is selected, it sends back an echo signal which goes to a flip-flop register on the buffer bus. This is matched against the output of a translator attached to the F register.

#### V. SUMMARY OF ESS ORDER STRUCTURE WITH OPTIONS

This section is a summary of No. 1 ESS instructions with their available options, as shown in Table I. As has been previously indicated, every instruction has three main modifiers: the data field; the RM field, which includes index register modification and indirect addressing for transfers; and the LCJ field, which includes masking options, complementing, and the setting up of the J register on transfers. In addition, although this is not specified in the actual writing of each instruction, many instructions set up the C flip-flops, which can then be examined on a subsequent TC conditional transfer instruction.

The constant in the data field may serve one of three purposes: it may be directly used data that is part of the instruction; it may be part of the address used for finding such data; or it may be the mask that is to be used in the instruction. In Table I, the symbol S in the data or address section indicates that data or an address may be specified unless an S occurs in the LCJ field, in which case no data or address constant may be specified, since the constant of the instruction must be used for setting up the mask.

The R subfield usually contains the identity of the indexing register; in a few instructions, the R subfield specifies a register indicated by the instruction. The latter include the instructions for adding the contents of two registers (since only one of the registers can be specified as part of the instruction, the other register must be specified as an option); the CWR instruction, which is used to compare a register with a constant in the order; and the TR family of instructions which sense the

TABLE I—SUMMARY OF BASIC ORDERS AND AVAILABLE OPTIONS

Symbolic Order Fields .....	DA		RM					LCJ										
	D	A	R*				M	L†			CJ							
			Data	Address	Identity of R in OP Code	Indexing Register		A (R + 1 → R)	W (DA + R → R)	S (DA → R)	Indirect Addressing	PL Masking	EL Masking	PS Masking	ES Masking	C Complementing	J (RA → J)	
General-purpose operation codes																		
WK, AWK, SWK, PWK, UWK, XWK	S			✓	✓					✓		✓		✓				
WF, WJ, WX, WY, WZ, CWK, CWKU	S			✓	✓					✓		✓		✓				✓
WL, PWX, PWY, PWZ, UWX, UWY, UWZ	✓			✓	✓									✓				✓
WB	S			✓	✓					✓	✓	✓	✓	✓	✓			✓
H, HC, Q, QC	✓			✓	✓													
MC, MCH		S		✓	✓	S	S			✓		✓		✓				✓
CWR	S		✓							✓		✓						✓
MB		S		✓	✓	S	S					✓						
BM		S		✓	✓	S	S						✓					
ABR, AFR, AJR, AKR, ALR, AXR, AYR, AZR	L		✓							✓		✓		✓				✓
SBR, SFR, SJR, SKR, SLR, SXR, SYR, SZR	L		✓							✓		✓		✓				✓
FM, JM, KM, XM, YM, ZM,		S		✓	✓	S	S			✓	✓	✓	✓	✓	✓			
MK, AMK, SMK, PMK, UMK, XMK, MKII		S		✓	✓	S	S			✓		✓		✓				
MF, MJ, MX, MY, MZ, CMK		S		✓	✓	S	S			✓		✓		✓				✓
LM		S		✓	✓	S	S				✓	✓	✓	✓	✓			
ML, PMX, PMY, PMZ, UMX, UMY, UMZ		✓		✓	✓	✓	✓							✓				✓

Set C Flip-Flops

TABLE I — Continued

Symbolic Order Fields.....	DA		RM					LCJ							
Subfields .....	D	A	R*			M	L†				CJ				
General-purpose operation codes	Data	Address	Identity of R in OP Code	Indexing Register	A (R + 1 → R)	W (DA + R → R)	S (DA → R)	Indirect Addressing	PL Masking	EL Masking	PS Marking	ES Masking	C Complementing	J (RA → J)	Set C Flip-Flops
T, TK... TC...		✓		✓	C	C	C	✓						✓	
TR...		✓	✓		C		C	✓	✓				✓		
Input-output operation codes															
MA, MAS, MSF, MD		S		✓	✓	S	S		✓		✓		✓		
WA, WAS, WSF, WD		S		✓	✓				✓		✓		✓		
Combined operation codes															
TZRFU, TZRFZ		✓		✓	✓	✓		✓						✓	
TAULM, TCMMF		+ S		+ ✓	+ ✓	+ S	+ S		+ ✓		+ ✓		+ ✓		

\* Of options A, S, and W, only one may occur in any one instruction.  
 † Not more than one of the four L options can be used in an instruction.

Key to symbols:

- ✓ — indicated use of field is available
- S — ✓ unless S appears in L subfield
- C — conditional and late: occurs only if transfer occurs and after register is used (or, in the case of TR... orders, after the register is tested)
- L — ✓ only if S appears in L subfield
- + — action of options occurs only if transfer does not occur.

contents of some register and transfer accordingly. Three index register modifications are available, of which only one, the A option, is available if setup masking is used. (This is a direct result of the fact that only one constant may be specified in any instruction; if this constant is used to set up the mask, then it cannot be used to modify an index register.)

Some general observations may be made concerning the types of options available with various instructions. W instructions do not have W or S index register modification options. While these index modification options are meaningful, they would have a relatively low utility and require a great deal of code space. The W and S options are much more useful on memory instructions for setting up a register to a full

address, so that on a subsequent instruction, the constant in the data field will be available for masking.

Certain instructions, such as PWX, UMY, WL, ML, etc., do not permit setup masking, since they either set up the mask directly as part of the instruction (WL), or the logic register is used in carrying out the instructions (UMX, etc.).

On MB instructions, no masking or complementing is possible, since the contents of the buffer register are fed directly by the memory and do not pass over the bus. The PS option is included and allows a programmer to set up the logic register for a subsequent instruction even though he does not use it in this instruction.

Insertion masking is used only on WB instructions and on the instructions which write the contents of some register to memory, since the insertion mask can only be associated with the B register.

The C flip-flops in general are set up on all compares and on all orders which gate W or M to some register other than K.

As can be seen, the rules for checking on which options are allowed on any particular instruction are not too simple. Therefore, even though they are summarized in Table I, the No. 1 ESS program compiler<sup>13</sup> checks for violations of the allowed options. There are, of course, a number of other restrictions that the compiler can check for. The chief restriction within No. 1 ESS is the fact that the accumulator may not be used as the indexing register on the instruction immediately following AMK, PMK, UMK, XMK, or SMK instructions. Following all peripheral instructions, there are a number of very complicated restrictions; in general, the Y and F registers cannot be altered on the next instruction, and the L register cannot be changed following MAS, WAS, MSF, and WSF.

## VI. ENCODING OF THE ORDER STRUCTURE

Each instruction or program order word obtained by central control contains 37 information bits designated 36, 35,  $\dots$ , 0. Each such order consists of an order field and a data-address field. The order field includes an order selection subfield, an index register subfield, and order option subfields. When the order includes a data word, the data-address field contains the data word in bits 22 through 0, and bits 36 through 23 compose the order field. When the order word contains an address in the data-address field, the address occupies only bits 20 through 0 of the word, and a larger order field appears in bits 36 through 21 of the program order word.

The 21-bit address field permits full access by memory reading and

writing orders to all locations in the program stores and call stores, as well as to a group of flip-flop registers in central control. When an order option subfield indicates the setting up of the logic register (e.g., PS or ES masking), the data-address field contains the 23-bit word of data; in such instances, the order field is restricted to bits 36 through 23, even when the order selection subfield indicates a memory reading or writing order.

The encoding of the instruction repertoire or order structure represents a compromise between: (1) attempting to provide an order structure with the maximum flexibility that can be represented by the available binary combinations and (2) decomposing the combinations of order selection, index register selection, and option selection into simple subfields. Counting all meaningful combinations of order selection, index register selection, and nonconflicting order option values, the encoding provides over 12,000 distinct combinations in the 14- and 16-bit order word.

The index register subfield is always encoded in bit positions 34 through 32. Bit position 35 is reserved for the complement option for all orders except regular transfer orders; bit position 35 serves as a J option subfield for transfer orders. A complete decomposition is not possible for index register modification options or masking options without an excessive waste of code space. However, the encoding includes a grouping of the classes of related orders, this grouping being represented by relatively simple bit combinations; within each grouping the index register modification and masking options are each grouped into one-, two-, or three-bit subfields according to the number of meaningful and useful combinations.

The encoding is shown as the Karnaugh maps in Figs. 23 through 26. These maps represent the four binary combinations of bits 31 and 30; this division of the encoding is representative of the grouping of several large classes of orders. For example, the binary combination 31-30 = 01 is assigned exclusively to orders reading memory; all such orders are encoded within this combination and its corresponding map in Fig. 23. The combination 31-30 = 11 is assigned to W orders in which the destination register is given explicitly by the mnemonic code and not by the index register field ( $R_D \neq R_I$ ); the encoding of this class is shown in Fig. 24. Related orders, such as MA and WA, occupy corresponding positions within the two maps. This correspondence provides for economy in designing the gating functions which carry out the same steps for related orders. For example, a single destination register selector can be activated by either a memory reading order or its corresponding W order.

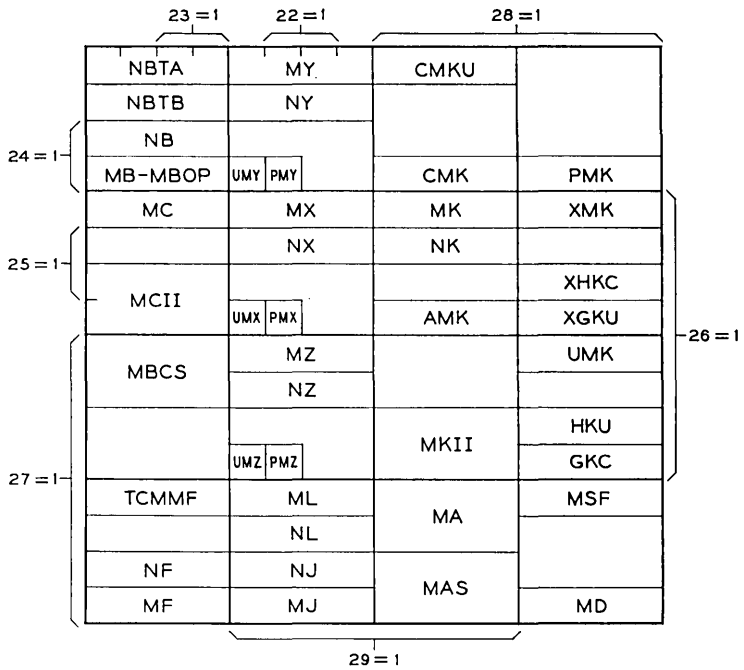


Fig. 23 — Memory reading orders (31-30 = 01).

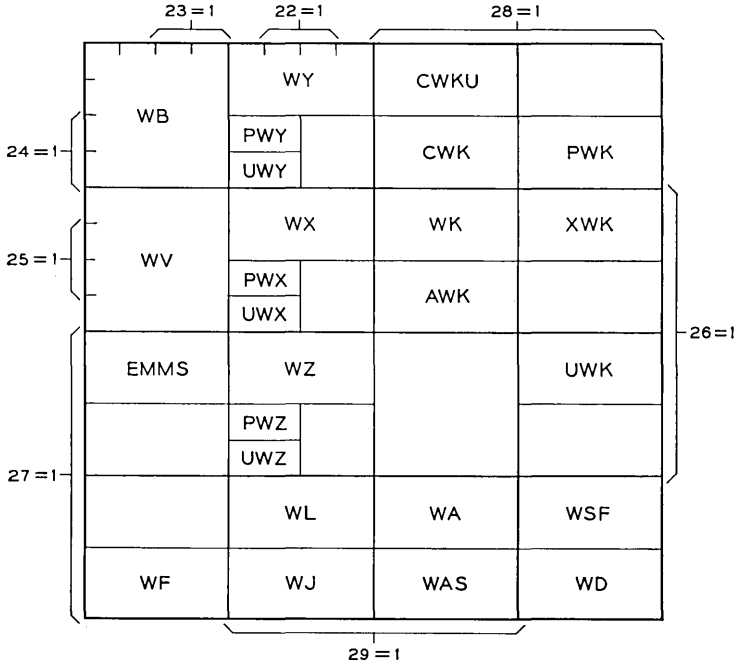


Fig. 24 — Word orders ( $R_D \neq R_I$ ); (31-30 = 11).



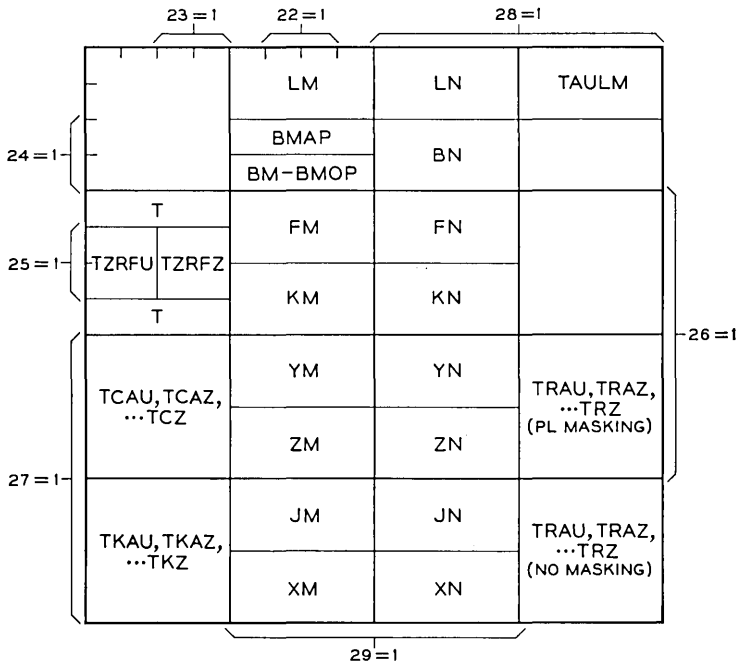


Fig. 25 — Writing orders and regular transfer orders ( $R_D = R_I = 00$ ).

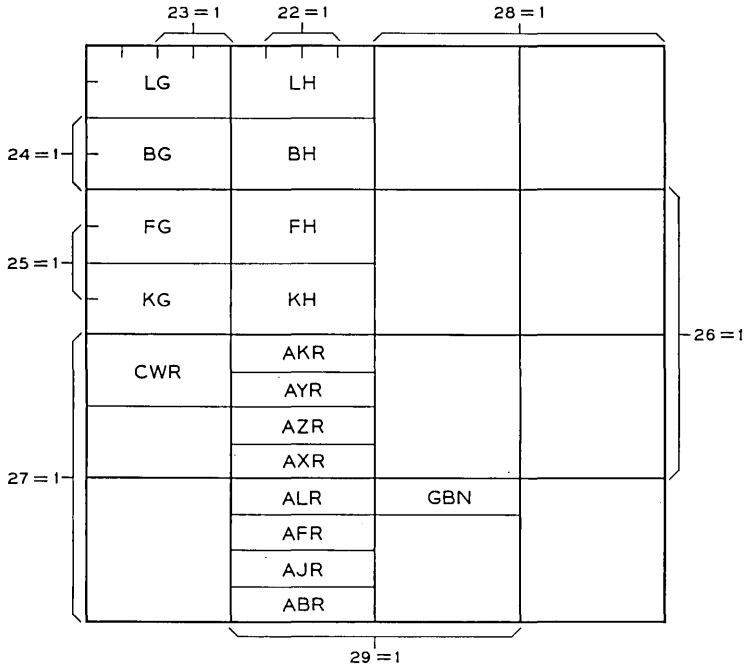


Fig. 26 — Maintenance writing orders and word orders ( $R_D = R_I$ ); ( $R_D = R_I = 10$ ).

The remaining classes of orders require less coding space than the two classes just described and consequently occupy only portions of the remaining maps in Figs. 25 and 26. The combination 31-30 = 00 includes all regular transfer orders, all but a special class of memory writing orders, and a small class of miscellaneous orders including rotate and shift orders. Fig. 26 represents the binary combination 31-30 = 10 and includes the class of maintenance writing orders and the word orders ( $R_D = R_I$ ). Figs. 23 through 26 include special classes or orders described below. The maintenance writing orders include control mode memory orders (represented by the letter N in the mnemonic equivalents), G-mode memory orders, and H-mode memory orders (see Section 7.7).

The early transfer orders are encoded according to the data processing actions to be taken whenever the decision is made not to transfer. Thus the order TCMMF is encoded as a memory reading order (31-30 = 01), and the order TAULM is encoded as a memory writing order.

With the encoding just shown, the decoders were designed to optimize whatever decompositions are made available for both the optional data processing gates and in the selectors and gates common to related orders. In addition to the buffer order word decoder and the order word decoder, two classes of data processing functions must be included to complete the above summary. The memory address decoder controls the generation, transmission, and central control response of each program store command and each call store command; the data processing for memory reading orders, memory writing orders, and indirect transfer orders includes the use of the memory address decoder. The decoders described here carry out those gating actions necessary to obtain and process single-cycle orders. Many classes of orders cannot be executed in a single machine cycle; the additional gating actions for such orders are handled by a group of sequencers; these actions may include automatic retrieval and/or correction of program order words and data words.

## VII. MAINTENANCE OBJECTIVES AND CIRCUITS

The No. 1 ESS must be able to provide continuing service to customer lines in the face of occasional and random occurrences of circuit troubles.<sup>14</sup> Duplication of subsystem units or portions of such units provides a set of potential replacement parts. Whenever a circuit trouble occurs within a subsystem, the detection of that trouble is followed by the required replacement; this replacement is made at electronic speeds to minimize the interfering effect of the trouble.

Each subsystem includes maintenance circuits which: (1) serve in detecting symptoms of circuit troubles as they appear and (2) aid in determining the location of such trouble to facilitate repairs

The detection of circuit troubles leads to the execution of maintenance programs which first determine whether a fault exists and if so whether the circuit trouble occurred within the active (controlling) switching system or within a standby duplicate unit. If a fault has occurred in an active subsystem unit, the next step is the necessary switching of associated active and standby units. The system is therefore quickly restored to an operable state and returned to the normal business of processing telephone calls; the subsystem unit in trouble is placed in an out-of-service state; and finally, special program sequences are interleaved with call processing programs to determine the faulty circuit element. The maintenance actions last described constitute a diagnosis of the out-of-service unit by the switching system; the results of this diagnosis appear as a printout on a special teletypewriter unit. Corresponding to each such printout is an entry in a specially prepared dictionary<sup>14</sup> which the maintenance man consults; the "definitions" in this dictionary are a listing of plug-in circuit packs to be replaced.

The entire procedure just described, from the detection of a circuit fault to the replacement of associated circuit packs, takes place in a matter of a few minutes; on completion of the repair of the out-of-service subsystem unit it is returned to the standby state for protection against future occurrences of circuit troubles. The maintenance circuits and associated program sequences serve in meeting a primary maintenance objective: essentially continuous telephone service with a minimum degradation in the quality of service in the presence of occasional circuit troubles.

Certain of the maintenance actions operate continuously and independently of the program sequences being executed in the central processor; other actions are obtained with the maintenance circuits and special program sequences. The integration of such program sequences and maintenance circuits is described elsewhere in this issue.<sup>14</sup>

### *7.1 Circuit Checks of Communication Channels between Central Control and Connecting Subsystem Units*

Communication of information between central control and the remaining subsystems comprises the transmission of commands and addresses to one or more such units; each command specifies the required circuit response, and each address specifies the location or locations

which are to respond to the command. These responses include setting (resetting) flip-flops, scanning a group of ferrods, reading or writing a 24-bit word of call store data, etc. Accordingly, the circuit responses in some instances include the return of information to the subsystem unit generating the command and address. Circuit check signals accompany redundantly encoded commands, addresses and responses transmitted between the central control and (1) the program stores, (2) the call stores, (3) the central pulse distributors, and (4) peripheral units such as scanners, network controllers, and signal distributors. Maintenance circuits in both central control and the connecting units provide a continual check of communication and serve as a safeguard against noise and circuit troubles in the communication channels. Since the rate of communication between central control and its connecting units is quite high, most occurrences of circuit troubles within (1) the circuit generating the redundancy in the commands, addresses, and responses; (2) the registers for transmitting and receiving these commands, addresses, and responses; and (3) the associated check circuits, will quickly result in the detection of a check failure. For example, the check of communication between central control and the program store is briefly outlined in Fig. 27. The decoder combines clock signals from the microsecond clock with dc inputs to generate synchronizing, command and mode signals. These are transmitted along with a 16-bit address and four-bit code to select the appropriate program store information block via cable drivers connecting a twisted-pair cable leading to the program stores. The selected program store responds with synchronizing and check signals and a 44-bit reading from the twistor memory which is returned to the buffer order word register of central control. As this response is being returned, the contents of the program address register are transmitted to an auxiliary storage register so that the next information block code and address may be gated to the program address register. This last gating action permits the simultaneous check of one program store response and the generation and transmission of a following program store command. Under control of timing signals derived from the decoders, the check circuits carry out single-error and double-error checks of the information contained in the auxiliary storage register (corresponding to the address of the word obtained)<sup>15</sup> and the buffer order word register. The check circuit also verifies that the program store response included a check (all-seems-well) signal from the responding program store. When one or more of these checks fail, signals on the corresponding check-fail conductors lead to the required remedial circuit action.

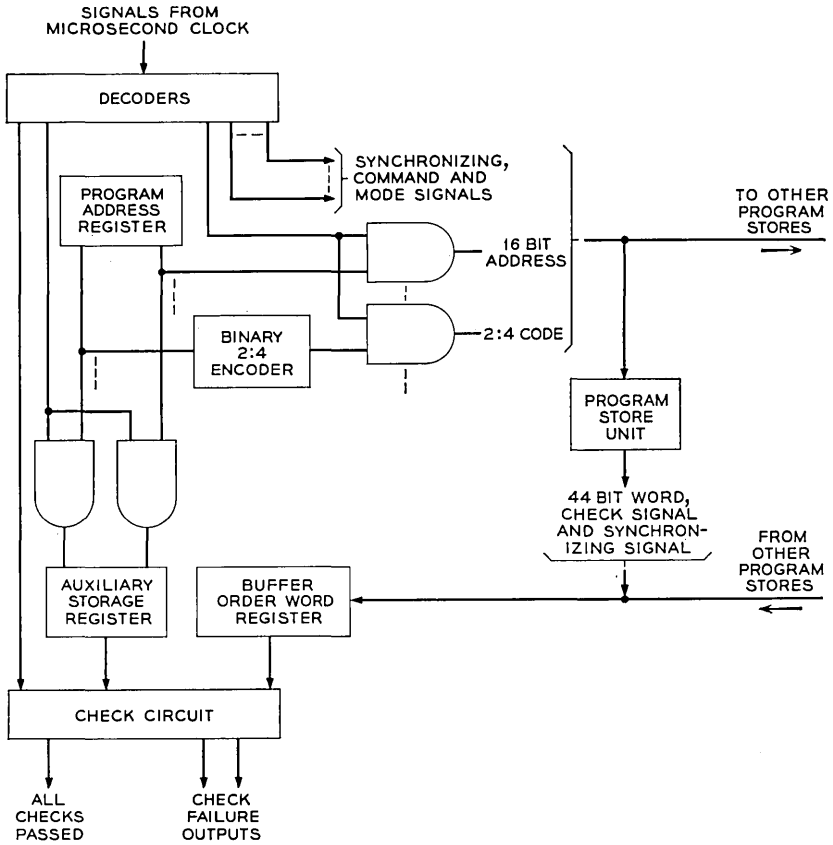


Fig. 27 — Check of central control-program store communications.

7.2 Interrupt Actions

A simplified block diagram of the interrupt system is shown in Fig. 28; it includes the three flip-flop registers in central control that are part of the buffer register bus system. This access permits single-cycle reading or writing access to these registers similar to that available to call store memory locations.

The interrupt source register comprises a number of interrupt source flip-flops; input signals to this register arrive from the millisecond clock and various check circuits within central control. The interrupt-level activity register serves to record the level interrupt corresponding to the program sequences being executed in central control. That is, cor-

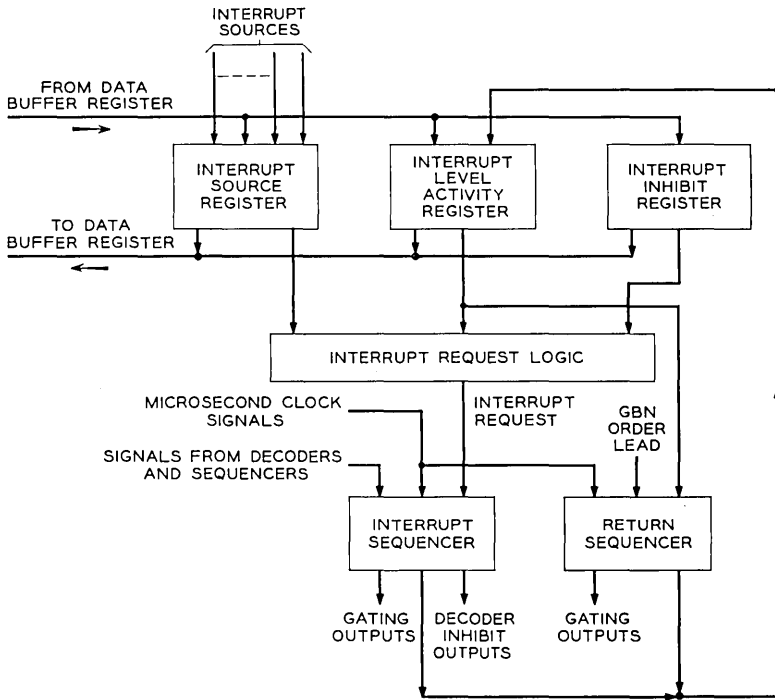


Fig. 28 — Interrupt system.

responding to each of the priority classes is a flip-flop within the interrupt-level activity register; whenever the interrupt system responds to an interrupt request, the wired transfer of program control is accompanied by the setting of the corresponding flip-flop in the interrupt-level activity register. The setting of flip-flops within the interrupt-level activity register also serves to inhibit the interrupt system from honoring interrupt requests from the level just served and all lower levels.

Assuming first that all flip-flops in the inhibit interrupt register of Fig. 28 are reset, the sequence of actions associated with each interrupt may now be described. When a given interrupt program sequence is required, a signal appearing on the corresponding interrupt source signal lead sets the corresponding interrupt source flip-flop. This signal propagates through the interrupt request logic and enables the interrupt request output conductor. The enabling of the interrupt request conductor is combined with clock signals and signals from the decoders and other sequencers to initiate the interrupt sequence. This assures that the interrupt allows any multicycle order or order following an

MCII or MKII instruction to go to completion before generating the wired transfer of program control. This consideration simplifies the hardware and program design for returning to interrupted program sequences.

Once activated, the interrupt sequencer carries out a number of functions extending over a period of several machine cycles; accordingly, the interrupt sequencer inhibits the order word decoder and buffer order word decoder outputs and generates independent gating signals to carry out a sequence of actions which include the following: (1) update the interrupt level activity register by setting the flip-flop in that register corresponding to the level interrupt currently being served (the interrupt request logic will then respond only to interrupt requests which may have a preassigned priority over the first interrupt program), (2) generate a transfer address corresponding to the entry point of the interrupt program sequence corresponding to the class of interrupt being served and gate this address to the program address register to effect this entry, (3) store the contents of the data buffer register in a first reserved location in the call store (this location depending upon the level of interrupt being served), and store the address of the instruction immediately following the last instruction executed prior to the interrupt.

Having completed these tasks, the interrupt sequencer returns to the inactive state, and the interrupt system is then responsive to further interrupt requests at higher levels. At this time, the entry to the corresponding interrupt program is made; this program begins the further storing of index registers, the logic register, etc., to complete the construction of the central control image in a set of reserved call store locations.

Upon completion of the required interrupt work functions, a program sequence restores the image of central control from the block of reserved call store locations. The interrupt program sequence then ends with a special return order (GBN), which activates another sequencer which completes the reconstruction of the central control image and transfers back to the interrupted program in an efficient three machine cycle sequence. This sequencer (also shown in Fig. 28) utilizes the interrupt-level activity register to complete the restoration of central control to the state occurring at the time of the interrupt. The sequencer must: (1) reinitialize the program address register to reenter the interrupted program at the proper point, (2) restore the data buffer register, and (3) reset the flip-flop in the interrupt-level activity register associated with the interrupt level from which the return is being made. Having completed these actions, the return sequencer advances to the

inactive state and is thereby made available for subsequent returns from other interrupt program sequences.

The inhibit interrupt register shown in Fig. 28, as its name implies, serves to selectively inhibit the response of the interrupt system to selected interrupt sources. The inhibit interrupt register is also a buffer bus register to which reading and writing access are provided. This register is used to selectively inhibit the interrupt sequencer response to interrupt signals during the execution of special test program sequences which, as part of their normal execution, cause the generation of interrupt source signals. The interrupt inhibit register also serves to inhibit interrupts due to repeated signals from defective subsystem units.

### 7.3 *Matching Circuits and Match Control Decoder*

In normal operation the duplicate central controls are executing identical orders within the same program sequences, and since the microsecond clock in both central control units is driven from one of two crystal oscillators,\* the individual data processing steps for each order are closely synchronized in the two central control units. Normally, the two central controls are started by placing the same entry address in the program address register to simultaneously obtain and execute the same first program order. Each central control then continues in step with the other; the same data are read from memory or the scanners, the same data processing steps are performed on this data, and the outcome of each decision order is identical. Furthermore, certain trouble signals are cross connected between central control units so that any additional cycles inserted in one central control unit for remedial actions are accompanied by the insertion of the same number of cycles in the other unit.

The mode of operation just described is designated as the "in-step mode" and is utilized with the matching circuits to provide a continuing hardware check of the operation of the two central controls. This check consists of repeated comparisons of like information processing points in both central control units to obtain rapid detection of trouble conditions within either unit. The repeated comparisons are made with the matching circuits under the control of the match control decoder. A simplified block diagram of these matching circuits, decoders, and cross-connecting buses is shown in Fig. 29.

Within each central control are two internal match buses which pro-

---

\* A flip-flop in each central control defines one unit as the active central control and the other unit as the standby central control. The crystal oscillator in the active central control drives the microsecond clocks in both units.



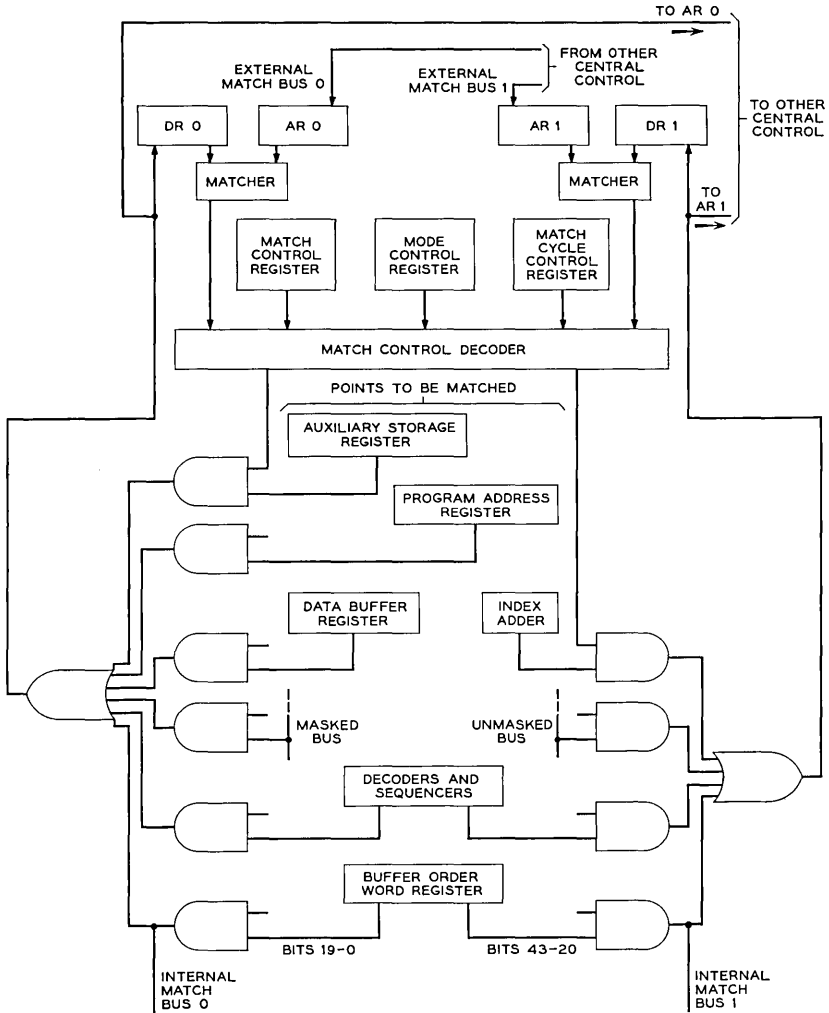


Fig. 29 — Central control matching circuits.

vide access to selected information processing points; these are labeled internal match bus 0 and internal match bus 1. Under control of the match control decoder, information from selected points is transmitted to these internal match buses, and from there other gates are enabled to place this information into internal match registers DR0 and DR1 and simultaneously transmit the sampled information via external match buses 0 and 1, respectively, to the other central control unit. The

match control decoders in both central controls are normally operating in step so that the information transmitted from the first central control to the second is stored in external match registers AR0 and AR1, respectively, in synchronization with the previously described gating actions. Two match circuits serve to compare the contents of AR0 with DR0 and AR1 with DR1; according to the state of the mode control register and the presence or absence of the match condition, the match control decoder generates the corresponding output signals. For example, when the matching circuits are employed in the routine matching mode, a selected sequence of common match points in each central control is matched at the rate of 2 matches per cycle; the detection of a mismatch condition generates a maintenance interrupt signal and further matching is automatically halted.

Since the matching circuits are limited to a maximum rate of two matches per machine cycle, the routine matching mode selects the specific sequences of internal points to be matched; the points to be matched depend upon the program and hardware actions being taken in central control to strategically examine those points most pertinent to the data processing steps that are occurring during a given machine cycle. Signals from the decoders and sequencers within central control are transmitted to the match cycle control register shown in Fig. 29. These signals set and reset specific flip-flops, which in turn direct the match control decoder and the selection of internal points for matching during each machine cycle.

The selection of points to be matched provides the detection of hardware troubles developing within central control as soon as the effect of that trouble would be communicated to other units in the switching system. It should be noted that the routine four-cycle match does not include the matching of all points to which the internal match buses have access. These additional points serve in other match modes described below.

#### 7.4 *Maintenance Matching Modes*

A number of maintenance matching modes provide program-controlled access to the array of register buses and test points connected to internal match bus 0 and internal match bus 1. During the performance of certain maintenance programs, the routine matching mode is inhibited, and instead one of a number of maintenance matching modes may be selected to use the matching circuits to monitor test points within the standby central control or use the matching circuits and

connecting buses to communicate selected data from certain of these points from one central control unit to the other.

The maintenance matching mode to be executed and optional gating actions ensuing the detection of mismatch (or match) conditions are selected by the information placed in the mode control register. Certain of the maintenance matching modes match selected points at selected time intervals; information placed in the match control register determines these selections.

The selection of the routine matching mode or one of the maintenance matching modes is made by writing the selected word into the mode control register and the match control register shown in Fig. 29. To switch from one matching mode to another, a special flip-flop in both central controls is reset by a central pulse distributor command; this inhibits the response of the match control decoder to the matching mode currently specified. This step is followed by updating the mode control register and match control register to the new matching mode desired; following these actions, the CPD-controlled flip-flop is again set and the match control decoder is responsive to the new mode. A description of each of these modes and its use appears in a companion paper.<sup>14</sup>

### 7.5 *Emergency-Action Sequencer*

The preceding sections describe a number of hardware checks and allude to both hardware and program remedial actions in response to circuit troubles detected in the central processor. These remedial actions include program sequences which are calculated to isolate circuit troubles to a particular unit and to control any required switching of units to obtain a working system. The execution of these sequences is in itself possible only if the active central processor includes an operable combination of the central control, certain program stores and the interconnecting bus system. That is, if the fault itself lies in one of these units, the central processor may be incapable of performing the necessary rearrangements. The emergency-action sequencer serves as a hardware checking and corrective means to handle this problem.

A block diagram of the interconnections between the emergency-action sequencer and its inputs and outputs is shown in Fig. 30. The inputs consist of four hardware checks; the failure of any one or more of these checks generates trouble signals to activate the emergency-action sequencer. These signals are dc-connected to the inputs of a monostable pulse circuit; the output of this monostable pulse circuit is connected to a series of monostable pulse amplifiers to provide a sequence of output

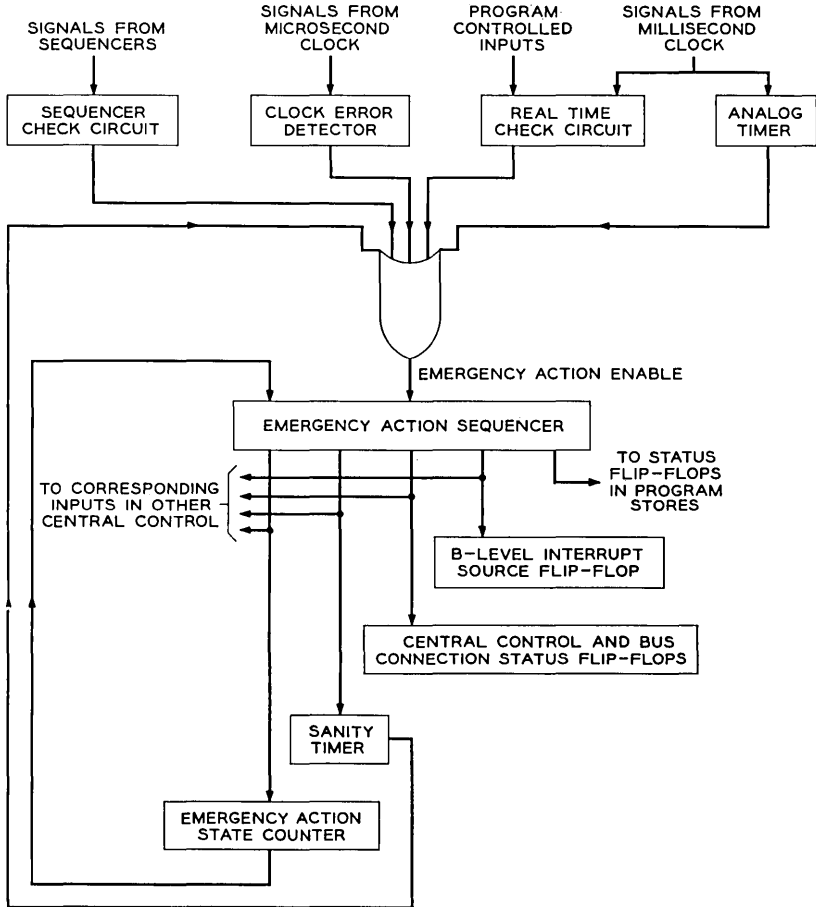


Fig. 30 — Central control emergency-action system.

pulses. The emergency-action sequencer is therefore operable in the presence of circuit troubles in the microsecond clock.

Each sequence of output pulses generated by the pulser and delay line defines an emergency action. Each emergency action increments an emergency-action state counter, initiates a B-level interrupt, and sets and resets selected status flip-flops in the duplicate central control. Switching these flip-flops results in a corresponding rearrangement of active and standby central controls, program store input and output connections, and selection of connecting buses. The rearrangements made during each emergency action depend on the internal state of the

emergency-action state counter. It is quite possible that a given rearrangement of duplicated units in the central processor will not result in an operable combination of subsystems; the result will be the reactivation of the emergency-action sequencer which, under control of the emergency-action state counter, forms a new arrangement.

When an operable combination of units have been connected together to become the active central processor, the execution of a test program (initiated with a B-level interrupt) can be completed in approximately 100 machine cycles. If the active central processor contains faulty equipment which inhibits the proper execution of the test program sequences, the sanity timer will recycle in approximately 128 machine cycles. Such recycling reactivates the emergency-action sequencer.

A more detailed description of sequencer actions and the additional hardware and program checks performed with the emergency-action sequencer are covered in a companion paper.<sup>14</sup>

#### *7.6 Maintenance Orders*

Included in the order structure of the central processor are classes of orders designed specifically for use in maintenance program sequences to obtain test results or to place test signals within central controls, call stores, and program stores. These orders perform special gating actions that are either inconvenient or impossible to obtain with combinations of other orders described above. The maintenance orders include the following classes: (1) G- and H-mode memory reading and writing orders, (2) control mode memory reading and writing orders, and (3) miscellaneous test and test signal orders.

#### *7.7 G- and H-Mode Memory Orders*

As previously noted, all program and data information is duplicated; each word stored in one program store unit or call store unit is also stored in another store unit. There are many situations where initial installation requirements or growth in an office require an amount of duplicated semipermanent and/or temporary memory that could be satisfied with an odd number of store units. The duplication scheme employed in the No. 1 ESS central processor permits the use of an odd number of units to gain economy over a system using only an even number of stores.<sup>14</sup> In this duplication scheme each store unit is divided into two blocks of memory. Each block or half-store is assigned a different code. One block in the memory unit is designated the H block; the remaining block is labeled the G block of memory. All the information appearing in the H block of one store unit is duplicated in the G block of another store unit.

Commands to read or write in memory include a code and address. Each code and address corresponds to one word of information in memory, but since each word is duplicated in the memory, the code and address correspond to two memory locations, one appearing in the G block of one store unit and the other location appearing in the H block of a different store unit.

The usual communication of data and program words between the central control and connecting stores is accomplished with normal mode commands. Each normal mode command is capable of generating a simultaneous response in the store units containing the G and H duplicate memory locations; however, no store unit simultaneously receives two commands to which it is to respond, and no central control unit simultaneously receives both of the duplicate responses.

When troubles are detected in the communication of information between the stores and the central controls, the remedial actions include program sequences to examine individual store units; it is desirable to read or write test information specifying the store unit which is to respond to these commands. Consequently, the order structure includes G and H memory reading and writing order words or instructions that specify which duplicate location is to be read or written. These memory reading and writing orders are processed in central control like the normal mode reading and writing orders: the codes and addresses are generated in the same fashion, but the command includes G- or H-mode signals. For example, a G-mode memory reading order will obtain information from only the duplicate unit which contains the G image of the memory location specified by the code and address. H-mode memory reading and writing commands make a similar distinction.

The G- and H-mode memory reading orders include additional features when applied to the reading of data in program stores. Certain of these orders call for the reading of data from the G or H locations, respectively; when the data is obtained from a program store, the readings are accepted and processed as valid data without carrying out any rereading or correcting steps as indicated by the check circuit. Other G and H memory reading commands, when applied to program store memory locations, permit the correction of data as required, but no rereadings may take place. The G and H memory reading and writing instructions just described are given in Table II.

### 7.8 Control Mode Orders

Control orders resemble the normal mode memory reading and writing orders in that central control carries out the reading or writing of data in

TABLE II — G AND H MEMORY ORDERS

Mnemonic Representation	Order	Available Options
HKU	Read H image of specified memory location; no remedial actions for invalid responses; data reading replaces contents of accumulator	Indexing; index register modification options, including incrementing, W, and register S options; product (PL and PS) masking and complementing of data reading
GKC	Read G image of specified memory location; remedial action limited to single-error correction of program store data readings; data reading replaces contents of accumulator	
XGKU	Read G image of specified memory location; no remedial actions for invalid responses; EXCLUSIVE-OR of data reading and accumulator contents placed in accumulator	
XHKC	Read H image of specified memory location; remedial action limited to single-error correction of program store data readings; EXCLUSIVE-OR of data reading and accumulator contents placed in accumulator	
BG, BH	Place contents of data buffer register in specified G (H) call store memory location; no remedial actions for invalid responses	
FG, FH, KG, KH, LG, LH	Place contents of index register F (accumulator K, logic register L) in specified G (H) call store memory location; no remedial actions for invalid responses	Indexing; index register modification options listed above; product (PL and PS) masking; complementing and insertion (EL and ES) masking of data to be stored

such units as the program store and the call store. These orders differ from normal mode memory order in that: (1) when commands are transmitted, the mode signals which appear as part of these commands indicate the control mode, and (2) control mode memory writing orders may be used also to treat flip-flop registers in the stand-by central control as memory locations in a call store.

Control orders are designed to provide the convenient setting, resetting, and reading of status flip-flops and other test points in the program stores, call stores, and standby central control.

The control mode orders are listed in Table III; this table includes comments regarding the specific application of certain control mode

TABLE III — CONTROL MODE ORDERS

Mnemonic Representation	Order	Available Options
NB	Read specified control location; place reading in data buffer register	Indexing; index register modification options, including incrementing, W, and register S options
NF, NJ, NK, NL, NX, NY, NZ	Read specified control location; place reading in index register F (J, K, L, X, Y, Z)	Indexing and index register modification options: product (PL and PS) masking and complementing of data
NBTA, NBTB	Special control reading orders to test call store address translators; translated address placed in data buffer register	Indexing and index register modification options
BN	Place contents of data buffer register in specified control location	
FN, JN, KN, LN, XN, YN, ZN	Place contents of index register F (J, K, L, X, Y, Z) in specified control location	Indexing and index register modification options: product (PL and PS) masking, complementing, and insertion (EL and ES) masking of data to be stored
WNPS	Transmit control command (specifying) location and data for control flip-flops in a specified program store	Indexing

orders. For example, the order WNPS is designed specifically for writing information into duplication status and test flip-flops within the program stores. The program stores are the semipermanent memory of the system. No on-line writing of information within the twistor memory is possible and therefore none of the previously described writing orders has access to the program store. The WNPS order is executed by sending a command on the program store bus which indicates a control mode writing operation, and part of the address transmitted is treated by the responding program store as the data to be placed in control flip-flop registers within that store.

### 7.9 Miscellaneous Maintenance Instructions

In addition to the two classes of special memory reading and writing instructions described above, there are a number of specific orders, including some normal mode memory reading and writing orders designed



specifically to create trouble conditions not encountered in other normal memory commands or to initiate special tests.

Executing the order WV causes a data word to be transmitted to a special V register in central control; the outputs of this register are then transmitted as half-microsecond pulses to special points within the central control and via cable drivers and connecting twisted-pair cables to the other central control unit. This order is used, for example, for transmitting signals from the active central control to standby central control to:

- (1) start or stop data processing in the standby central control;
- (2) reset certain registers in the standby central control unit such as the buffer order word register and the order word register, or
- (3) generate maintenance interrupt signals in the standby central control.

A mismatch sampling order, EMMS, concurrently carries out a number of information processing steps to initiate the mismatch sampling mode as described in a companion paper.<sup>14</sup>

The remaining miscellaneous maintenance orders (BMAP, BMOP, MBOP, and MBCS) comprise normal memory reading or writing orders which are specifically designed for exercising or examining the parity generation and check circuits in both central control and the call store.

#### VIII. TIMING CONSIDERATIONS

The central processor is a synchronous data processing system; a microsecond clock in central control provides clock pulses defining a machine cycle and intervals or phases within that cycle. A significant aspect of both the logical organization and detailed circuit specifications of the central processor is the integration of the response times of program store and call store systems and the multiphase data processing steps of central control in response to each of a diversity of program orders. This integration began with the preliminary design of a central control and its order structure.

The classes of orders considered for inclusion in the order structure consist primarily of different combinations of meaningful data processing operations or steps such as indexing, index register modification, and the placing of a data word obtained from memory into a specific index register in central control. Each major data processing step is assigned to one or more clock phases; the minimum time for a clock phase is fixed according to the maximum propagation time of information through the longest logic chains corresponding to the data processing

steps assigned to that phase. These phases are then fitted into a machine cycle which approximately equals the minimum cycle times of communication between the central control and both the call store and program store; the relative placement of each of the phases is dictated by the time of appearance of related data processed in central control to form commands, addresses and data for communications between central control and its stores.

In the design of classes of order words and the determination of the number of clock phases, an extra clock phase is provided by overlapping operational steps of successive orders. To maximize the average data processing rate, most of the orders are designed to be executed at the rate of one order per machine cycle. Certain orders, such as transfer orders and orders to read data words from the program store, require more time and are designed to fit into a timing framework of two or more machine cycles.

Detailed timing studies based on min-max component tolerances served to provide a "paper simulation" of the central processor. This simulation revealed that a number of logic chains and communication paths are limiting in fitting all the operations steps into a 5.5-microsecond clock cycle; accordingly, an improvement of any one logic chain or subsystem could not materially increase the central processor's data processing speed capabilities.

### 8.1 *Single-Cycle Call Store Memory Orders*

The central processor is designed to execute most of its sequences of program orders at the rate of one order per machine cycle; such orders are referred to as "single-cycle" orders. Multicycle orders require additional time (2 to 4 machine cycles, depending on the order) and are executed with the aid of special sequential control circuits described later in this article. The data processing time of the single-cycle orders determines the machine cycle time and the allocation of phase intervals within this cycle; included in this class are memory reading and writing orders which receive and transmit data from memory locations in the call stores.

Call store memory reading and writing orders comprise an estimated 60 to 70 per cent of the instructions executed in call processing program sequences; the ability to execute these orders at the rate of one order per machine cycle defines a machine cycle, which may be limited by one or more of three considerations:

- (1) the maximum repetition rate of obtaining program order words from a program store system including up to 6 program stores;

(2) the maximum repetition rate of a call store system of up to 37 call stores in response to a random sequence of reading and writing commands generated in the execution of a sequence of call store memory orders; and

(3) the maximum turn-around time at which a data reading can be obtained from a call store and combined in an index adder to generate an address for use in a call store reading or writing command.

An additional requirement in allocating clock phase intervals is noted in fitting call store writing orders into the machine cycle. An upper bound is placed on the interval from the beginning of the clock phase which initiates indexing (to obtain a call store address) to the beginning of the clock phase which moves the data to be stored from an index register to the data buffer register.

There is no difficulty in fitting the data processing steps of the non-memory data processing orders into this framework; the only orders employed in significant numbers in the program sequences requiring more than one machine cycle for their execution are orders to transfer program control and memory reading orders which obtain data words from the program store.

### 8.2 *Phases of the Machine Cycle*

The overlapped execution of sequential program orders combined with 3 basic data processing phases per machine cycle provides four data processing intervals per instruction. This is a sufficient number of intervals for all the data processing steps required for transmitting information via one or both of the unmasked bus and masked bus from one register to another through logic combining circuits such as the index adder or the mask and complement circuit.

Fig. 31 shows how call store memory orders are fitted into the overlapped execution of single-cycle orders. An order is obtained from a program store during phase 1 of cycle 1 in response to a program store command transmitted from the central control in the preceding machine cycle. At the beginning of this phase, the buffer order word register is reset to erase the preceding order word, and the register inputs are connected to the program store response bus. Depending on the response time of the program store unit and its distance from central control, the order word arrives at some time during phase 1. In this interval, the command and address for the first succeeding program order are transmitted to the program store.

Phase 2 and 3 clock pulses are applied to the buffer word decoder to control indexing and index register modification during cycle 1. The indexing addition is completed towards the end of phase 2 and the sum

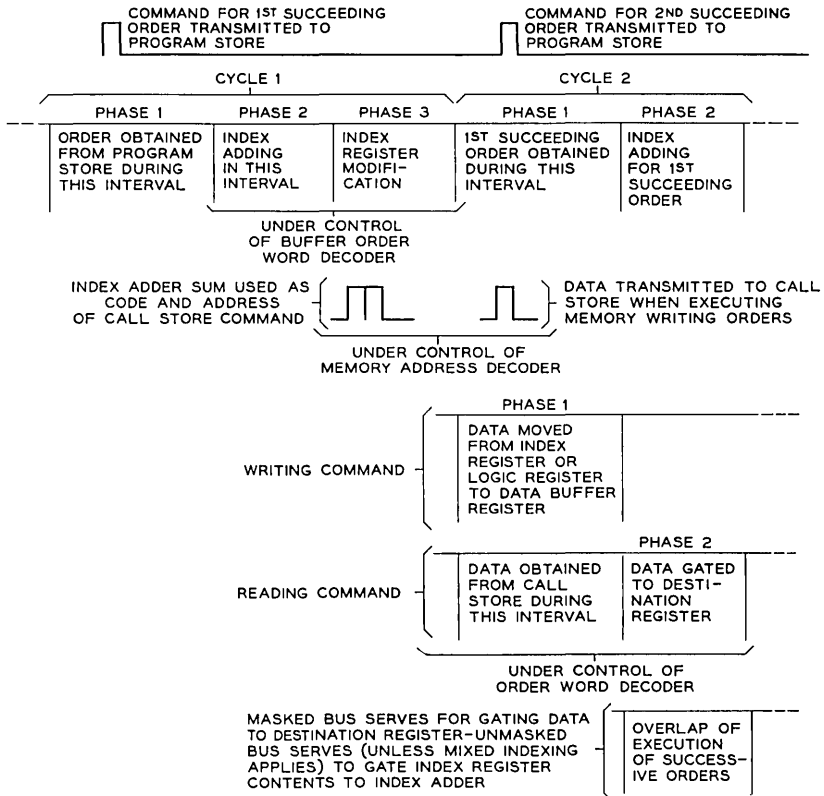


Fig. 31 — Timing of execution of single-cycle memory orders.

is gated to an index adder output register (contained in the index adder). This register serves as an address register for call store memory orders. Index register modification is performed during phase 3, and the call store command and address are concurrently transmitted under control of the memory address decoder to the call store.

At the beginning of phase 3 of cycle 1 the call store memory order depicted in Fig. 31 is transmitted from the buffer order word register to the order word register. Phase 1 and phase 2 clock pulses applied to the order word decoder and memory address decoder carry out the remaining steps for the order. Two sequences of these remaining steps are shown: the first sequence is that of a call store memory writing order, and the second corresponds to a call store memory reading order.

Call store memory writing orders select the contents of one of the

seven index registers or the logic register to be transmitted as data to the call store. The data buffer register in central control is connected to the call store data transmission and call store response buses; accordingly, the contents of the selected register are transmitted via the unmasked bus, the mask and complement circuit, the masked bus and the insertion mask to the data buffer register. A parity generator connected to the outputs of the data buffer register produces a parity bit which is then transmitted simultaneously with the 23-bit data word to the call store to complete the call store memory writing order.

Call store memory reading orders employ phase 1 of cycle 2 to accept data words by connecting the inputs of the data buffer register to the call store response bus. The data buffer register is reset at the beginning of phase 1 and the data word is received at some time during phase 1, depending on the response time of the call store unit and the distance between that unit and the central control.

During phase 2 of cycle 2, the data reading is transmitted to the destination register selected by the memory reading order by gating the reading through the data buffer register and the mask and complement circuit onto the masked bus.

Concurrently with the completion of the memory reading order, the first succeeding order begins its indexing. That is, phase 2 of cycle 2 includes the transmission of the contents of a selected index register to the index adder via the unmasked bus. Certain pairs of orders occur where the first order reads data from a call store and the second order is a call store memory order which selects as its index register the destination register of the first order. (These pairs of orders require the previously described mixed indexing.) To execute such pairs correctly, the lower bound on the machine cycle time must equal or exceed the maximum round-trip time of call from call store command to response, including the indexing addition in central control.

Indexing and index register modification of all other orders are also performed during phase 2 and phase 3 of the first cycle, as shown for memory orders in Fig. 31. W class combining orders move the data word to a specified destination register during phase 1 of the second cycle; shift and rotate orders are also completed in the same interval. The earlier completion of these orders (as compared to memory reading orders) permits the accumulator to be selected as the index register in the succeeding order; such a selection is not feasible when the first order is a memory reading order moving data to the accumulator, since the accumulator will not contain the correct result in time for indexing during phase 2.

The data processing steps of orders such as UWX, PMZ include moving the contents of an index register (X,Z) to the logic register; this step is performed using the unmasked bus during phase 3 of the first cycle. The unmasked bus is available, since the index register modification step requires the use of only the masked bus; accordingly, the two steps may occur concurrently, as required by the order.

Whenever an order specifies that the data word is to be transmitted to the logic register (PS or ES masking) this step is accomplished during phase 3 of the first cycle under control of the buffer order word decoder.

Decisions are made by the order word decoder for regular transfer orders at the beginning of the second cycle; decisions for early transfer orders are made at the beginning of phase 3 of the first cycle. In both cases, the decision to transfer is implemented by activating a sequencer at the stated times.

### 8.3 *Basis of Timing Specifications*

Figs. 32 and 33 show the response times of the program stores and call stores in terms of minimum and maximum calculated response times. To provide sufficient spacing for large numbers of program stores and call stores, bus lengths from a few feet to a maximum of one hundred feet were assumed for each bus. The buses are all twisted-pair cables having a delay constant of approximately 2 nanoseconds per foot.<sup>12</sup> A single program store command bus system connects both central control units to all program stores; each store has a lightly coupled set of pulse receivers connected in series with the bus system, and the bus ends with a terminating resistor. These receivers add a small fixed delay to the bus system, 1.5 nanoseconds per receiver, which is insignificant in most timing considerations. Similarly, a single program store bus system connects all program store memory outputs (cable drivers are shunt connected) to inputs in both central controls. A similar arrangement of shunt and series connections of cable drivers and pulse receivers connects the central controls and the call stores. These connections include a call store command bus system, a call store response bus system, and a call store data transmission bus system.

To predict the delays of logic chains in central control, the characteristic minimum and maximum delay times of the low-level logic (LLL) AND-NOT gate<sup>6</sup> and the medium-current logic (MCL) AND-NOT gate were determined, as tabulated in Fig. 34. These limits include allowance for lead capacitance, and wiring rules restrict the actual capacitances to less than this allowance.<sup>6</sup> The characteristics of the cable driver (CD), cable pulse receiver (CPR) and clock pulse output amplifier (CPOA)

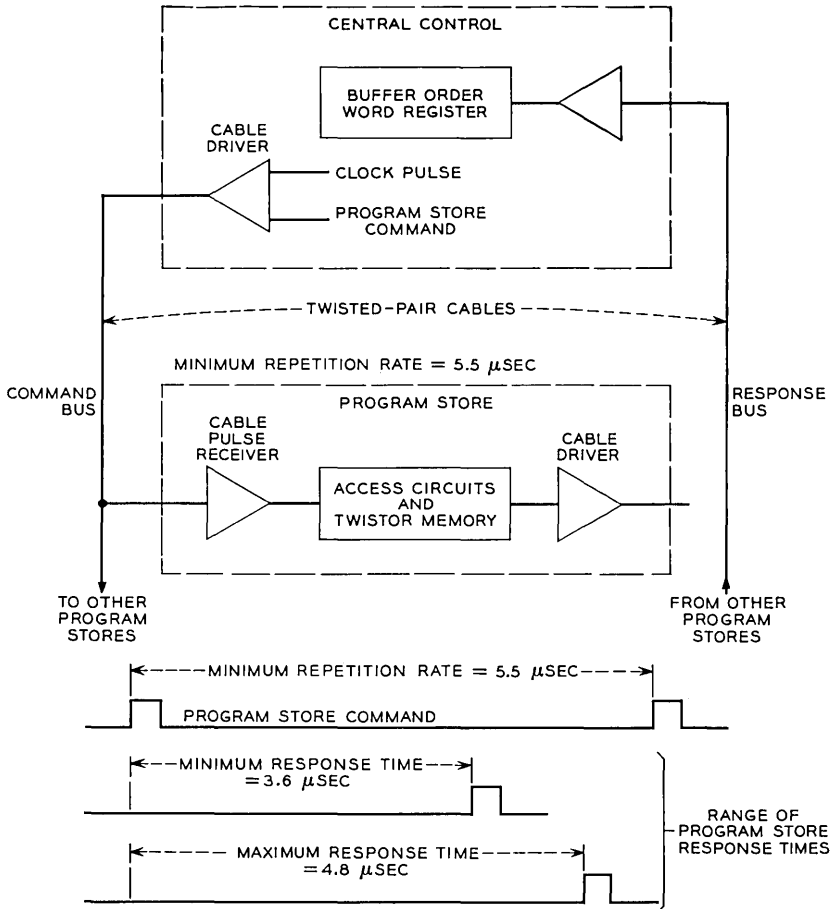


Fig. 32 — Program store timing tolerance limits.

gates complete the information needed for calculating propagation delays of central control.

In certain timing chains, minimum pulse widths become a part of the design considerations, and therefore a pulse-shortening characteristic for each class of gate is included in Fig. 34. It is assumed that the turn-on time of an LLL gate or an MCL gate cannot exceed its turn-off time; a consequence is that reduction in the width of a pulse propagating through a number of stages of logic gates due to differentials in turn-on and turn-off times is calculated to occur only in alternate stages which have negative-going pulse inputs.

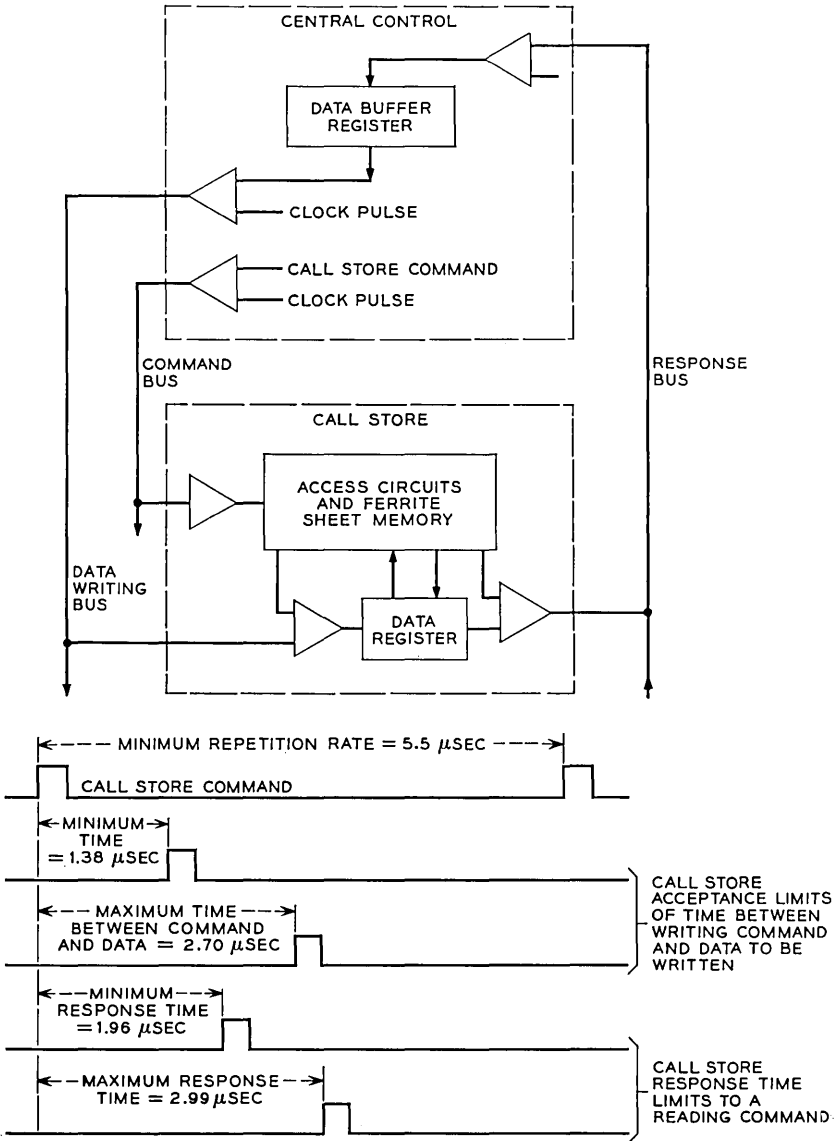


Fig. 33 — Call store timing tolerance limits.



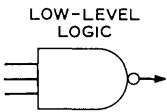
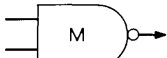
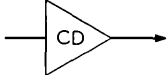
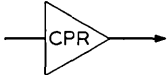
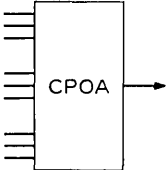
	TURN-ON TIME IN NANOSECONDS	TURN-OFF TIME IN NANOSECONDS	MAXIMUM SHRINKAGE OF PULSE WIDTH
 LOW-LEVEL LOGIC LLL	10-65 NS	10-65 NS	55 NS - FOR NEGATIVE-GOING INPUT PULSES ONLY
 MEDIUM CURRENT LOGIC M MCL	10-75 NS	20-85 NS	75 NS - FOR NEGATIVE-GOING INPUT PULSES ONLY
 CD CABLE DRIVER	10-75 NS	10 NS-UNSPECIFIED	NONE
 CPR CABLE PULSE RECEIVER	10-65 NS	10 NS-UNSPECIFIED	NONE
 CPOA CLOCK PULSE OUTPUT AMPLIFIER	10-75 NS	10-75 NS	20 NS

Fig. 34 — Logic package timing characteristics.

#### 8.4 Microsecond Clock Characteristics

The multiphase clock in central control defines the machine cycle of 5.5 microseconds and the three phases within the machine cycle. In addition, a number of shorter pulses are provided by the clock to carry out the gating of commands and addresses from central control through cable drivers onto twisted-pair cables connecting to stores, central pulse distributors and peripheral units such as scanner and network controllers. Other short pulses are required for resetting registers and gating informa-

tion from one flip-flop register to another at times other than the three principal phases; data processing results are thereby transmitted as they are completed. Half-microsecond pulses spaced at quarter-microsecond intervals over the entire machine cycle are provided to permit maximum flexibility in the logic design; timing chains were easily modified as additions and alterations were made in the development of the logical organization of the central processor. This large array of pulses thus satisfies the need for (1) selecting optimum gating times for internal data processing transmission and (2) obtaining a useful minimum pulse width (where reasonable clock pulse tolerances are a consideration) for (a) gating information from one register to another, (b) resetting a register, and (c) transmitting a pulse of sufficient width over twisted-pair bus systems to the most distant stores, central pulse distributors, and peripheral units.

The microsecond clock for central control includes a crystal-controlled 2-megacycle oscillator driving an 11-stage counter. The outputs of the counter drive logic chains which translate the states of the counter into the array of half-microsecond pulses and the basic phase pulses are shown in Fig. 35. The machine cycle is divided into 22 quarter-microsecond

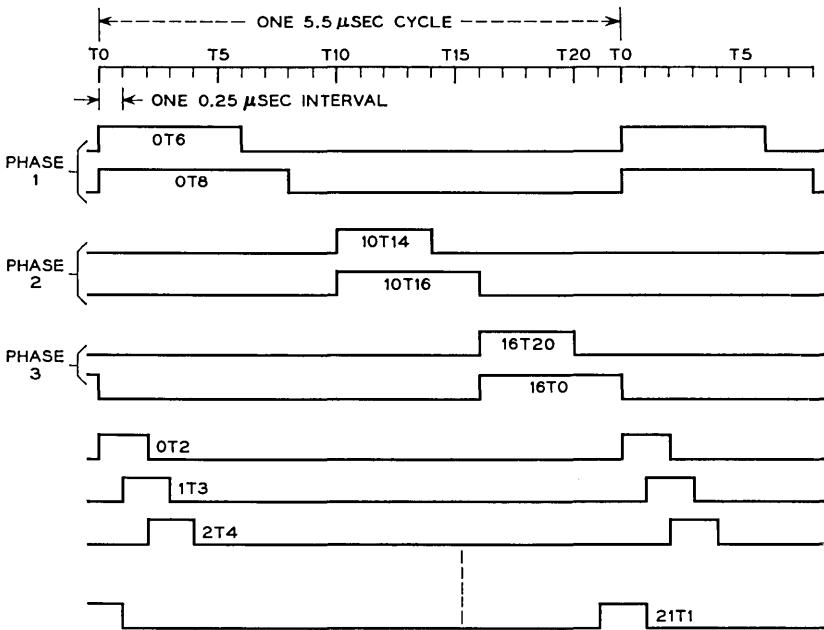


Fig. 35 — Phases of central control microsecond clock.

intervals and the beginning of each interval is denoted as  $T_0, T_1, \dots, T_{21}$ . Each clock pulse begins and ends at one of the times  $T_0, T_1, \dots, T_{21}$ , so that each clock pulse is labeled  $ATB$ , where  $A$  is a number corresponding to the time of the leading edge of the pulse and  $B$  corresponds to the trailing edge time.

Both central controls contain a complete clock, but the oscillator in the active central control drives the 11-stage counter in both central control units to keep the units closely synchronized. To keep the counters in step, a phasing signal is transmitted from the active central control counter to the standby unit once every machine cycle.

The tolerances on the clock outputs in each central control and the cross-connection tolerances are tabulated in Fig. 36. Fig. 35 depicts an ideal set of clock pulses and the table indicates additional delays from that ideal. That is, the figure represents all minimum delay conditions in the clocks, and only positive tolerances appear in Fig. 36. This approach simplified the many calculations to be made in the design. Only minimum/maximum values were substituted into delay equations, rather than nominal values plus or minus a tolerance figure. Examples of this technique follow.

8.5 Sample Timing Calculations

The half-microsecond pulses are employed in transmitting multibit commands and addresses to units connecting to the central control. To meet high fan-out requirements, a clock output amplifier is usually interposed between the clock pulse output and the array of cable drivers to be pulsed. This connection is exemplified in Fig. 37 using the pulse  $OT_2$  for purposes of illustration.

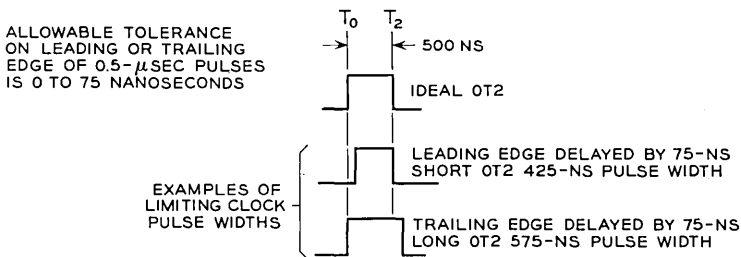


Fig. 36 — Microsecond clock tolerance limits. Allowable tolerance on leading or trailing edges of bus phase pulses ( $OT_6, OT_8, 10T_{14}, 10T_{16}, 10T_{20}, 10T_{22}$ ) is 0 to 150 nanoseconds. Allowable tolerance of propagation of oscillator signal from active to standby central control is such that idealized phases of Fig. 35 in standby central control lag by 0 to 75 nanoseconds.

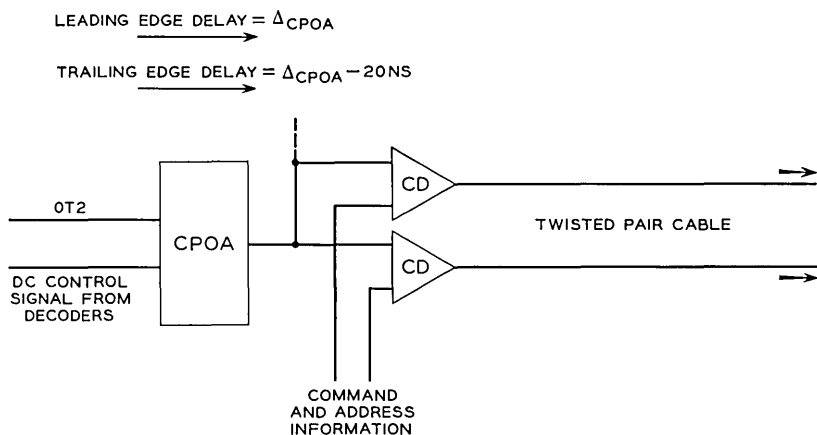


Fig. 37 — Clock pulse transmission chains.

A timing calculation was made to determine if such an arrangement would always provide command and address pulses of sufficient width (250 nanoseconds minimum) to a receiving unit located at the maximum allowable distance from central control. To determine this minimum pulse width, one first assumes the shortest allowable clock pulse and then calculates the maximum pulse shortening in logic chains and cables.

Referring to Fig. 36, the latest leading edge of the pulse would occur at  $T_0 + 75$  nanoseconds, and the earliest trailing edge would appear at  $T_2$ , yielding a minimum clock pulse width of 425 nanoseconds. According to Fig. 34, the clock pulse output amplifier may shorten this pulse as much as 20 nanoseconds, and the cable driver would contribute no pulse shortening. Thus the minimum pulse width at the cable driver output is 405 nanoseconds.

Empirical equations for the shrinkage of the width of current pulses on a twisted-pair cable with an arbitrary number of series-connected cable receiver transformers as a function of that number ( $N$ ) and the length of the twisted-pair cable ( $L$  in feet) have been derived.<sup>16</sup>

$$\text{Shrinkage in nanoseconds} = 50 \ln \left( \frac{950}{950 - L - 5N} \right). \quad (1)$$

The bus system connecting central control to the peripheral units may be required to serve a large number ( $N$ ) of such units and the bus length be correspondingly large. With limits of  $N = 50$  and  $L = 450$  feet the maximum shrinkage would be approximately 70 nanoseconds. Thus the

minimum pulse width at the input of the most distant peripheral unit would equal or exceed  $(405 - 70) = 335$  nanoseconds, which meets the peripheral unit minimum pulse requirements.

The previously described round trip of one call store reading to be used in generating a call store command is outlined in Fig. 38 and results in a round-trip time of 5.36 microseconds, which fits into a 5.5-microsecond cycle. The calculated round-trip time does not allow for a slight delay in the loop deriving from clock pulse gating between the index adder and the index adder output register. Further, it should be noted that the late clock pulse arriving at point Y cannot be considered, since the memory reading may be obtained in response to a call store command generated in the active central control and utilized in the standby central control for generating a call store address. The 10 LLL delays in the index adder represent the maximum delay of carry propagation signals. Other adder designs with shorter delays were considered; however, smaller delays

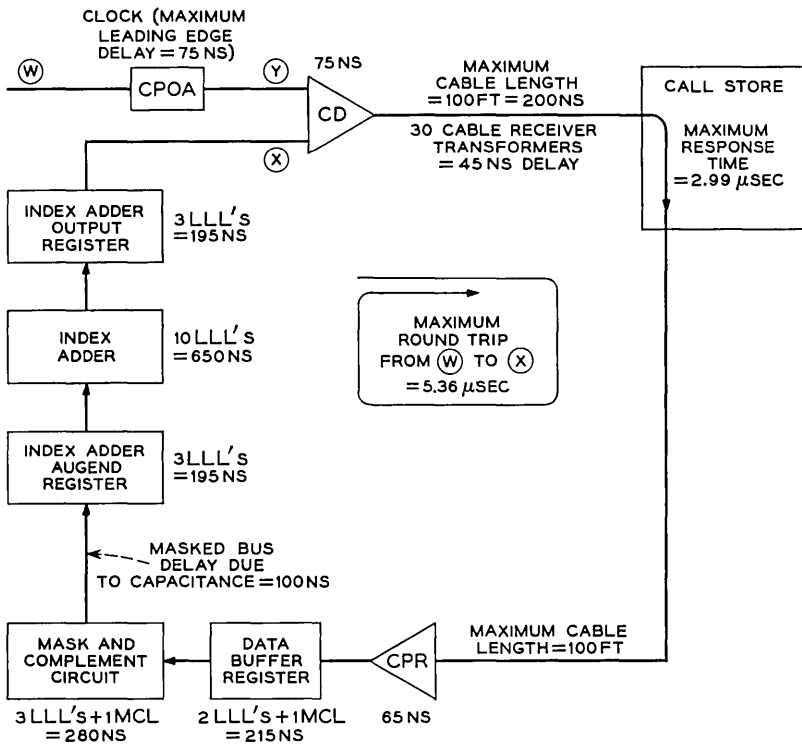


Fig. 38 — Call store round-trip time.

and a slightly shorter machine cycle could be achieved only with much larger adding circuits.

Other timing chains not described here also work with the 5.5-micro-second cycle but could be made to work with a shorter cycle only with additional logic circuits. Other limiting cases not described in detail here include:

- (1) communicating commands to peripheral units once every two machine cycles to establish a maximum rate for supervisory scanning of customer lines,
- (2) round-trip times of call store memory reading orders including the cross connection of associated check failure signals between central control units, and
- (3) various combinations of related sequences of data processing steps in the execution of consecutive orders.

Fig. 39 illustrates some of the constraints that govern the subdivision of the clock phases within the machine cycle. The longer of the pair of pulses for each phase must not overlap, but time intervals between phases is allowed.

Phase two data processing includes the last step of a memory reading order where the contents of the data buffer register are transmitted via the mask and complement circuit and the masked bus to a selected destination register. This step requires a one-microsecond interval to complete the required propagation of information, and therefore phase two comprises the one-microsecond bus sampling interval 10T14 and a corresponding source-to-bus interval 10T16. A similar propagation delay requirement for index register modification options applies to phase

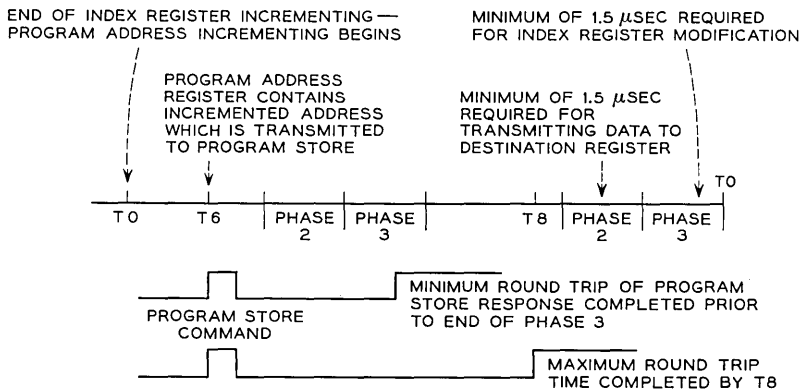


Fig. 39 — Fitting of clock phases into a machine cycle.

three, where the contents of the index adder output register are transmitted via the mask and complement circuit and the masked bus to the selected index register. Accordingly, clock pulses 16T20 and 16T22 define phase three.

Phase two and phase three require 3.0 microseconds of the machine cycle, and the incrementing circuit is employed throughout this interval for index register incrementing. The end of this interval is represented as T22 or T0, the beginning of the next machine cycle. Starting at this time, the contents of the program address register are transmitted to the incrementing circuit; the incremented quantity is then returned to the program address register and transmitted as part of a program store command to obtain the next program order word in sequence. The maximum time for the above steps is 1.5 microseconds, so that the program command is transmitted during 6T8 to the program store command bus. The maximum round trip returns the program word by T8 of the following cycle. This leaves a reasonable margin of 500 nanoseconds for signals to propagate through the buffer order word decoder before beginning gating actions starting at T10.

Calculations indicate that the minimum round-trip time to the program store returns the next program word before the buffer order word decoder has completed gating actions on the immediately preceding order. To circumvent a conflict on the use of the buffer order word register, an auxiliary register is placed between the program store response bus and the buffer order word register. For phase three gating actions, the operation field of the order in the buffer order word register must be retained, but not the accompanying 21-bit data field portions and 7-bit Hamming and parity portions of that program word. Accordingly, only the 28 corresponding cells of the buffer order word register are reset and connected to the program store response bus system during phase three; the auxiliary register serves to receive and retain the 16-bit operation field of the succeeding order until the buffer word decoder has completed phase three data processing steps for the current order. The contents of the auxiliary register are then transmitted to the buffer order word register during phase one of the following cycle in preparation for data processing steps beginning at phase two.

The above sample calculations are representative of those made as part of determining the realizability of the logic organization and detailed circuit design of the central processor for No. 1 ESS. These calculations served not only as a check of feasibility of proposed circuit design but also as guides in determining specifications for:

(a) tolerance limits on duplicate multiphase clocks driven from one of two crystal-controlled oscillators,

(b) maximum lengths of twisted-pair cable connecting central control to its communicating units, and

(c) tolerances between individual pairs and between buses of twisted-pair cables.

#### IX. DESCRIPTION OF SEQUENCER FUNCTIONS AND DESIGN

As indicated in Fig. 31, the buffer order word decoder, mixed decoder, and order word decoder serve to execute sequences of single-cycle overlap orders at the rate of one order per 5.5-microsecond machine cycle. The overlap of data processing occupies only one phase of the machine cycle, but when the generation of commands to obtain program order words is considered an additional degree of overlap is evident. The execution of certain orders (as well as hardware remedial actions such as the automatic rereading of information from the program stores and call stores) requires either the insertion of additional cycles of data processing or an extension of overlap to provide the necessary time. In providing these additional gating actions in central control a number of sequencers are provided; each sequencer carries out a specific class of data processing or remedial actions. Those sequencers which insert extra machine cycles inhibit the decoders to momentarily halt the flow of instructions obtained from the program store, and in this interval additional gating actions are carried out. Other sequencers do not insert cycles but extend the overlap by operating concurrently with the decoders; the decoders continue processing sequences of orders without regard to sequencer gating actions; in this latter instance, programming restrictions are applied to prevent conflicts in the overlapping flow of data processing. A brief description of the functions associated with some typical sequencers follows.

##### 9.1 *Data Reading Sequencer*

All of the memory reading orders previously described (including G- and H-mode memory reading orders and control mode memory reading orders) may obtain data not only from the call stores but from any location within one of the program stores. Each of these memory reading orders generates a code and address during its indexing cycle, and the code so generated determines the memory location to be read and hence whether the data word is to be obtained from a call store or a program store. If the code and address refer to a call store location, then the central control carries out the single-cycle memory reading operation previously described. Whenever the code address refers to a program store



location, the data reading sequencer is enabled to obtain these data. Once this has been accomplished the data reading is in the same position as data obtained from the call store (i.e., placed in the data buffer register), and the data reading sequencer returns to the inactive state. The order word decoder then proceeds to complete the moving of the data from the data buffer register to the destination register specified in the memory reading order.

As the data reading obtained is returning from the program store the data reading sequencer simultaneously returns the address of the next instruction in sequence to the program address register. Consequently, as the order word decoder is completing the processing of the data reading the next order word in sequence is returned and data processing from that order begins under control of the buffer word order decoder.

### 9.2 *Transfer Sequencer*

Because of the degree of overlap in the central processor a transfer sequencer is enabled when a transfer is to be executed. The transfer sequencer stops the flow of instructions for at least one cycle, until: the transfer address has been placed in the program address register, transmitted to the program store, and the first program order word of the new sequence of instructions has been returned to the central control. In addition, the transfer sequencer controls gating actions required for indirect transfers, and in such instances one or two additional machine cycles are inserted, depending on whether the transfer address is to be obtained from a call store or program store location.

A class of early transfer orders is included in the order structure; the term "early" alludes to the enabling of the transfer sequencer at a time in the machine cycle earlier than that performed for regular transfer orders. The early enablement of the transfer sequencer serves to inhibit the decoders, so as not to carry out the reading or writing operation when the decision is to execute the transfer of program control.

### 9.3 *Program Store-Correct Reread Sequencer*

When program or data words are read from the program store, checking circuits in central control examine the 44-bit word received and the previously transmitted program address which has been retained in central control for such checking purposes. Output signals from the checking circuit indicate either:

- (1) that all checks are passed and central control continues the processing of that word,
- (2) a single error is detected in the 44-bit word received, or

(3) an error is detected in the address transmitted, or an even multiple error is detected, or the program store response did not include an all-seems-well signal. The program store correct-reread sequencer responds to conditions (2) and (3) to correct or to reread and recheck the program store response. The program store correct-reread sequencer increments one of two binary counters for each word corrected or reread. The counters are periodically interrogated to determine the rate at which central control is receiving program store responses containing single or multiple errors.

The failure of the program store correct-reread sequencer to succeed on a retry is designated a "reread failure condition," which requires maintenance action, since the repeated trouble condition is not assumed to be due to a transient error condition. The program store correct-reread sequencer carries out several hardware steps leading to these maintenance actions. First, a maintenance interrupt source signal is generated to switch control to the E-level maintenance interrupt program sequences. Second, the program store correct-reread sequencer stores in a special register in central control the information indicating the trouble condition associated with the rereading failure. The contents of this program store error summary register indicate whether the detected trouble condition was due to a double-error condition, an error in the transmitted address, or a failure of the program store to return its hardware check signal.

Each of the sequencers described here consists of an individual counter and gating circuits within central control; however, the actions taken by one sequencer may serve as part of the data processing steps of other sequencers in central control. For example, the program store correct-reread sequencer is activated to correct or reread program order words, but it is also responsive to data readings obtained from the program store by the data reading sequencer and to indirect transfer addresses obtained from the program store by the transfer sequencer. Whenever the data reading sequencer obtains a data word from the program store, the previously described checks are made, and when correction or rereading is required the program store correct-reread sequencer is activated. In such instances the program store correct-reread sequencer inhibits the gating actions of the data reading sequencer and prevents the counter of the data reading sequencer from advancing to the next internal state. The program store correct-reread sequencer then proceeds to carry out the required correction or rereading, and upon conclusion of its remedial actions returns to the inactive state. This last action automatically releases the data reading sequencer, which continues its handling of the

corrected or reread data word. Similar interactions occur between the program store correct-reread sequencer and the transfer sequencer.

#### 9.4 *Accumulator Sequencer*

This sequencer provides gating actions necessary to the completion of a special class of memory reading orders. The orders are those which perform a memory reading and transmit this memory reading to the accumulator system to become one of the operands in an arithmetic or logical combining operation to be completed in the accumulator. Since the logic combining operation requires additional data processing time, the last gating action (the gating of the resulting combination into the accumulator register) cannot be completed in the time framework for the single-cycle instruction shown in Fig. 31. Accordingly, the accumulator sequencer is enabled whenever an order of this class is executed, and it completes its gating action in one additional clock phase. This sequencer differs from the previously described sequencers in that it shares control of gating actions with the decoders and extends the degree of overlap, rather than inserting additional machine cycles. Such additional time is required to process a memory reading when it is transmitted to the accumulator system, and since extended overlap rather than inserted machine cycles is used in this instance, a memory reading order which uses the accumulator as the destination register may not be followed by an order which uses the accumulator as the index register.

#### 9.5 *Peripheral Sequencer*

The peripheral sequencer is similar to the accumulator sequencer just described in that it carries forward the data processing and instructions on an extended overlap basis rather than within inserted machine cycles. It differs from the accumulator sequencer in that its actions extend over a period of nearly two machine cycles to carry out the transmission of commands and addresses to the central pulse distributor and (as required) to all of the peripheral units such as scanners, signal distributors, network controllers, and so on. The sequencer continues to remain active in order to gate check signals and scanner responses from the addressed units, and to generate trouble signals if any of the appropriate checks fail.

The peripheral orders are used in repetitive scanning operations of lines, trunk circuits, and junctors. To achieve efficient use of real time, the maximum scanning rate of one scan for every two machine cy-

cles is provided. In such instances, the peripheral sequencer remains active during the execution of such a sequence of orders and does not return to the inactive state until (1) the scanning sequence has been completed, (2) an error in the response of the central pulse distributor or peripheral unit is detected, or (3) an interrupt intervenes in the execution of the scanning sequence.

### 9.6 *Sequencer Design*

A total of ten sequencers is contained in central control. The remainder of the sequencers serve in multicycle operations — for retrieval of call store reading and writing operations, for special circuit actions and subsequent return from interrupted programs, for stopping and restarting data processing in the standby central control, and for obtaining special maintenance program sequences from the call store.

The ten sequencers perform different data processing steps as required in the execution of program sequences by central control. The sequencers are designed as a collection of individual counter circuits rather than as one large counter. As noted, one sequencer may “call” a second as required to carry out a class of data processing steps associated with the second sequencer. Thus the decomposition of a sequencer counter results in relatively efficient use of the number of internal states required of the counters. Perhaps more importantly, the flexibility of design gained by this decomposition has proved helpful in developing circuit specifications in the face of simultaneously evolving requirements for each of these circuits.

The heart of each sequencer is a synchronous counter. This synchronous counter responds to input signals from the decoders and check circuits and selected phases of the microsecond clock. It is enabled and advanced through various active states at definite times within a sequence of machine cycles. Although the counters are synchronous in the sense just described, the counters are designed as asynchronous counters and do not require delay lines in their feedback loops. This simplification is achieved by requiring that at least two changes of state must occur in any 5.5-microsecond interval of time (unless of course the sequencer is inactive or being inhibited by another sequencer).

An example of a sequencer counter is shown in Fig. 40 along with the time diagram showing the advancing of the sequencer through a succession of its active states.

Assuming the inactive state corresponds to both flip-flops being reset and assuming that an enable signal appears by T0 of a given machine

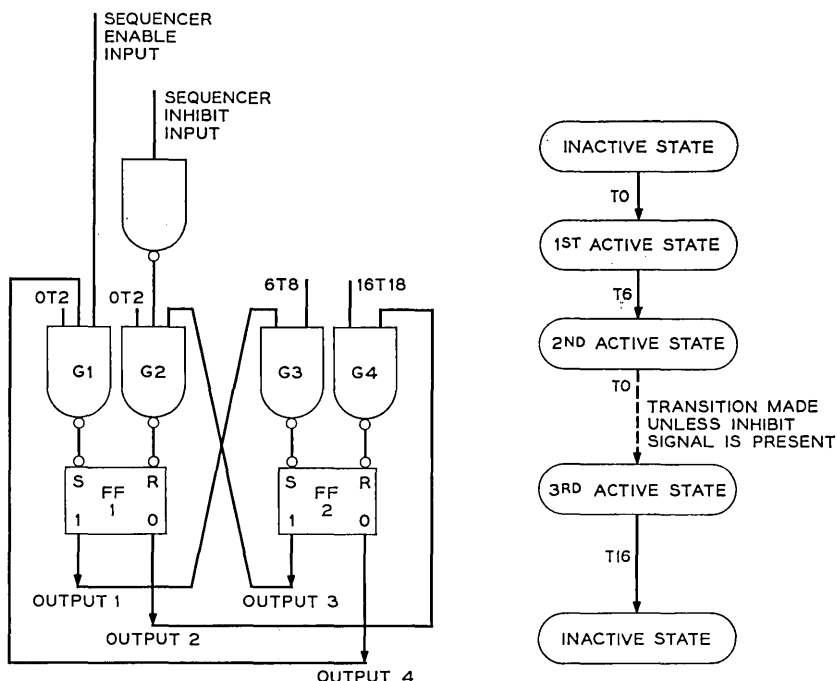


Fig. 40 — Sequencer counter.

cycle, this sequencer is enabled by activating gate  $G_1$  at  $T_0$  to set flip-flop 1. This in turn enables output 1 so that gate  $G_3$  is enabled at  $T_6$ , and the sequencer advances from its first active state to the second active state at this time. Accordingly, by  $T_0$  of the following cycle a signal appearing on output 4 will disappear so that gate  $G_1$  cannot be again activated in the immediately succeeding machine cycle. Instead, gate  $G_2$  will be enabled to reset flip-flop 1 unless a signal appears on the inhibit input at this time. If an inhibit signal appears (e.g., the output of another sequencer), the sequencer will remain in the second active state until the inhibit signal disappears and a new machine cycle begins with the reappearance of the clock pulse  $0T_2$ .

The example in Fig. 40 illustrates a relatively simple counter design. Some of the more complicated counters, such as the transfer sequencer counter, include alternate sets of active states according to the type of transfer order being processed; the different sets of states serve to carry out different gating actions over different numbers of machine cycles.

The selection of clock pulses for the various counters is made to span

selected phases of various machine cycles to provide the simplest translation of counter state outputs into gating control signals. Where possible, these transitions were placed at such a time as to make sequential lockout of one sequencer to another possible; the extra connection of inhibit signals required in simultaneous lockout schemes is thereby avoided.

Although sequencers seize and release control of each of the decoders at different times (according to the function being performed), the grouping of decoder leads into four classes according to decoders and clock phase suffices for all but a few of the decoder-controlled gates. Accordingly, the outputs of the sequencers are combined to generate four sets of inhibit signals to selectively seize and release control of central control gates for all situations in which the sequencers insert additional machine cycles. The gates that could not be so grouped consist primarily of those which control the flow of program order words from the program store to central control. These had to be handled separately to provide sequencer control gating actions that performed the necessary functions with a minimum number of inserted machine cycles.

#### X. SUMMARY

This article has described some aspects of the logic design of the central processor, including (1) design considerations, (2) a development of the program order structure, (3) a development of the logic blocks and their interconnections to implement the order structure, (4) a description of order encoding, (5) the circuits and program orders needed to meet maintenance objectives, (6) a discussion of timing requirements, and (7) a description of sequencing circuits required for multicycle data processing functions.

Table IV summarizes the number of components in each of the functional units in one central control unit.<sup>17</sup> One such unit occupies four standard No. 1 ESS bays<sup>6</sup> and requires approximately 2800 printed wiring boards of various logic packages.

This design of the central control represents a balanced compromise between the data processing capability, economy, and reliability of operation in a telephone switching system. Furthermore, the system is able to grow through the addition of modular memory and input-output equipment without extensive wiring changes. The use of overlap operation and the inclusion of special orders to carry out several steps of highly repetitive input-output functions simultaneously assist in obtaining high data processing capability.

TABLE IV — NUMBER OF TRANSISTOR GATES IN FUNCTIONAL UNITS IN ONE CENTRAL CONTROL UNIT

Functional Unit	No. Transistor Gates	Percentage of CC Total
Program store bus circuits	260	2.1
Call store bus circuits	290	2.3
Instruction registers	390	3.1
Decoders (including memory address decoder)	1160	9.3
Error-checking and -correcting circuits and parity generators	640	5.1
Sequencers and sequencer-controlled gates	870	7.0
Index adder system	810	6.5
Program store register, increment circuit, and auxiliary storage register	720	5.8
Mask and complement circuit and insertion mask	280	2.3
Peripheral communication circuits	1140	9.2
Index registers and logic register	1140	9.2
Accumulator system, including shifting and find-rightmost-one circuit	1050	8.4
Masked bus sampling gates (C flip-flops and logic)	80	0.6
Matching circuits	1860	15.0
Emergency-action circuit	170	1.4
Clock circuits	330	2.7
Miscellaneous buffer bus registers, including interrupt sources, central processor status and error summary registers	1010	8.1
Miscellaneous circuits, including power control and maintenance scanning access	240	1.9
Total	12,440 gates	100.0%

## XI. ACKNOWLEDGMENTS

Many of our colleagues contributed materially to central control organization. Mrs. E. S. Hoover and I. D. Nehama performed many of the early and fundamental systems studies which led to the present plan, A. H. Doblmaier contributed to the over-all organization of the central control, M. P. Fabisch worked out much of the encoding scheme, R. W. Downing specified most of the maintenance facilities, J. S. Nowak specified the emergency-action circuit, E. Graeve designed the clock circuits, and R. B. Smith and Miss V. R. Smith created the mnemonic representation of orders and the programmer's manual.

## REFERENCES

1. Keister, W., Ketchledge, R. W., and Vaughan, H. E., No. 1 ESS: System Organization and Objectives, B.S.T.J., this issue, p. 1831.
2. Biddulph, R., Budlong, A. H., Casterline, R. C., Funk, D. L., and Goeller, L. F., Line, Trunk, Junctor, and Service Circuits for No. 1 ESS, B.S.T.J., this issue, Part 2.
3. Freimanis, L., Guercio, A. M., and May, H. F., No. 1 ESS Scanner, Signal Distributor, and Central Pulse Distributor, B.S.T.J., this issue, Part 2.

4. Danielson, D., Dunlap, K. S., and Hofmann, H. R., No. 1 ESS Switching Network Frames and Circuits, B.S.T.J., this issue, Part 2.
5. Dougherty, H. J., Raag, H., Ridinger, P. G., and Stockert, A. A., No. 1 ESS Master Control Center, B.S.T.J., this issue, Part 2.
6. Cagle, W. B., Menne, R. S., Skinner, R. S., Staehler, R. E., and Underwood, M. D., No. 1 ESS Logic Circuits and Their Application to the Design of the Central Control, B.S.T.J., this issue, p. 2055.
7. Harr, J. A., Hoover, Mrs. E. S., and Smith, R. B., Organization of the No. 1 ESS Stored Program, B.S.T.J., this issue, p. 1923.
8. Carbaugh, D. H., Drew, G. G., Ghiron, H., and Hoover, Mrs. E. S., No. 1 ESS Call Processing, B.S.T.J., this issue, Part 2.
9. Ault, C. F., Gallaher, L. E., Greenwood, T. S., and Koehler, D. C., No. 1 ESS Program Store, B.S.T.J., this issue, p. 2097.
10. Genke, R. M., Harding, P. A., and Staehler, R. E., No. 1 ESS Call Store — A 0.2-Megabit Ferrite Sheet Memory, B.S.T.J., this issue, p. 2147.
11. Ulrich, W., and Vellenzer, Mrs. H. M., Translations in No. 1 ESS, B.S.T.J., this issue, Part 2.
12. Connell, J. B., Hussey, L. W., and Ketchledge, R. W., No. 1 ESS Bus System, B.S.T.J., this issue, p. 2021.
13. Martellotto, N. A., Oehring, H., and Paull, M. C., Process III, A Compiler-Assembler for No. 1 ESS, B.S.T.J., this issue, Part 2.
14. Downing, R. W., Nowak, J. S., and Tuomenoksa, L. S., No. 1 ESS Maintenance Plan, B.S.T.J., this issue, p. 1961.
15. Tuomenoksa, L. S., and Ulrich, W., Coding and Information Identification, IEEE Trans., **67**, 1963, p. 403.
16. Butler, T. T., unpublished work.
17. Menne, R. S., unpublished work.



# Organization of the No. 1 ESS Stored Program

By J. A. HARR, MRS. E. S. HOOVER and R. B. SMITH

(Manuscript received January 22, 1964)

*The stored program of the No. 1 ESS must perform switching functions reliably and promptly. Its design must be economical of memory and execution time, and must accommodate office growth easily. The program is organized so that an interrupt system initiates the input-output programs that must be performed to accurate timing tolerances. The data collected by these input-output programs are passed to the call processing programs which decide what course of action the calls shall follow. The program also assigns an appropriate share of central control time for maintenance and administrative functions. The program is generic in the sense that specific quantities which change from office to office are looked up in tables and are not embedded in the program itself. This article describes the basic program structure and illustrates it with some typical programs.*

## I. INTRODUCTION

The program for the No. 1 electronic switching system (ESS)<sup>1</sup> constitutes the operating intelligence of the system. In fulfilling this function it must meet the stringent operating requirements of a telephone switching office. This article describes the most important of these requirements and outlines the implementation selected as a result.

There are six main classes of requirements. First, the system must respond appropriately in real time to demands for service. Second, the system must perform a large variety of actions to provide the many services which are offered and to work compatibly with a wide range of connecting systems. Third, the system must be extremely reliable. In forty years of continuous operation, the total time the central processor may be out of service is measured in minutes. Fourth, the system must work in wire centers which are growing in the amounts of equipment and in the scope of features offered. Fifth, although there will be a large number of installations which may differ from each other in the kind

and quantities of equipment and in the services which they offer, costs of compiling programs must be kept at a minimum. Finally, the program should be designed to keep system costs low whenever possible, principally the cost of program storage, the cost of call storage, and the cost of the required number of central processors needed for the total market.

### 1.1 *Requirements on Processing Capability*

Telephone customers are used to receiving prompt service whenever they request it. As a result, much of the call processing work performed by a switching system cannot be postponed for long. For example, when a customer answers a ringing telephone, ringing stops in less than a quarter of a second and the two customers may begin conversation.

A particular wire center also receives signals from other wire centers as well as from customers directly. The size and extent of the total investment in switching equipment makes it necessary for new equipment to communicate with older systems. For example, because step-by-step switching trains are driven by the customer's own dial pulses, the step-by-step office will spill digits into another office without checking to see if the receiving office is ready. Therefore, if calls are to be processed correctly, the No. 1 ESS must be ready to receive dial pulses from a step-by-step office in a few hundredths of a second from the time the trunk is seized.

A third source of demand for prompt action comes from the character of the switching system which the program must control. The relay circuitry has been much simplified compared to that in previous common control systems. However, relays are still used both in the network controllers and in trunk, junctor, and service circuits. Simplification has been possible because the program has taken on the job of triggering the operation of relays in proper timing and sequence. In the case of the dial pulse receiver, for example, pulse detection, counting, and memory functions are performed by the program. In order to make sure that the pulses are detected under the worst cases of pulse distortion, the program must look every 11.5 milliseconds for a change in the line current. The higher the repetition rate of scanning, the larger the percentage of time which the central processor must spend in scanning. Because the duration of the scanning program itself varies with the number of pulses detected, there is variation in the actual rate of scanning a given receiver. Study has shown that when a group of receivers is scheduled to be examined every 10 milliseconds, the interval between inspections of any particular receiver will almost never exceed

11.5 milliseconds. Thus, certain functions are performed by the program on fairly tight timing tolerances in order to simplify circuitry in the rest of the switching system. The need to perform a number of tasks with different tolerances is inherent in the nature of the switching function. Failure to organize the program properly would result in inefficient use of the central processor, which in turn would mean a larger investment in data processing equipment.

### 1.2 *Versatility*

A fundamental reason for using a stored program in the No. 1 ESS is the need for versatility. The No. 1 ESS must be compatible with other switching equipment and must provide the ever growing list of special services which the telephone industry offers to its customers. By using a stored program, these objectives can be attained economically. Even though much development work may go into designing programs for new features, the program, once written, can be easily installed. Only the contents of memory must be changed and not a large number of wired connections. Because the program must perform so many functions, it is very large. To keep it manageable, it is divided into functional blocks which are utilized in different ways to perform a large number of complicated tasks. When a new service is needed, much of the required program may be already available. The manner of dividing the program is discussed in the paper on call processing.<sup>2</sup>

### 1.3 *Reliability*

The ability to process calls is the principal function of the central processor and its program. Of almost equal importance is the ability to do so reliably. A complete failure of a central office for even a period of 15 minutes is an event so rare that it generally is reported in newspapers. Yet the No. 1 ESS has so centralized the intelligence of the central office that no call can proceed without the attention of the central processor. Therefore, when a failure or malfunction is detected in the central processor, the faulty unit must be switched out and a working system put together from the duplicated units. The faulty unit must then be diagnosed so that plant personnel may quickly repair it and put it back into service.

About half of the total program instructions are devoted to fault recognition, diagnosis, and routine maintenance. The detection of a fault, the analysis of which unit is faulty, and the actions taken to assemble a working central processor are assigned the highest levels of

priority. Once a working system has been put together, the detailed diagnosis of the faulty unit can proceed at a more leisurely pace than that of normal call processing.

Included within the call processing programs are checks for abnormal inputs and processing errors. For example, a customer ought not to be able to generate more than ten dial pulses before there is a pause while the dial is wound up again. If, however, there is a weak power cross which has been undetected by the power cross test, a longer series of dial pulses may be generated. The input program which counts pulses checks for an excessive count and treats such a call as a partial dial.

#### 1.4 *Ability to Accommodate Growth*

A given wire center will have a unique set of engineered equipment. With growth, the amount of equipment changes. It would be expensive to rewrite the program every time an additional frame of equipment is added to the center. Therefore the basic program is designed to treat all information about quantities of office equipment as data which can be changed as the office grows.

#### 1.5 *Standardization of the Program*

Additional flexibility is required of the program to meet the needs of wire centers with differing features. One wire center needs to communicate with step-by-step switching equipment, another with panel switching equipment, a third offers TOUCH-TONE calling service to its customers, and so on. Producing a tailor-made program for each office would increase programming and compiling costs and entail problems in guaranteeing an error-free program. Therefore, a further requirement is that a single program contain all features needed for a wide range of applications. This standard program, called a *generic* program, may thus be used in many installations.

#### 1.6 *Economic Balance*

The length of the program in program store, the number of call store words, and the number of machine cycles required to perform a task can all be traded off against each other. These quantities have associated costs; for example, the number of machine cycles affects the total amount of processing equipment required to serve a given market. Therefore an over-all requirement is achievement of a sound economic balance in the use of these quantities.

The following pages describe the organization and implementation of the program plan chosen to meet these requirements.

## II. PROGRAM STRUCTURE

The organization of the No. 1 ESS program is strongly influenced by the fact that it must operate in real time. That is, the program must respond promptly to signals and data submitted to it by customers and other switching systems. In addition, it must respond quickly to errors detected by the many trouble detector circuits designed into the hardware to assure dependable operations within the system at all times. Whenever it fails to do so, the result may be improper handling of calls and a general degradation of service. For example, failure to detect digital signals may result in directing a call to a wrong number, or failure to outpulse digits to another office promptly will cause the other office to return overflow tone to the calling customer. Therefore it is necessary to establish a hierarchy of program tasks. Some tasks must be performed on a strict schedule; others may be delayed without significant adverse effects.

### 2.1 *Interrupt System*

The central processor has an interrupt mechanism<sup>3</sup> within it which seizes control of the system momentarily when a manual, trouble detector, or clock signal is received. The interrupt circuit causes the system to stop its present program task, store the program address at which it was interrupted, and then transfer to the appropriate fault-recognition program or clock-controlled input-output program. When the interrupt programs are completed, control is returned to the program that was interrupted.

Fig. 1 illustrates this over-all program plan. The interrupt sources and their associated programs are arranged in a hierarchy of nine interrupt levels; from highest to lowest, these levels are designated A, B, C, D, E, F, G, H, J. An interrupt source assigned to a particular level can interrupt programs of lower levels only. The majority of the programs are subject to interruption by any of the nine levels, and are therefore called *base-level* programs.

The highest interrupt source is the A level, initiated from the master control center, which allows manual selection of operating modes. Interrupt levels B through G are activated by system trouble detectors. These fault-recognition programs are discussed in the article describing maintenance<sup>4</sup> appearing in this issue.

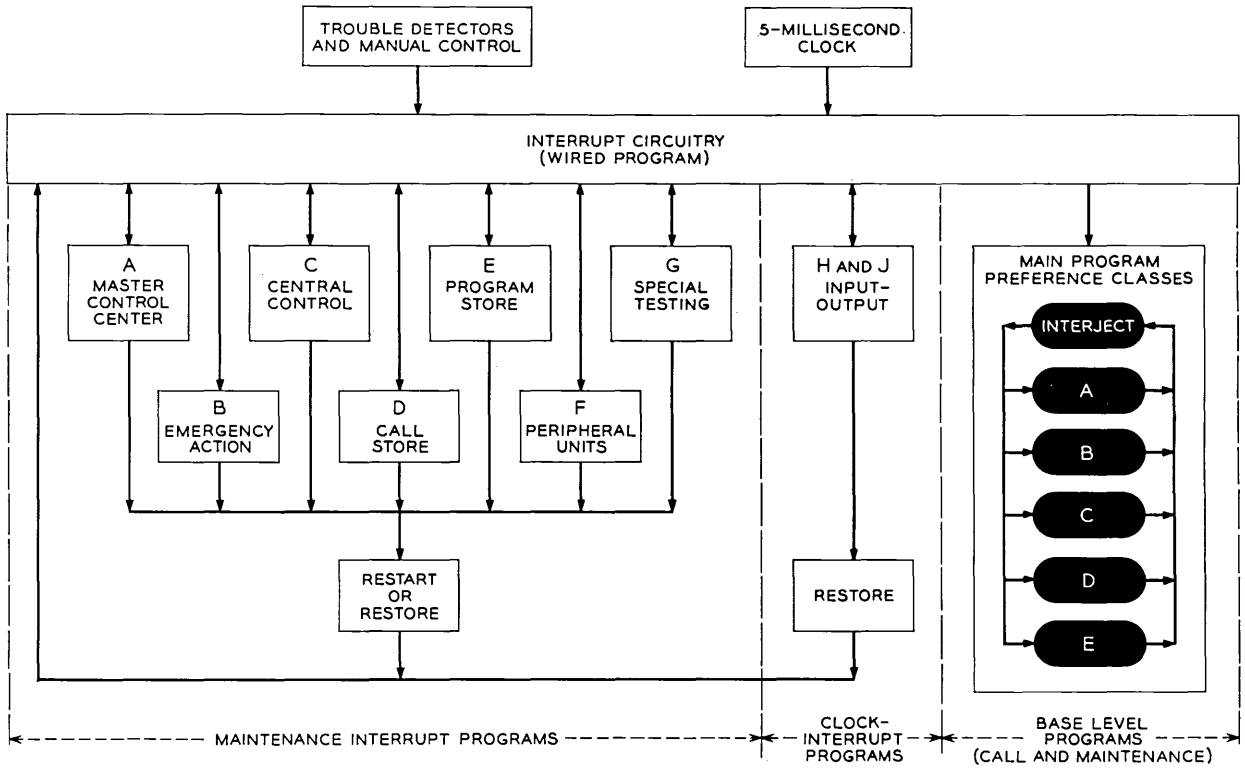


Fig. 1 — Program control plan.

Every 5 milliseconds a system clock activates interrupt level J, which in turn gives control to input-output programs. Level H is used to interrupt the level-J input-output program when tasks being performed exceed 5 milliseconds. The level-J programs are normally in control for about 0.2 to 2 milliseconds, according to the size and traffic of the office.

## 2.2 *Input-Output Buffering*

In order to go to the next task reasonably promptly, the individual tasks must not take too long. In particular, it is necessary to limit the amount of processing performed by interrupt-level programs. For example, the scan of the lead on a TOUCH-TONE receiver that indicates that a digit is present ends by reading the ferroids associated with the individual frequencies and placing the result in an area of call store known as a "hopper." It would be possible to carry the processing of the call further. The area in call store where the other digits dialed on this call are stored could be interrogated to determine whether dialing is finished, and if so, the network path hunt and ensuing actions required to set up ringing could be carried out. However, all this work would consume several milliseconds of continuous processing for a single call. Since the scan for a signal present on a TOUCH-TONE receiver must be performed every 10 milliseconds, an excursion of several milliseconds each time a digit is detected would frequently exceed tolerances. Hence, work on the call is terminated by buffering the digit in a hopper, from which a base-level program will later unload it and carry out the processing just described.

Fig. 2 is a schematic representation of the program control and flow which carries out the sequence of actions needed to process calls, described elsewhere in this issue.<sup>2</sup>

The input-output programs are shown in the upper part of this figure. The input programs are confined to scanning for and recognizing input signals and storing this information in a call store hopper. The hopper stores the input information until the base-level programs inspect it for data. When data are present, appropriate base-level programs, as shown at the bottom of Fig. 2, start or continue the processing of the call. Likewise, call store buffers are provided for the base-level programs to load with output data. At an appropriate time, these data are unloaded by an output program which delivers them to the peripheral equipment. Some examples of the hoppers and buffers used in the system are shown in the middle of Fig. 2. For example, the peripheral order buffer (POB) is used to store address and control data for network controllers and

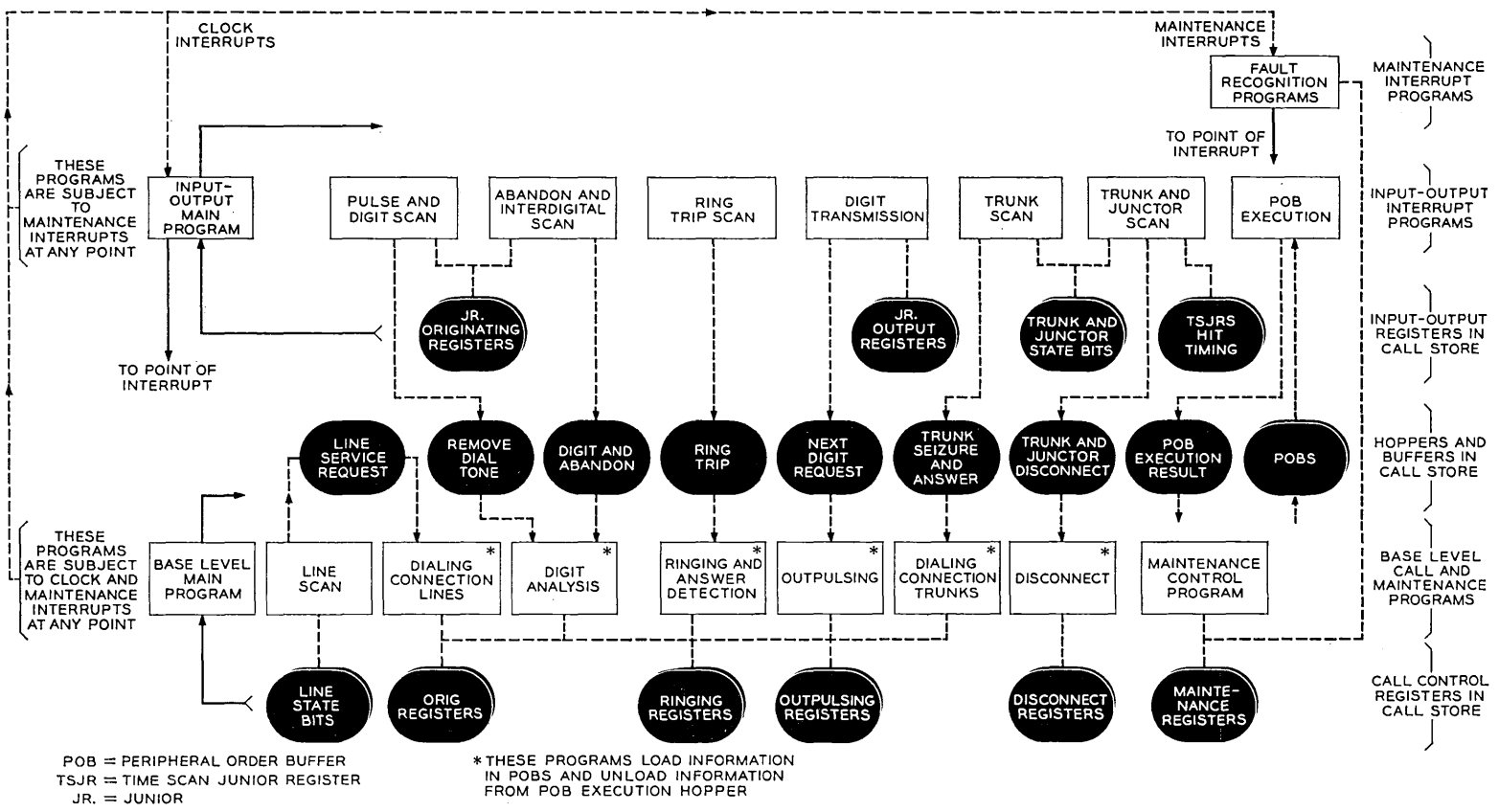


Fig. 2 — Schematic representation of program control and flow.



signal distributors. These buffers provide the means for communication between the scheduled input-output programs and the base-level call processing programs.

### III. PROGRAMS PERFORMED ON THE CLOCK INTERRUPT LEVEL

The 5.5-microsecond clock pulses in the central control are counted, and every 5 (actually 5.005) milliseconds the counting circuit generates an output signal which interrupts the base-level program being performed. The interrupt circuit makes the central processor transfer to the J-level input-output main program. Some input-output tasks are performed every 5 milliseconds; others are performed at multiples of 5 milliseconds up to 120 milliseconds. The input-output programs are classified into high-priority and low-priority tasks, according to the frequency and accuracy with which they must be performed.

The low-priority tasks can be delayed for a few milliseconds without adverse effect on the operation of the system. This indeed will be the case when the coincidence of input work under a peak traffic load causes the system to take more than 5 milliseconds to complete the high- and low-priority tasks. In this event the H-level interrupt will occur and the low-priority work will be interrupted. The accumulated high-priority work will again be performed before returning to the low-priority program that was interrupted.

Accordingly, each input-output program is assigned to either the high- or low-priority timetable. A list of high-priority tasks, the frequency at which they must be executed, and the call store memory words needed for carrying out these tasks are shown in Table I. A similar list of low-priority tasks is shown in Table II.

#### 3.1 *Input-Output Main Program*

To assist in understanding the design of the input-output main program, a description of some additional program attributes is needed.

TABLE I — HIGH-PRIORITY INPUT-OUTPUT TASKS

Task	Frequency	Call Store Memory
(1) Dial pulse and digit scan	10 ms	junior originating registers, remove dial tone and digit hoppers
(2) Abandon and interdigital timeout	120 ms	junior originating registers, digit and permanent signal partial dial hoppers
(3) Twistor card writing	5 ms	twistor word storage register
(4) Call charge magnetic tape output	5 ms	message storage registers

TABLE II — LOW-PRIORITY INPUT-OUTPUT TASKS

Task	Frequency	Call Store Memory
(1) Teletypewriter scan	25 ms	teletypewriter buffer
(2) Peripheral order	25 ms	peripheral order buffer (POB), POB execution hopper
(3) Power cross test scan	*	POB execution hopper
(4) Ringing current test scan	†	POB execution hopper
(5) Detection of outgoing trunk wink	100 ms	outpulsing junior register, next-digit request hopper
(6) Multifrequency outpulsing	25 ms	outpulsing junior register, next-digit request hopper
(7) Disconnect scans	10 ms	timed-scan junior registers, trunk and junctor disconnect hopper
(8) Interrupt sanity test	100 ms	emergency-action register
(9) Interject timing	100 ms	interjected ordered bits buffer

\* This task is performed on three consecutive 5-ms intervals after a power cross scan order is encountered during a POB execution.

† This task is performed on the first and third of the five 5-ms intervals after a ringing current test scan order is encountered during a POB execution.

### 3.1.1 Characteristics of the Input-Output Main Program

Since this program must be executed every 5 milliseconds, the time required by the central control to cycle through all of the input-output task programs is held to a minimum, even at the expense of a small increase in the total number of program words. For example, it is expedient in some cases to have a number of program blocks that perform nearly equal tasks instead of a common program capable of performing all of the tasks, since the common program would in general involve more machine operations to accommodate the small variations in each of the individual tasks.

The program plan is sufficiently flexible that the same program can provide service after changes in the system due to growth. Also, the same program must operate during and after certain changes in the features offered by an office. For example, a feature included in the generic program may require a type of trunk circuit which is not provided in all offices. These circuits may be introduced into any office without changing the program. To meet this growth requirement, the information relating to size, traffic, and features for an office is not embedded in the data field of program instructions, but instead is provided in parameter tables within the program store. In cases where the real time of the system is quite sensitive to the access time for retrieving a particular parameter, it is also stored in the call store, from which it can be read more quickly than from the program store. In particular,

two parameters of this type are the number of input-output tasks and the frequency with which they are performed.

3.1.2 *Organization of the Input-Output Main Program*

A block diagram of the input-output main program is shown in Fig. 3. When the J-level interrupt occurs, control is transferred to the input-output main program, which operates as follows:

(1) It saves the contents of the central control index registers to allow resumption of the base-level program at the point of interruption and sets the H-level inhibit flip-flop.

(2) It updates the time counter and activates, one at a time, all input-output programs that require action according to the high-

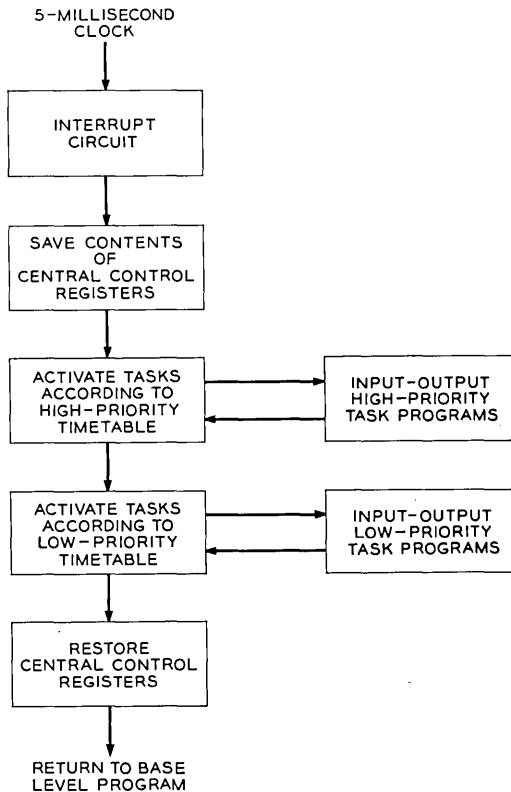


Fig. 3 — Input-output main program.

priority timetable. When all high-priority tasks are completed, it uninhibits the H-level interrupt by resetting the control flip-flop.

(3) It activates, one at a time, all input-output programs that require action according to the low-priority timetable.

(4) It then refills the central control index registers with the information saved in (1) and returns control to the interrupted base-level program.

The principal parts of the input-output main program are the high- and low-priority timetable programs. Since both programs are identical in structure, it is necessary to examine only one of them to understand the operation of the input-output main program.

### 3.1.3 High-Priority Timetable Program

Figs. 4 and 5 show the layout of the call store used to support the high-priority timetable program. The transfer table of Fig. 4 consists of 23 consecutive words. Here and in the following descriptions, symbolic designations will be used — in this case, P0 to P22 — instead of

P 0	DIAL PULSE AND DIGIT SCAN PROGRAM ADDRESS 1
1	DIAL PULSE AND DIGIT SCAN PROGRAM ADDRESS 2
2	
3	
4	
5	
6	ABANDON INTERDIGITAL TIMING SCAN PROGRAM ADDRESS
7	
8	
9	CALL CHARGE PROGRAM ADDRESS
10	
11	
12	
13	
14	
15	TWISTOR CARD WRITING PROGRAM ADDRESS
16	
17	
18	
19	
20	
21	
P 22	ZERO CNT PROGRAM ADDRESS

Fig. 4 — Call store transfer table for high-priority timetable program.

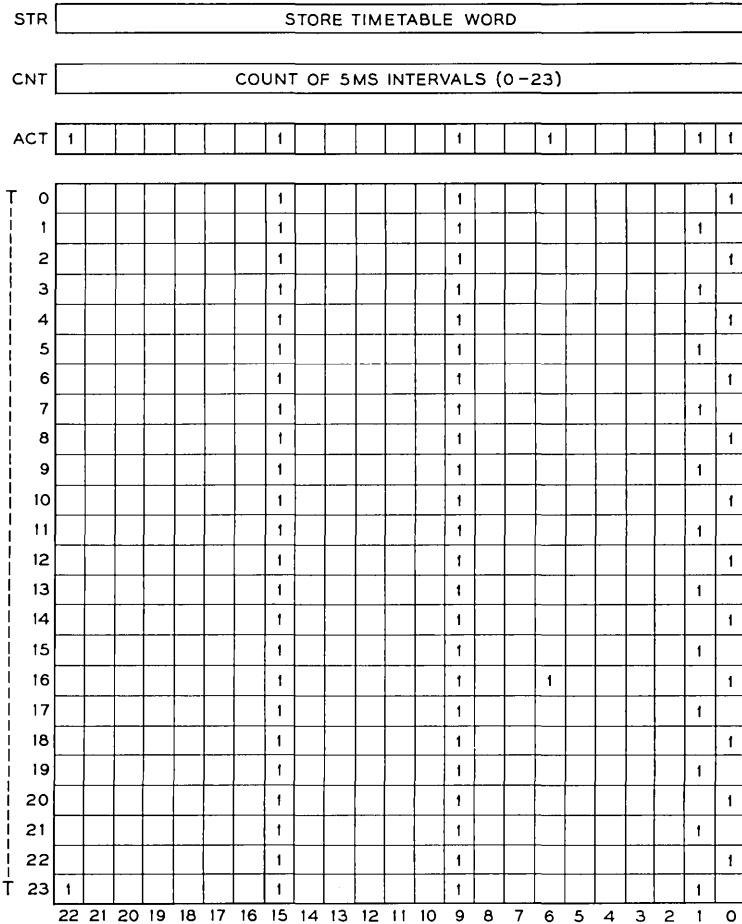


Fig. 5 — Call store timetable for high-priority timetable program.

numerical call store addresses. Each of the 23 words is used to store the starting address of an input-output program assigned to the high-priority timetable. For instance, words P0 and P1 contain the program store addresses of the dial pulse and digit scan programs.

The timetable of Fig. 5 consists of 24 consecutive words designated T0 to T23. Each word is associated with a particular 5-millisecond interval within a 120-millisecond cycle. A count kept in the CNT word is used to identify the 5-millisecond interval and the associated T word in the timetable.

Within each T word, bits 0 to 22 are associated with the input-output

programs whose starting addresses are stored in the corresponding words P0 through P22. Thus each column of the timetable is associated with an input-output program. Each column has an activity bit which is part of the ACT word. If the activity bit of a column is equal to 1, the 1's marked within the column designate the 5-millisecond intervals during which the associated input-output program is due for execution. The table shows that program P0 is activated on all even 5-millisecond intervals and that program P1 is activated on all odd 5-millisecond intervals. Program P15, when active, is due every 5 milliseconds. Program P22 is activated only once every 120 milliseconds. This program recycles the count in CNT from 23 to 0.

The word STR is used to store the pattern of 1's and 0's that indicate which input-output programs are due for execution during an interval. As these programs are executed, one at a time, the corresponding flag bits are reset to 0 until all the programs have been executed.

A step-by-step description of the program shown in Fig. 6 is given in Table III. Assume that the count of 5-millisecond intervals that is kept in CNT is initially 0.

As shown in Fig. 6, to activate the first input-output program requires eight program steps and subsequent ones only five. Note that only six of the possible 23 flag columns are presently used in this high-

LOCATION	OPERATION	ADDRESS OR DATA, REGISTER AND OPTION FIELDS
ENTER	MX	CNT
	MK	TO, XA
	XM	CNT
	PMK	ACT
	TZRFZ	LP
	KM	STR
	MY	PO, F
	T	O, Y → TO INPUT-OUTPUT TASK PROGRAM →
LOOP	MK	STR ← FROM COMPLETED TASK ←
	TZRFZ	LP → TO LOW PRIORITY TIMETABLE PROGRAM →
	KM	STR
	MY	PO, F
	T	O, Y → TO INPUT-OUTPUT TASK PROGRAM →

Fig. 6 — High-priority timetable program.

TABLE III — STEP-BY-STEP DESCRIPTION OF HIGH-PRIORITY TIMETABLE PROGRAM

(1)	MX	CNT	The content of CNT is read into central control (CC) register X. (This value of the interval count is used as a pointer to select one of the 24 timetable entries.)
(2)	MK	T0, XA	The address T0 is indexed with the value of the X register to obtain the address of the timetable entry to be read into CC register K. Then the content of X is increased by 1. (In successive interrupts, this instruction results in reading the timetable entry at address T0+0, T0+1, T0+2, and so on up to T0+23.)
(3)	XM	CNT	The new value of the interval count is stored in CNT.
(4)	PMK	ACT	The timetable entry contained in K is ANDed with the activity bits in ACT; the resulting word is placed in register K. This word contains a 1 in every position in which both the timetable entry and the ACT word contain a 1. Thus, each position marked by a 1 designates a program that is active and due for execution. For instance, when T0 is read, three bits are equal to 1 in columns 0, 9, and 15.
(5)	TZRFZ	LP	If all the bits in K are 0 (if no programs need to be acted on), the program transfers to address LP where the low-priority timetable program starts. If one or more bits of K are equal to 1, the position of the rightmost 1 is stored in register F. The rightmost 1 itself is set to 0, and the program advances to the next step. Clearing the rightmost 1 allows the next 1, if any, to be recognized later as the new rightmost bit of the word in K.
(6)	KM	STR	The new binary word in register K, modified by the removal of the rightmost 1, is stored in STR.
(7)	MY	P0, F	The address P0 is indexed with the value in register F to obtain the transfer table entry to be read into register Y. This is the address of the input-output program corresponding to the rightmost 1, originally in register K.
(8)	T	0, Y	The program transfers unconditionally to the address stored in register Y as the result of step (7). When the input-output program has been completed, a transfer is made back to step (9).
(9)	MK	STR	The word stored in STR is read into register K.
(10)	TZRFZ	LP	If all the bits of the word in K are 0, the program transfers to the low-priority timetable program. If one or more bits are equal to 1, the position of the new rightmost 1 is stored in register F, that 1 is erased, and the program advances to the next step.
(11)	KM	STR	This step performs the same function as step (6); it stores any remaining flag bits in STR.
(12)	MY	P0, F	This step performs the same function as step (7); it obtains the address of the next input-output program to be executed and stores it in Y.
(13)	T	0, Y	A transfer is made to the address in register Y. When the input-output program has been completed, a transfer is made to step (9). Steps (9) to (13) are used over and over until, one by one, the flag bits are removed. The TZRFZ instruction of step (10) will then lead to the low-priority timetable program.

priority timetable. Therefore ample space for future input-output programs is provided. All that has to be done to add programs is to place appropriate time flag bits in any one of the unused columns, mark the column activity bit to 1, and place the corresponding entry address of the new input-output program in the transfer table.

This flexibility has been gained at the cost of only 17 additional call store words in the transfer table. By writing appropriate data into the call store (and a copy in the program store for reliability), an input-output program included within the generic program can be added to an office, and changes can be made in the order and frequency of execution of any input-output program. Furthermore, the input-output main program need not be changed even if a new issue of a generic program requires the addition of new programs.

An example of a specific input-output high-priority task program is given in Section 3.2.

### 3.2 *Dial Pulse and Digit Scan Program*

This program is used to scan the signal-present leads of dial pulse, multifrequency, and TOUCH-TONE receivers for signals every 10 milliseconds. It is activated during every 5-millisecond interval by the high-priority timetable program. However, only half the receivers are scanned in each of the 5-millisecond intervals in order to even out the work load. The functions of this program are as follows:

(1) For dial pulse receivers this program detects pulses by observing a change in the scanner reading from 0 to 1 (off-hook to on-hook condition of the handset). When such a change is found, the program adds one to the pulse count, which is located in a call store area called a "junior register" associated with each receiver. If this is the first pulse received, the address of the originating register in which digits are to be accumulated for this call is read out of the junior register and loaded into the remove dial tone hopper. In case the pulse count overflows (becomes equal to 16) the program stops incrementing the pulse count so that the permanent signal partial dial program will detect a timeout and make an entry in the permanent signal partial dial hopper.

(2) For TOUCH-TONE receivers this program detects that TOUCH-TONE signals are present by observing a change in the signal-present scan point reading from 0 to 1. Upon finding a change, the tone frequency scan points are read and their values together with the address of the originating register serving the call are stored in a TOUCH-TONE digit hopper.



(3) For multifrequency receivers this program detects the presence of multifrequency signals by observing a change in the signal-present scan point value from 0 to 1. The scanner leads associated with each frequency are read, and their values together with the address of the originating register serving the call are stored in the multifrequency digit hopper.

Observe the similarity of the program actions required to detect and report inputs from the three types of receivers. This design permits flexibility in the assignment of scan points for all three types of receiver, since the input program scans a row of scanner points for a change in reading from 0 to 1 without regard to the type of receiver being interrogated.

It is beyond the scope of this paper to discuss the complete dial pulse and digit scan program. However, a description of the core of this program reveals its basic design.

Fig. 7 shows the call store memory layout used by the core program. Each word in the scanner address table (SCA) contains a trunk scanner number of a row of digit receivers. This number addresses a word of 16 scanner outputs, among which at least one is assigned to a receiver.

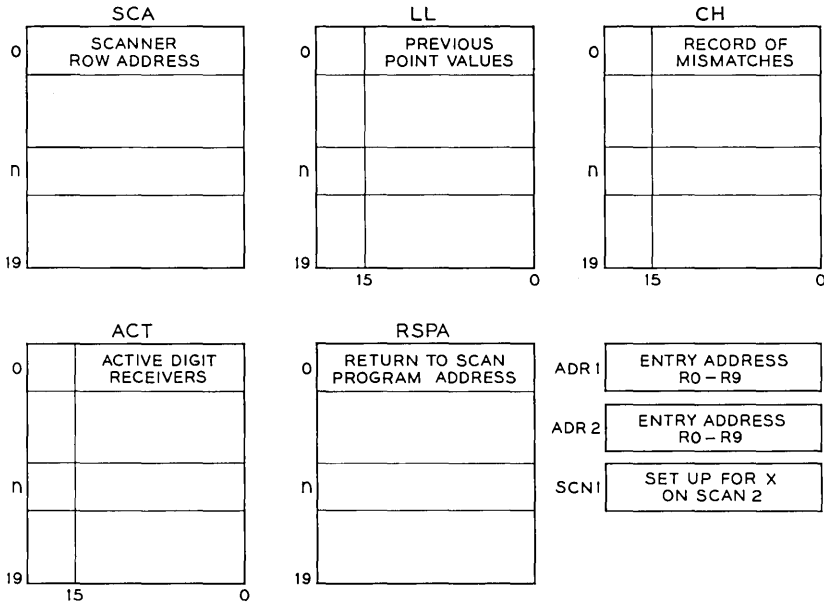


Fig. 7 — Call store memory layout for core of scan program.

The last look table (LL) stores the previous scanner outputs of the row in a memory location corresponding to the location of the row address in the scanner address table.

The change status table (CH) is used to record a mismatch between the present and previous scanner readings when detected. The corresponding change status bit for the scanner is set equal to 1. The abandon and interdigital timing scan uses this information to determine interdigital timeout or abandonment of the call by a customer.

The activity table (ACT) specifies the active receivers within each scanner row and is used by the dial pulse or digit present program to determine the active receivers reporting a change in state from 0 to 1.

The return to scan program address table (RSPA) contains the appropriate return address for use by the dial pulse or digit present program to return control to the core program after the pulse counter(s) is (are) incremented and/or the appropriate hopper(s) is (are) loaded.

Observe that all of the tables described above are blocks of 20 words each. This design was made to accommodate the maximum number of receivers required in a wide range of offices. The symbolic name of the call store address of the first word of each table is SCA, LL, CH, ACT or RSPA, respectively. The binary addresses corresponding to these names will be defined by the compiler-assembler<sup>5</sup> in the fixed area of the call store. That is, these tables have been assigned a fixed location in call store in all central office programs. However, for small offices only a small number of rows in each table will be used to accommodate the receivers for the office. Therefore, the core program is designed to scan only the number of rows required in a given office. Since approximately half the number of rows are to be scanned in each of the 5-millisecond intervals, three other control words are needed by the core program.

At a call store location with the symbolic name ADR1 is stored one of the entry addresses R0 through R9 according to the number of rows to be scanned during the even 5-millisecond intervals. Likewise, at address ADR2 is stored the appropriate entry address for the number of rows scanned during the odd 5-millisecond intervals. At address SCN1 is stored a number one less than the number of rows scanned in the even interval. This number is used in the odd interval scans to set an index register to the proper value for controlling the row words read and stored in memory.

The functions performed by the core program shown in Table IV are:

- (1) Initialize central control registers according to whether the scan occurs in an even or odd interval.
- (2) Read the row of 16 scan points represented by the first scanner

TABLE IV — DIAL PULSE AND DIGIT SCAN CORE PROGRAM

Location	Operation	Variable Fields	Comments
SCAN1	WX	-1	This is the entry point on even 5-ms intervals. Set index register X to -1.
	MSF	SCA+1,X	Send scanner address in first word of SCA table to scanners. Present state of 16 scan points returns to central control register L.
	WZ	DPDP	Store in central control register Z the address (DPDP) of entry point to dial pulse or digit present program.
	MK	CH+1,X	Move content of first CH word into central control register K.
	T	ADR1,M	Transfer program control to the address in call store location ADR1; this will be one of the addresses R0 to R9, depending on the number of rows to be scanned in this office.
SCAN2	MX	SCN1	This is the entry point on odd 5-ms intervals. Set index register to one less the number of rows scanned in the even interval.
	MSF	SCA+1,X	(Same as SCAN1+1)
	WZ	DPDP	(Same as SCAN1+2)
	MK	CH+1,X	(Same as SCAN1+3)
	T	ADR2,M	(Same as SCAN1+4, except that call store address ADR2 is used instead of ADR1.)
R0	UMKMJ	LL+1,X	R0 is the entry point from SCAN1+4 or SCAN2+4 if ten rows are to be scanned. Match (exclusive-OR) present scan point values in register L with previous scan point values read from corresponding word in LL table. Store in central control register J the 16-bit mismatch word (1's at bit positions of mismatch), and OR it with the CH word in register K, storing the result in K.
	LM	LL+1,X	Move present scan point values from register L into the LL table.
	KM	CH+1,XA	Move into table CH from K the new CH word, which now contains 1's in bit positions corresponding to scan points whose state changed since the last abandon-interdigital scan. Also add one to register X.
	JKMSF	SCA+1,X,PL	AND the mismatch word in J with the present scan point word in L and store in K this logical product, which now contains 1's only in the bit positions where a 0-to-1 change occurred in the corresponding scan points. Set a decision-control flip-flop, to be used by the following order, according to whether the product in K is zero or not. Also read the next scanner address from the SCA table and send it to the scanners. The scan point values will return to L in two machine cycles.
	TAUMK	CH+1,X	If the product in K is nonzero, transfer program control to the address in central control register Z. If zero, move into K the contents of the next word in CH table.

The next 45 steps essentially repeat the preceding five instructions nine more times to take care of a possible ten rows of scan points to be examined. The last few instructions in the tenth set of five are modified slightly because it is not necessary to prepare for a next row in that case. A concluding instruction transfers control back to address LOOP in the high-priority timetable program (Fig. 6).

row address in the SCA table if it lies in the even interval, or scanner row address one greater than the number stored in SCN1 if it lies in the odd interval.

(3) Match the 16 new scan point values with the previous scan point values stored in the corresponding row of 16 LL bits.

(4) Update the corresponding row of 16 CH bits. That is, write 1's into the CH word wherever a mismatch (change in scan point state) occurred.

(5) Update the row of LL bits. That is, read present scan point values to the corresponding row in the LL table.

(6) Determine those scan points whose value changed from 0 to 1.

(7) Transfer to the dial pulse or digit-present program if there is at least one change from 0 to 1.

(8) Repeat steps (2) through (7) for the remaining scanner rows to be examined during this 5-millisecond interval.

(9) Return program control to the high-priority timetable program.

Because of the frequency of use of this core program, an economic balance clearly required an emphasis on machine cycle minimization at some cost in total memory. Two means were used to achieve this real time efficiency. First, three special instructions (UMKMJ, JKMSF, and TAUMK) were designed, each of which accomplishes functions that would require two or three general instructions. Second, the five-word core program is repeated ten times in the program store to avoid the additional time for executing loop control orders and transferring. The program is generic, in that it is designed to handle traffic over the range of office sizes without appreciable loss of efficiency. This flexibility is attained in exchange for a moderate increase in the total call store and program store words used.

#### IV. BASE-LEVEL PROGRAMS

The bulk of No. 1 ESS programs, both call processing and maintenance, are executed on the base level — that is, with none of the interrupts A through J in effect. An appreciable amount of time is spent in the J level, as described above, looking for inputs and disposing of outputs, but this time is consumed in repeated execution of a relatively small number of fairly short programs. The complex work called for by the enormous variety of call situations and equipment configurations and possible malfunctions is carried out by a correspondingly large number of programs. These are executed as required, when time permits, and while no interrupts are in effect. All base-level programs

can be deferred to some extent, but the amount of delay they can tolerate varies widely. It is for this reason that a preference system and a program organization to implement it are required within the base level.

#### 4.1 *Main Program*

This plan and its associated programs are referred to as the base-level main program, or simply the "main program." The question of precisely which associated programs should be considered a part of the main program is a rather arbitrary matter of classification. The fact is that the entire collection of base-level programs becomes in a real sense a *single* program, though subdivisions are useful for various purposes such as functional design and explanation, assignment of work to individual programmers, convenience of assembly and testing, and so forth.

Maintenance programs on the base level, as well as call processing programs, receive control from the same main program. They follow the same general rules and share many devices and techniques that are discussed below. However, since the structure of the maintenance programs is described in detail elsewhere,<sup>4</sup> details and examples in this paper are drawn primarily from the processing of telephone traffic.

Just as the bulk of No. 1 ESS programs are base-level, so the bulk of base-level programs are "task" programs. These are simply the programs at the end of the line of control that perform the ultimate work of the office. Most do a particular kind of work on a single call at a time; others perform various administrative functions, such as interpreting a teletypewriter input message. They differ, too, in the manner in which they receive control from the main program. Some receive it directly from a single dispenser program, while others require a series of dispensers, each triggering the next.

Within the main program complex are distinguished several kinds of "dispenser" programs. These form the links between the basic schedule of the main program and the task programs. One of the principal jobs of the dispenser programs is unpacking the input data that the H and J interrupt-level programs have buffered and distributing them to the task programs for analysis and use. Another important dispenser function is providing timed entries to programs requiring them.

All base-level work is divided into six classes. The highest priority class, called "interject," is described below. The other five are the classes A, B, C, D, and E, in descending order of frequency of examina-

tion. Specifically, the main program delivers control to these classes according to the pattern

ABACABADABACABAEABACABADABACAB

repeated endlessly. Thus class A is examined most frequently, and tasks assigned to that class will suffer shorter delays on the average than those in B, and so on. But if, for example, class E work is about to be done, the existence of waiting class A work cannot change the scheduled order of execution.

Each class consists simply of a number of dispenser programs that are executed in a fixed order, so that if  $a_1, a_2, \dots, a_5$  represent the five dispenser programs in class A,  $b_1, b_2, \dots, b_5$  the five in class B, and  $c_1, c_2, \dots, c_6$  the six in class C, the sequence of execution stated above can be expanded to:

$$a_1a_2a_3a_4a_5b_1b_2b_3b_4b_5a_1a_2a_3a_4a_5c_1c_2c_3c_4c_5c_6a_1a_2 \dots$$

#### 4.2 Dispenser Programs

As an illustration, class B consists of the dispenser programs that administer work from the following five buffers:

class B ordered bits buffer  
 ring trip hopper  
 remove dial tone hopper  
 dial pulse and abandon hopper  
 trunk and junctor disconnect hopper.

“Buffer” is used as the general term for any memory used to store or record work to be carried further at a later time; a “hopper” is a buffer used to accumulate inputs from peripheral equipment.

The ring trip and remove dial tone hoppers are typical single-purpose hoppers. The nature of the input, in each case, is known automatically by the interrupt-level program that detects it, so it is just as easy for it to be loaded into its own hopper, and more efficient for the unloading dispenser program. The remove dial tone hopper is shown in Fig. 8. The “pointer block” of four words contains both a loading and an unloading address, so that first-in, first-out service can be given. Also, it contains the addresses of the beginning and end of the actual hopper, so that the loading and unloading programs are completely independent of the hopper’s size and location. The information that is entered in this hopper requires only one word, and consists merely of the call store address of the originating register whose call is ready to have dial tone

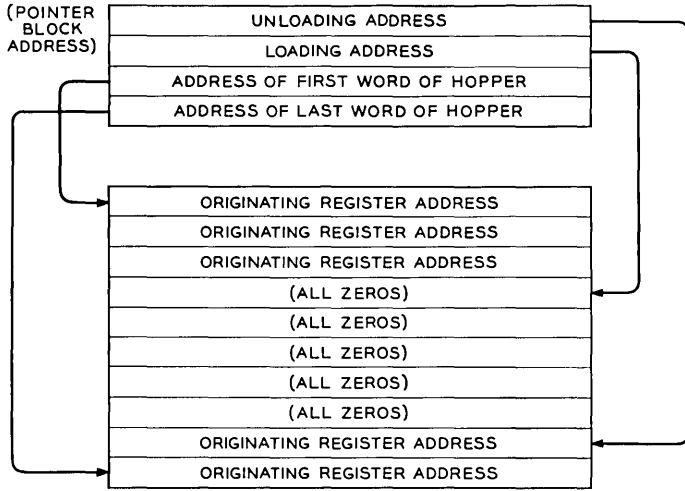


Fig. 8 — Release dial tone hopper.

removed. (A request to remove dial tone is made after detection of the first pulse, so that it will actually be removed by the time the first digit is dialed.) Many hoppers are of this same form, except for different kinds and quantities of information in each entry; some require two words per entry.

Once a dispenser program receives control, it empties its buffer of all work it finds there. Thus, if the remove dial tone dispenser finds the five entries shown in Fig. 8, it will pass control five times to the same task program, each time with central control index register X containing one of the originating register call store addresses unloaded from the hopper.

The dial pulse digit and abandon hopper (whose entry format is shown in Fig. 9) has a dual purpose. An interrupt scan program detects both types of input by the same timing; the only difference between

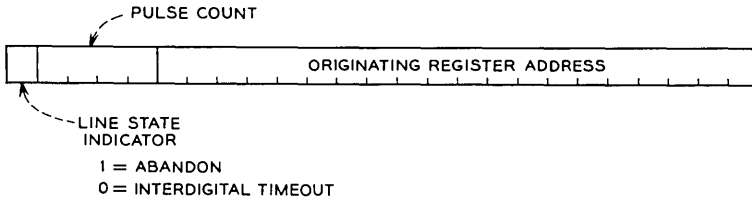


Fig. 9 — Format of entry in dial pulse digit and abandon hopper.

them is that the line is off-hook if a digit has been completed, but on-hook if the call has been abandoned. It is more efficient, therefore, for the interrupt-level program to put both types into the same hopper, with one bit of the entry indicating on-hook or off-hook, than to make the decision itself and place them in separate hoppers.

Each priority class has an ordered bits buffer; the one for class B is shown in Fig. 10. Such a buffer consists of a call store word, each bit of which serves as a flag for its associated program. The address of this program is in an accompanying table in a position corresponding to the flag's bit position in the word. Thus, in Fig. 10, the word BOBB contains two 1's. The rightmost 1, in bit position 3 (starting with 0 on the right), indicates there is work to be done by the program whose address is in word number 3 of the table BP0. This program, as seen from the

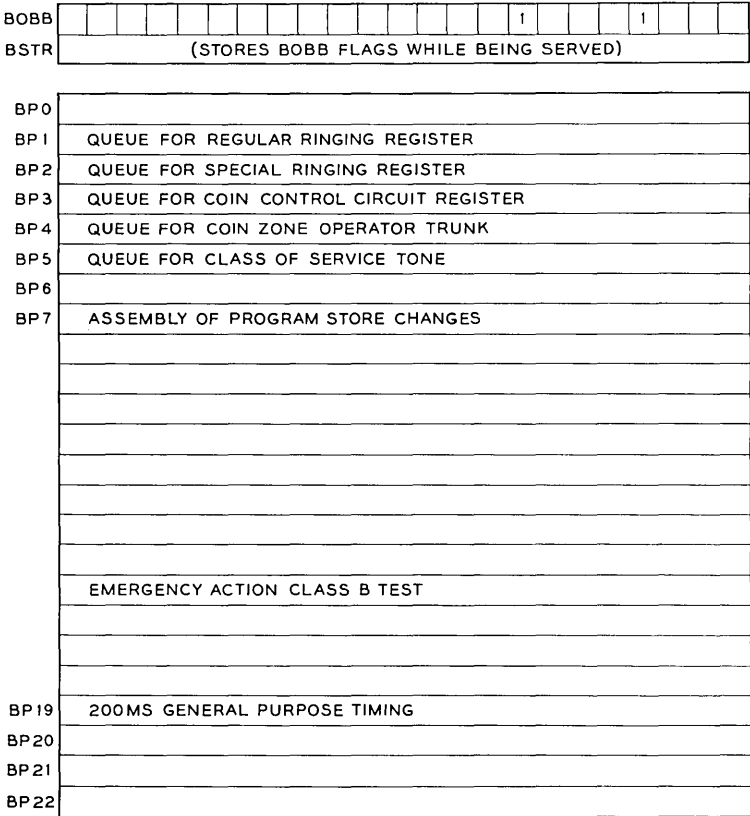


Fig. 10 — Class B ordered bits buffer.



figure, serves the coin control circuit register queue, so the flag would have been set to 1 by some program that needed such a call store register but found none available. At the same time, it would have left waiting on the corresponding queue whatever call register (perhaps an originating register) was already associated with the call. The program triggered by this flag from the ordered bits buffer will check to see if any coin control circuit registers have become available; if so, the one or more calls represented by registers on the queue will be served.

The other 1 in word BOBB of the ordered bits buffer, in bit 7, indicates that there are program store changes to be assembled preparatory to writing new cards for the translation area of the program store. This does not occur for days or weeks at a time, and the flag would have been set to 1 by a special teletypewriter message requesting this action.

The ordered bits buffer dispenser program begins by copying the contents of BOBB into the following word BSTR and zeroing BOBB. Then new flags can be set in BOBB even by interrupt-level programs while the 1's in BSTR are being served and erased from right to left, using the special TZRFZ instruction. (See Table III, step 5.)

#### 4.3 *Interject Work*

The sixth class of base-level work is the interject class; it is not regularly scheduled at all in the ABACA . . . sequence. It has the highest priority of all, however, for a check is made after each task program in each of the other five classes for the existence of interject work, and if found it is done immediately. It is initiated by interrupt-level programs when they encounter work that cannot tolerate the delay it might suffer if put even in a class A buffer, but which should not be done in the interrupt program itself. The latter constraint might be that it is too lengthy for an interrupt level; or that it might interact harmfully with the interrupted base-level program, if both should try to change the contents of the same call store location. Therefore, it is scheduled to be done at the first natural break, by being interjected between task programs. The check for the presence of interject work is accomplished automatically as each task program returns control to the proper dispenser program by transferring to the standard program address "RETURN." At this point, the pair of instructions

MK	RETN	(move contents of "RETN" to index register K)
TKP	0, K	(if K's sign bit is positive, transfer to address in rest of K)

occur. "RETN" is the symbolic address of a call store word containing the dispenser program address to which control should be returned by each of its task programs; the first thing a task dispenser program does upon gaining control initially is to place this address in RETN. The sign bit of RETN is normally positive, but is made negative as a flag by any interrupt-level program wishing to interject a job. Thus the first two instructions of the program at RETURN read this call store word RETN and transfer to the address it contains, unless the sign bit has been made negative. In that case, program control passes to the succeeding orders, which form the interject control program. This program has certain special bookkeeping functions to perform, but essentially it transfers control to the one or more interjected jobs and then returns control to the dispenser address that was in RETN.

The interject priority class consists of a single dispenser program, the interject ordered bits buffer. At present only three kinds of jobs are definitely planned for this buffer, though more can be added if future requirements dictate. One is an emergency-action interject test, and another is the unloading of the hopper containing highest-priority reports from the network execution program. The third is the updating and distribution of information from the 100-millisecond timetable. For the first two, flags are set in the interject ordered bits buffer (and the master flag in RETN) when the relevant interrupt-level programs encounter the need either for the emergency test or the highest-priority network report. The flag for the 100-millisecond timetable is set by the 5-millisecond input-output low-priority timetable every twentieth entry.

The interject 100-millisecond timetable is central to all base-level timing. It is shown in Fig. 11, and will be described only briefly, because it is similar to the high-priority timetable for the H and J interrupt levels which has already been discussed in some detail in Section 3.1.3.

The supervisory line scan should be made every 100 milliseconds, nominally. There is no advantage in scanning excessively often, even in slack periods; routine maintenance checks and various administrative programs can usually make better use of the time. To prevent excessive scanning the line scanning program receives control from the class D ordered bits buffer when its flag bit is found to be 1. That flag is set once every 100 milliseconds from the interject 100-millisecond timetable. On the other hand, an extension of the 100-millisecond period, even if long enough to be noticeable, carries no serious penalty; in fact, since line scanning does take an appreciable amount of the processor's time, it is desirable that during overload proportionately less time be expended looking for work and more spent on work already in progress.

KCNT	(COUNT OF 100 MS INTERVALS, 0-9)																			
KSTR	(STORES LOGICAL PRODUCT OF KT WORD AND KACT WORD)																			
KACT	1	1	1	1	1															
KT 0			1	1	1					1										
					1					1										
	1			1	1					1										
					1					1										
				1	1					1										
			1		1					1										
				1	1					1										
	1				1					1										
KT 9	1				1					1										
KP 0																				
KP 1																				
KP 2																				
KP 18	SET FLAG IN CLASS D FOR 100MS LINE SCAN																			
KP 19	SET FLAG IN CLASS B FOR 200MS GENERAL PURPOSE TIMING																			
KP 20	SET FLAG IN CLASS C FOR 500MS GENERAL PURPOSE TIMING																			
KP 21	500MS PERIODIC TIMING																			
KP 22	RECYCLE AND ONE SECOND PROGRAM																			

Fig. 11 — Interject 100-millisecond timetable.

The periods between executions of class D programs may exceed 100 milliseconds during periods of heavy traffic, and when they do the class D supervisory line scan flag will be set to 1 more often than it is served and made 0.

In column 19, 1's alternate with 0's, so the program whose address is

in row 19 of table KP0 will be entered every 200 milliseconds. All this program does is set a flag in the class B ordered bits buffer for 200-millisecond general-purpose timing, which will be discussed later.

All the programs entered from this table KP0 are, of course, executed as interject jobs, and as such must be kept short lest they themselves delay other interject work. The program whose address is in row 21, entered every 500 milliseconds, updates another timetable which contains 24 rows and thus is cycled through completely every 12 seconds. This timetable, however, has no associated table of transfer addresses; instead, after selecting the current row and ANDing it with the activity bit word, it ORs the result into the class C ordered bits buffer. (This divides the latter's bit positions into two kinds: those set by the 500-millisecond interject timetable; and those set by other means, like those of the class B ordered bits buffer already discussed. The ones set by other means render the corresponding columns of the 500-millisecond timetable useless, of course.)

#### 4.4 *Timing*

As explained in earlier sections, the detection and discrimination of input signals is assigned to interrupt-level programs, and the resultant work of processing them to the base level. Somewhat analogously, time — in the form of a signal every 100 milliseconds — is an input to the base level, where it is detected and analyzed by interject programs, and the work that is found due is passed on to the lower base-level classes. When the intervals being timed become so long that the delays in performing base-level work are negligible by comparison, some of the timekeeping itself can be removed from the interject level. Thus class C has a 3-second timetable, and class E, 15-second and 15-minute timetables.

There remains an assortment of timing requirements not fulfilled by the timetables. Typically, certain kinds of call register programs may reach a point where a delay of a few hundred milliseconds or a number of seconds is required. One example is a TOUCH-TONE test trunk program, which must send out a failure signal 15 seconds after initiation of the test if a correct sequence of signals has not been keyed in by that time. General-purpose timing for such uses is provided by timing linked lists.

A hypothetical one-way 200-millisecond linked list is shown in use in Fig. 12. The only call store memory required for this list when not being used is the single word TIM200, sometimes called the "head cell"

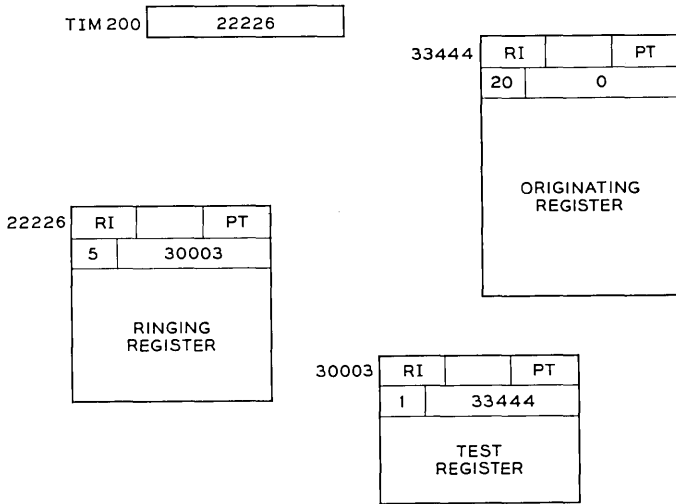


Fig. 12 — One-way 200-millisecond linked list for general-purpose timing.

of the linked list. When not in use, this word contains zero. In the figure, the ringing register was the last to be added of the three registers undergoing timing. The program which added it to the list found TIM200 containing 30003, the address of the test register. The ringing register was inserted in the list by having the "30003" stored in its second word, along with the "5" which specifies the number of 200-millisecond units of timing that are desired. Its own address, 22226, becomes the new contents of TIM200. The timing for whatever registers have been put on the list is administered by a task dispenser program given control every 200 milliseconds from the class B ordered bits buffer (Fig. 10). This program goes through the list, subtracting one from the count in the left part of the second word of each register. When the count is reduced to zero (as it will be next time for the test register, whose count is shown as "1"), the register is removed from the list (by replacing 30003 in the ringing register by 33444) and control is given to the appropriate task register program. The procedure by which this appropriate program is determined is used not only here, but in other situations in which a general program reports to a variety of other programs through their associated registers. Instead of a full program store address being stored in the call register, the RI and PT items in the register's first word are used. The RI ("register identification") selects a table of transfer addresses associated with the type of register,

and the PT ("program tag") is used to select from this table the address of the proper task program. (In this case, of course, that will be whatever program has been designed to take appropriate action when the test register times out.) This selection from the table, however, is made by altering the PT number slightly before using it as an index. This same alteration is performed by all programs reporting a certain class of information — including timing notification — to any call register. The report by some other program of a different class of information — e.g., a scan point change — would not make this alteration, and therefore could use the same PT number to select from the same table a different program. This means of distinguishing inputs eliminates many decisions within the task programs.

Note that to remove a register from a one-way linked list at an arbitrary time between scheduled inspections requires tracing through the list from the beginning. Since this could consume considerable time for a long list, two-way timing linked lists are also provided. Their use is quite similar, but each register must supply two words for the linking, one pointing forward to the next linked register and the other pointing back to the preceding. This means that any register in the chain can be directly dropped out, since it contains enough information for the linkage to be mended.

Most programs requiring general-purpose timing have an associated register to link; for those that do not, a common pool of timing registers is provided.

#### 4.5 *Task Programs*

The variety of task programs precludes a comprehensive treatment within this paper, but at least some of this variety may be indicated. The word "program," when used to designate a subdivision of some larger program, is a static term referring to a set of instructions capable of performing a certain action or group of related actions when some or all of these instructions are executed in sequence. When control is given to such a program at one of its entry points, however, the sequential execution that follows typically runs through only part of the orders in that particular program, and then on into parts of other programs. It is this dynamic unit, consisting of the complete set of instructions executed in unbroken sequence, that is most significant in understanding the flow of control governed by the main program. "Task program" is often used in this dynamic sense, but "task execution" might be a less ambiguous term for the complete sequence of order executions from the

time a dispenser relinquishes control until control is returned to that dispenser. Thus, during a task execution only task programs are executed (apart from possible insertions of interrupt programs), but many and varied task programs may contribute to a single task execution.

There can be no "real-time gaps" within a task execution; such a gap necessarily signals the end of a task execution, and the suspended call or other function can regain control only through the main program mechanism. Task executions differ greatly in length, of course, depending on the amount of continuous work it is possible and desirable to do. As explained in earlier sections of this paper and in the paper on call processing,<sup>2</sup> the work that is done for one call at one time tends to be small, due to circuit limitations and timing requirements as well as the natural fragmentation of customer actions.

Task executions vary also in the number of different "static" task programs they involve. One well known but important programming technique that tends to use many programs in one task execution is the use of subroutines. In the No. 1 ESS program these range from short programs performing simple conversions of equipment addresses from one form to another, to complex network and translation routines which themselves use several levels of subroutines internally.

A subroutine, as the term is generally used, returns control to its client program when it has completed its work. Often, however, two or more task programs arrive at a point where the remaining work to be done in both or all cases is identical, and they can be joined at that point. There is no need to preserve the identity of the source merely in order to return control properly at the end of the task execution, because the standard transfer to RETURN, explained earlier, solves that problem.

The supervisory line scan is an example of a self-contained task program. It could be considered, of course, as two programs: one that repetitively compares the line scanner points with the line state bits, and another program or subroutine to which control is transferred when a proper match of these readings occurs that makes the entries in the service request hopper. If these two closely related programs are combined, however, they become a self-sufficient whole.

A contrasting example is provided by the task execution that begins with a digit just unloaded from a digit hopper. The initial task program is a part of the general call processing functional subdivision called "digit analysis, lines." It begins with the assumption that the K register contains the new digit in bits 21 through 18 (with the digit "0" registered in binary as "1010," corresponding to the ten pulses generated by a dial

telephone), and that the X register contains the address of the originating register associated with the call. This task program could have received control from either of two task dispenser programs, depending on whether the digit was produced by a dial or TOUCH-TONE telephone set. In handling the digit up to this point, the actions of the two task dispensers differ. The dialed digit task dispenser distinguishes the digit entry from the abandoned call entry it may also find in its hopper; the TOUCH-TONE digit task dispenser converts its hopper entry from a code indicating two active tone generators into a binary digit. From this point on, however, the distinction vanishes, and subsequent programs treat the digit in the same way regardless of its original form.

The initial decisions depend on the contents of the originating register whose address is in the X register. This register is a block of 16 consecutive call store words, assigned to a call as soon as the originating line is connected to a receiver and given dial tone, and retaining information for that call until a ringing connection is set up (for a local call) or until outpulsing is completed (for an outgoing call). The initial arrangement of information in the register is shown in simplified form in Fig. 13. (The type of information stored in some words or parts of words varies with the type of call, and can change as the call progresses.) The program first checks the single bit END in word 6 to see if dialing has already been interpreted as completed; if so (an unlikely event) the extra digit is ignored and control is immediately returned to the dispenser program by a transfer to RETURN. Assuming that dialing has not been previously completed, the program adds one to the digit counter (item DC in word 12), stores the new digit in its proper digit slot (DS1-DS10) as determined by the new value of the digit counter, and then checks to see whether that new value matches the number in item DCA in word 6. This number has been put there at some earlier point in the call to tell the present program whether additional decisions must now be made or whether nothing is required beyond the digit storage that has just been accomplished. If DC is not equal to DCA, control is returned to the dispenser program. If DC equals DCA, control is given to the program determined by item AADD in word 6. (The common program device used here to reduce item AADD from a full 18-bit call store address size is to transfer into a fixed table of transfer instructions, using the value of AADD to select the point in the table to which control is transferred.) The program receiving control could be any one of about thirty. (Since AADD is eight bits, the maximum table size is 256, of course.) For example, it might be the program analyzing the first digit, in which case DCA would have been set to



0		RI				PT
1	QUEUE WORD					
2	LINK WORD					
3	SCAN WORD					
4	PATH MEMORY					
5						
6	END	STATE OF CALL BITS		AADD		DCA
7		DS9	DS10	DS1	DS2	DS3
8		DS8	DS4	DS5	DS6	DS7
9	LINE EQUIPMENT					
10	NUMBER TRANSLATION OUTPUT					
11						
12	SCAN POINT IDENTIFICATION					DC
13	CONVERTED DIRECTORY NO. OF ORIG. LINE					
14						
15	OUTPULSING DATA					

RI = REGISTER IDENTIFIER  
 PT = PROGRAM TAG  
 END = END OF DIALING  
 AADD = ACTION ADDRESS  
 DCA = DIGIT COUNT LIMIT  
 DS1 = DIGIT SLOT 1 (FIRST DIGIT)  
   |  
   |  
 DS10 = DIGIT SLOT 10 (TENTH DIGIT)  
   |  
   |  
 DC = DIGIT COUNTER

Fig. 13 — Simplified layout of originating register.

"1" by the program that connected the line to a receiver and dial tone. The analysis of the first digit involves checking to see whether it is a "1", "0", or the eleventh button on a TOUCH-TONE set, since any of these can have special significance. In an office without the "0" prefix option, an initial "0" necessarily completes dialing and requires a connection to an operator. In this case the task execution continues through subroutines in the translation and network area, transferring to RETURN only when it has made its first request for network action by placing proper entries in a peripheral output buffer. The foregoing assumes that no difficulties are encountered. If all operator trunks are busy, a request for a connection to overflow tone will be made instead; or, if no peripheral output buffer is available, the originating register will be left waiting in the corresponding queue before control is relinquished.

Referring again to the possible actions following a match of DC and

DCA, the program to which index AADD points might be one involved in providing a special service such as temporary transfer to a customer. The temporary transfer program would be controlling the call in this case, and would have seized an originating register to serve as a digit collector, placing the desired number of digits in DCA and a number in AADD that would lead to the appropriate temporary transfer task program.

Thus this digit analysis task execution may complete its work in a few order cycles, or it may branch into a multitude of different task programs before its sequence of instructions finally reaches a transfer to RETURN.

#### V. THE GENERIC PROGRAM

One of the important basic requirements stated earlier for the No. 1 ESS program is that it be generic — that is, that a single program be able to serve many offices with different features and characteristics, and during continuing growth. If the program does not change, then data to which it refers must change to take account of the different features, numbers of lines and trunks, types of signaling, and other variables. From a theoretic viewpoint, this concentration of all changeable information into one place instead of scattering it through the data and address fields of many instructions may seem a trifling difference. However, when a large number of offices are involved such an arrangement has substantial economic advantages in compiling programs, adding features, and making changes.

The constraint this places on the program design can be stated quite simply as the inability of the program to know anything that can change. Any quantity or address or option that can vary, as an office grows or from one office to another, must be looked up by the program in a concentrated data area of the program store. The call store may be used where faster reference is needed, but this may be considered an indirect reference to the program store, which must back up all long-term call store information. (In particular, when the office is first put into operation, the call store must be written to its proper initial state from program store data.)

The effect the generic program requirement has on memory allocation is essentially that, apart from the generic program itself, program store and call store are each divided into two parts. One area is of fixed size and fixed layout (though not fixed contents, of course), and the other

can be expanded and rearranged. A good call store example is provided by the hopper shown in Fig. 8. The four-word pointer block must be in the fixed layout part of call store, so the loading and unloading programs can refer to PBA, PBA+1, etc., in the address fields of their instructions. The hopper itself, however, could not be included in this area unless it were made big enough for the largest one needed in any office; but it is economical to reduce its size in smaller offices. The storage in the pointer block of the beginning and end addresses of the hopper itself allows the latter to grow or shrink as required in the expandable part of call store.

Similarly, call registers such as the originating and ringing registers are assigned in expandable call store, and while idle are chained together in linked lists, one for each register type. Only the two-word "head cell" of each linked list, containing the addresses of the first and last registers, is assigned in the fixed layout part of call store.

Nearly all call processing programs use such linked lists as their sole reference in seizing and releasing registers as needed. In program store, however, there must be a record in some form of the absolute addresses of all registers of each type. The call store initialization program, used when the office is turned on initially, must have such information; a few other programs also find it useful. This information is packed quite compactly, since reference to it is made only infrequently. Even so, the amount of program store space required for recording the locations of all registers in the biggest office is more than can be afforded in a small office. Hence the same division into fixed layout and expandable memory is made within the program store data area. A single word for each call register type is assigned at a certain address in the fixed layout portion. This word contains the address in the expandable part of program store of a variable-size block containing the locations of all registers of that type.

## VI. SUMMARY

The reliable and prompt switching functions required of No. 1 ESS put stringent requirements on the program organization. The types of equipment with which the program must work and the services that must be provided are large in number. For a wide range of office sizes and traffic volumes, economy must be stressed, both for the initial installation and for growth. The program organization outlined was chosen because it satisfies these requirements. Certainly other means of implementation are possible.

The interrupt system helps to guarantee prompt attention to tasks that require it. To satisfy the input-output tolerances, it is also necessary to limit the time during which a given task is continuously processed. Most call programs have natural break points, and others must be created to be compatible with the equipment. Lengthy maintenance and administration programs are arbitrarily divided into segments whose execution time does not exceed a suitable limit.

This fragmentation of work, in turn, requires a system of call store registers and buffers in which to store data until processing can be resumed. These and other memory units are designed for ease of growth and economy throughout the range of office sizes.

Both the input-output and base level main programs follow simple schedules which check for work to be done according to its urgency, without consuming an inordinate amount of time.

To increase the efficiency of the very frequently executed input-output programs, special orders were devised, and certain groups of these orders are repeated in program store to avoid the extra transfer time that program loops would cost. For the greater part of the program, however, a very general order structure is used, which should be equally well suited to functions not yet conceived. For this majority of programs, furthermore, subroutines are heavily used to reduce program storage space. This organization also makes it easier to add new features to the program, since use can be made of existing program blocks and subroutines.

However, a major aim has been to minimize the causes that require an addition or any other change to the program itself. This is done by effecting a separation of the program proper from the parameters associated with the features of a particular office and the variables of growth.

#### VII. ACKNOWLEDGMENTS

The program organization of No. 1 ESS is the product of many contributors, from the areas of system engineering and design as well as call and maintenance programming. The choice of program techniques was closely coordinated with the analysis of the actions required for call processing, and depended also on the instructions, both general and special purpose, provided by the central control design. In particular, we wish to acknowledge the contributions of S. Silber, who provided many of the basic ideas for the main programs and is responsible for their actual design and programming.

## REFERENCES

1. Keister, W., Ketchledge, R. W., and Vaughan, H. E., No. 1 ESS System Organization and Objectives, B.S.T.J., this issue, p. 1831.
2. Drew, G. G., Carbaugh, D. H., Ghiron, H., and Hoover, Mrs. E. S., No. 1 ESS Call Processing, B.S.T.J., this issue, Part 2.
3. Harr, J. A., Taylor, F. F., and Ulrich, W., Organization of No. 1 ESS Central Processor, B.S.T.J., this issue, p. 1845.
4. Downing, R. W., Nowak, J. S., Tuomenoksa, L. S., No. 1 ESS Maintenance Plan, B.S.T.J., this issue, p. 1961.
5. Martellotto, N. A., Oehring, H., and Paull, M. C., PROCESS III, A Compiler-Assembler for No. 1 ESS, B.S.T.J., this issue, Part 2.



# No. 1 ESS Maintenance Plan

By R. W. DOWNING, J. S. NOWAK and L. S. TUOMENOKSA

(Manuscript received January 28, 1964)

*The No. 1 electronic switching system has a much higher concentration of control than previous telephone systems. This makes the task of providing continuous telephone service more challenging than ever. It is important that troubles be detected almost instantaneously, before many calls are affected. The main subjects of this article are (1) a scheme for duplicating control equipment and (2) a description of circuit and program facilities for taking advantage of this duplication to detect, to automatically recover from, and to analyze troubles.*

## I. OBJECTIVES

The success of a commercial telephone switching system can be measured on the basis of customer satisfaction and economic considerations. For customer satisfaction, a telephone switching system must provide continuous and accurate service without unreasonable delays. This quality of service must be provided 24 hours a day throughout the design life of the system. Thus a successful system must be both dependable and maintainable. Dependability is defined as a measure of service continuity and accuracy; maintainability is defined as a measure of the ease with which component failures can be detected, diagnosed, and corrected. A high degree of system dependability and maintainability have been considered extremely important design objectives<sup>1</sup> throughout the development of No. 1 ESS.

The No. 1 ESS dependability and maintainability objectives have been chosen to be competitive with the existing electromechanical telephone switching systems. The dependability objectives are: the system down time should not exceed 2 hours over its 40-year life, and the calls handled incorrectly should not exceed 0.02 per cent. The maintainability objectives of No. 1 ESS are: to design a system in which troubles can be located and repaired easily and rapidly and which can be left unattended for extended periods of time.

## II. OVER-ALL PLAN

The use of high-speed data processing circuits in No. 1 ESS has brought about an increase (as compared to existing switching systems) in the time sharing of circuits and consequently in the centralization of control functions. This centralization has many advantages, but it can make the system more vulnerable to component failures if not properly handled. For example, if no redundancy were provided it would be possible for a single component failure in the No. 1 ESS central control to cause a complete failure of the system. To avoid this catastrophe, duplication of units and other forms of redundancy are used throughout the system. Thus in No. 1 ESS, unlike its predecessors, redundancy is provided primarily for dependability rather than to increase the traffic handling capacity of the system.

Duplication is necessary to allow the system to operate in the presence of troubles. In addition, to insure that calls are handled correctly, interruptions of call processing due to component failures must be minimized. No. 1 ESS depends primarily on circuits for trouble detection. The check circuits activate a circuit labeled the "interrupt sequencer." This circuit transfers program control from the call processing programs to maintenance programs called "fault-recognition" programs. The function of these programs is to determine quickly an operational system configuration, establish it by appropriate switching of duplicate units, and then return to call processing. To minimize the length of interruptions to call processing, duplicate units can be switched at speeds comparable to the cycle time of the unit.\* For errors and most faults,† the fault-recognition programs can be completed before the interruption has interfered with call processing, i.e., in a few milliseconds. For faults in the circuits interconnecting the units, the analysis by fault-recognition programs may extend to a point where some interference to call processing results.

Duplication increases system up time. However, a finite probability of simultaneous failures in duplicated subsystems remains (see Section 3.2). To decrease this probability, an extensive effort has been made in No. 1 ESS to design intrinsically reliable circuits and to minimize the repair time. To achieve intrinsically reliable circuits, long-life components such as silicon and magnetic devices have been chosen. Liberal margins have been provided between component ratings and the actual operating

---

\* For example, the central controls can be switched between active and standby states (see Section 3.3) by operating flip-flops, whereas network controllers, which have a 25-millisecond cycle time, are relay switched.

† An error is defined as a malfunction, the symptom of which cannot be reproduced under program control; a fault is a malfunction whose symptom can be reproduced.



conditions. All logic circuits were designed using worst-case techniques.<sup>2</sup> To decrease the repair time, programs and circuits are provided to conduct diagnostic tests on a faulty unit and then inform the maintenance personnel of the test results. Maintenance dictionaries are provided for translating these test results to the location of the faulty package(s). By using standardized packages<sup>3</sup> and plug-in techniques,<sup>4,5</sup> faulty components are made readily accessible and repairable. No in-service adjustments (short of package removal) are required.

To summarize, the No. 1 ESS maintenance plan has the following major facets:

- (1) long-life components and conservative circuit design are used to achieve an intrinsically reliable system;
- (2) vital parts of the system are duplicated to retain an operational system in the presence of component failures;
- (3) circuits are provided for trouble detection and switching of duplicate units, and fault-recognition programs are provided to identify the faulty unit and to control the switching to enable the system to recover from component failures rapidly and automatically;
- (4) diagnostic programs are provided to automatically test a faulty unit, maintenance dictionaries are provided for translating diagnostic results into the location of the faulty package, and plug-in techniques are used to facilitate the replacement of the faulty circuit.

Section III of this paper describes the duplication and switching plan. The maintenance circuits are described in Section IV, and maintenance programs in Section V. Section VI reviews the man-machine reaction to trouble and describes the plans for producing the maintenance dictionaries.

### III. DUPLICATION AND SWITCHING

Functionally, the system can be divided into a central processor and peripheral units (see Fig. 1). The central processor consists of program stores, call stores and the central control. The peripheral units include the switching network,<sup>6</sup> scanners,<sup>7</sup> signal distributors,<sup>7</sup> and the master control center.<sup>8</sup>

To process a call, a chain of units consisting of a central control, program stores, a central pulse distributor, some scan points, etc., must be operating properly. A failure in any one link in this chain will appear as a system failure. Our dependability objectives dictate that, on the average, the sum of the down times of all the links in this chain should not exceed 2 hours during the 40-year design life of the office.

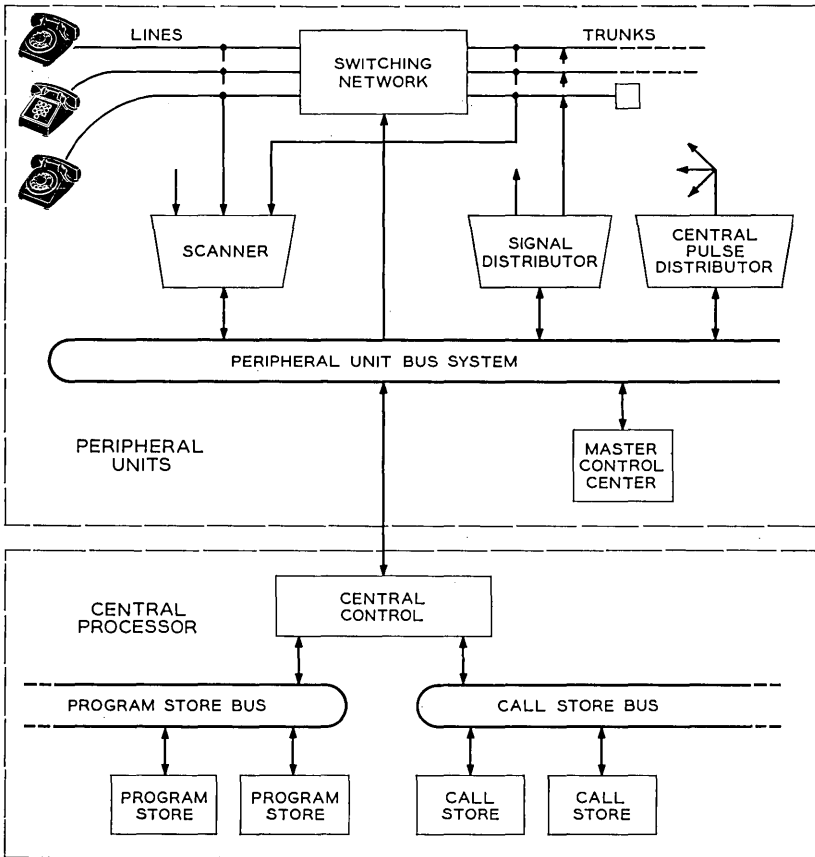


Fig. 1 — No. 1 ESS block diagram.

### 3.1 Reliability Predictions

To determine what form of duplication, if any, is required for each subsystem, one is interested in calculating the expected down time for each subsystem. The expected down time depends on the mean time between failures for each subsystem and the mean repair time.

Each subsystem is made up of large populations of many different component parts. These are operated at varying degrees of severity with respect to their nominal ratings. Failures observed in these types of units have, in general, been found to be distributed randomly, and the failure rate tends to remain constant with time. Assuming the com-

ponent failure rates remain constant through the 40-year design life of the system, i.e., that there is no wear-out effect, mean time between failures for the various subsystems can be calculated from the component failure rates. For some components, such as resistors, the failure rates can be predicted with considerable confidence, since plentiful data are available from systems similar to No. 1 ESS. For other components, such as diodes and transistors, there is more uncertainty about the failure rates, since there is no previous experience available with the particular transistors and diodes under the circuit stresses encountered in No. 1 ESS. Accelerated testing of components and the performance statistics of similar components in other systems give some indication of the failure rates we may expect.

Fig. 2, which plots down time as a function of the mean time between failures, indicates the extent to which duplication improves subsystem dependability. Consider the central controls as an example. A central control contains approximately 45,000 diodes, 13,500 transistors, 35,000 resistors, 225,000 soldered connections, 55,900 connector terminals, and a small number of pulse transformers, capacitors, etc. To meet our dependability objective of 2 hours down time in 40 years for the system, the central control down time should not exceed approximately 1 hour in 40 years.\* Fig. 2 shows that with duplication we require a central control with a mean time between failures of 1200 hours to meet our objective.

Points A and B in Fig. 2 represent calculations which were made for two different sets of failures rates.† We expect that the component failure rates that will actually be experienced will fall in the range between these two points, and that they will be reasonably close to the point where our objectives are met.

\* Based on the number of components in central control as compared with the rest of the system.

† Component failure rates assumed, in FITs:

	A	B
transistors (29A)	5	50
diodes (mostly 447A and 449A)	3	25
resistors	2	10
pulse transformers	20	40
solder connections	0.5	1
connector terminals	1	2.

(A FIT is defined as one component failure in 10<sup>9</sup> operating hours.)

The component failure rates listed in columns A and B above will result in mean times between central control failures of 1800 and 360 hours respectively. Component failure rates in this range have been achieved in other systems. Although no system known to us has achieved the rates shown in column A, accelerated tests on components indicate that these rates may be realizable.

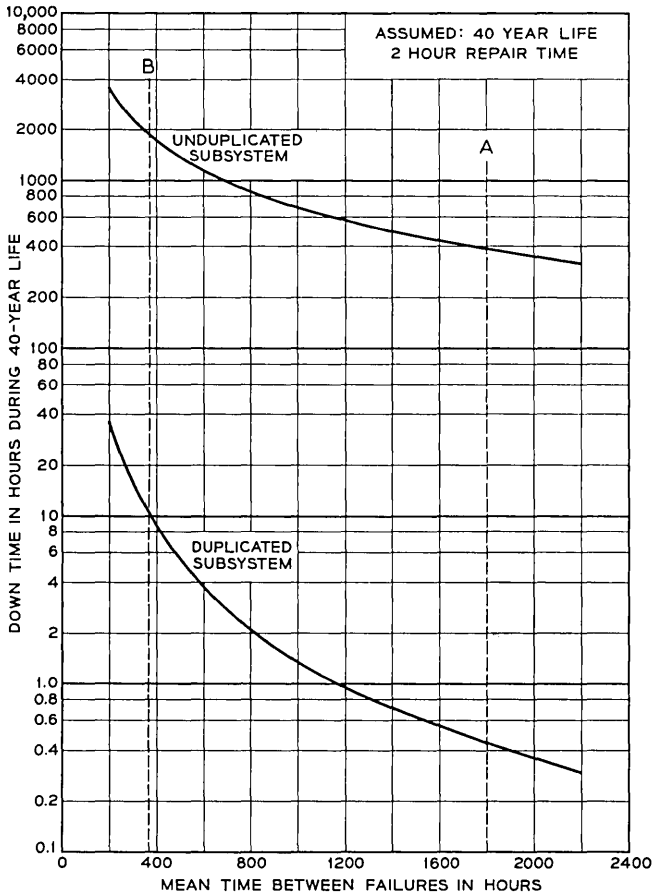


Fig. 2 — Down time vs mean time between failures for system with assumed 40-year life, two-hour repair time.

Similar studies in the early stages of development indicated the necessity for some type of duplication of all circuits where failure could affect a substantial number of customers.

Duplication on a subsystem level rather than on a circuit level was, in general, found to result in a more economical and maintainable design. The following sections describe the duplication plan adopted.

### 3.2 Duplication and Switching in the Central Processor

The major units of the central processor are interconnected by two bus systems:<sup>9</sup> the program store bus and the call store bus. The duplica-

tion in the community formed by the central control, program store bus, and the program stores will first be described.

The central control and the program store bus are completely duplicated. Each office, large or small, will have need for only one (duplicated) central control and one (duplicated) program store bus. Therefore no growth problem exists.

The program store requirements, on the other hand, are expected to vary from approximately 130,000 to 490,000 words, depending on the size of the office. To allow the number of program stores to vary from the minimum of two to the maximum of six in single-store increments, a "split" type of duplication was adopted. As shown in Fig. 3, the program stores can be viewed as links in a closed chain. Each store contains two 65,536-word blocks of information (labeled G and H) that are duplicated in the units to the left and to the right.

Each information block is assigned a 4-bit name, coded in 2/4 code. Identical names are assigned to the duplicate G and H information blocks. When a central control wants to fetch an instruction or data word from program stores, it broadcasts a 25-bit command on the address bus. Four bits identify the information block; 16 bits identify the word within the block; and one bit is a synchronizing pulse which gates the contents of the bus to the program stores. The remaining 4 bits identify one of the five modes in which the store can be addressed (see Section 4.2). Flip-flops called "route flip-flops" <sup>10</sup> within the stores control the inputs and outputs to the buses. One flip-flop controls the selection of an input bus. Two flip-flops determine whether the readouts from the H

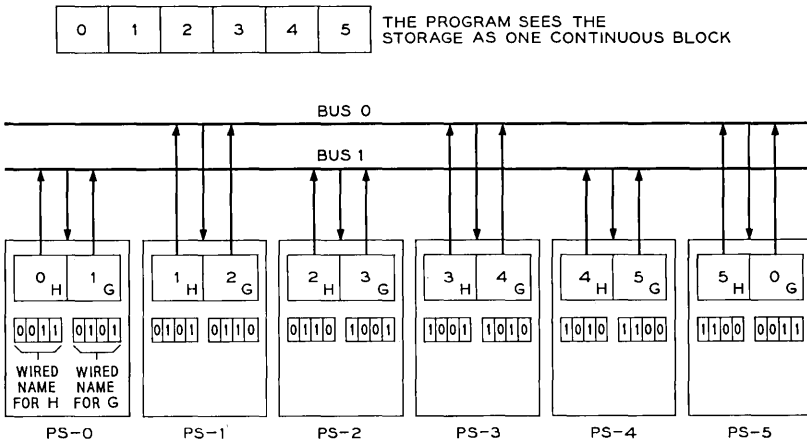


Fig. 3 — Program store duplication, six-store office.

block will be sent on bus 0 and/or bus 1. Similarly, two flip-flops control the sending of words from the G block. In each central control, there are three flip-flops which determine whether the central control transmits commands on address bus 0 or bus 1 or on both buses, and from which bus the central control receives the program store readouts.

Fig. 4 illustrates one possible address and readout routing in a three-store office. Consider a readout of an instruction from information block 1. Central control 0 broadcasts the synchronizing pulse, the 4-bit name of block 1, the 16-bit address of the word within the block 1, and the 4-bit mode (in this case normal mode) identification on bus 0. In synchronism with central control 0, central control 1 sends the identical information on bus 1. In store A, which is receiving from bus 0, the synchronizing pulse gates the address into the store. The 4-bit name code received from the bus matches the name of the G block in the store. The store therefore reads the location indicated by the address and sends the word stored there back to central control 0 via bus 0. Similarly, store B responds to the address on bus 1 and sends the duplicate word back on bus 1 to central control 1. Thus the identical word is fetched by the two central controls using separate routes to the two different stores.

One of the central controls is referred to as the active, the other as the standby. The status of which central control is active and which is the standby is controlled by a flip-flop in each central control. The active has a preferred selection with respect to store access; i.e., the active always fetches its own instructions and data. The standby may or may not fetch its own instructions, depending on the information block being read and the existing configuration. The active may transmit on one or

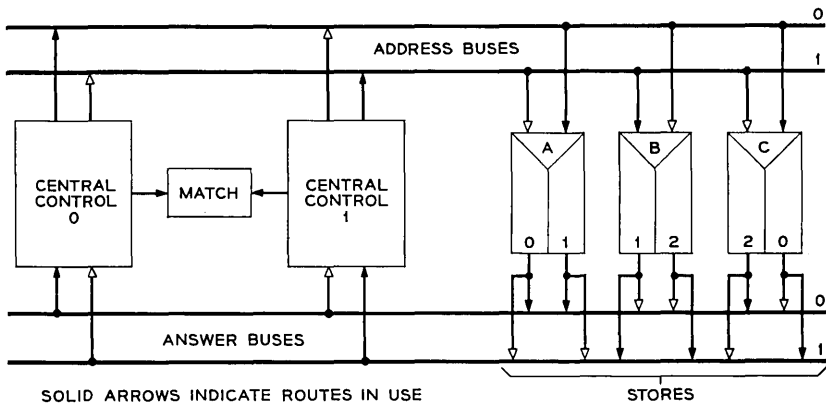


Fig. 4 — Three program store system — normal configuration.

both buses of a duplicate pair; the standby may transmit on at most one bus, and only if that bus is not used by the active. Also, as will be noted below, the active normally controls all the peripheral units.

The programs, with the exception of some maintenance programs, can be written without regard to the interconnection configuration. The program addresses a word; the address consists of the block name and the identification of the word within the block. The particular route used to obtain the word is determined solely by the route flip-flops within the stores and the central controls. The route flip-flops provide considerable flexibility in choosing configurations. Two examples will be described to illustrate this flexibility under trouble conditions.

As the first example, consider a situation where store C is not operating. Store C is isolated from the bus system by its operated trouble flip-flops. The configuration shown in Fig. 5 is established by appropriately setting route flip-flops in stores A and B and in the central controls. Central control 0 addresses both stores via bus 0. Words from information blocks 0 and 2 from stores A and B are sent back on both buses, since only one copy of these blocks is available at this point. The central controls still operate synchronously, and words from block 1 are obtained from independent sources, i.e., from stores A and B.

As long as any one copy of all information blocks is available, and one of the central controls and one of the buses are available, an operational configuration can be established. Fig. 6 shows the configuration which would be established if central control 0, store B and bus 1 were inoperative.

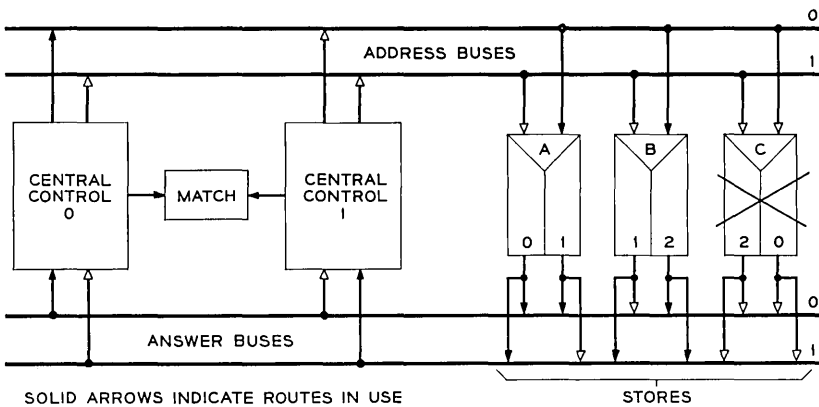


Fig. 5 — Three program store system — store C out of service.

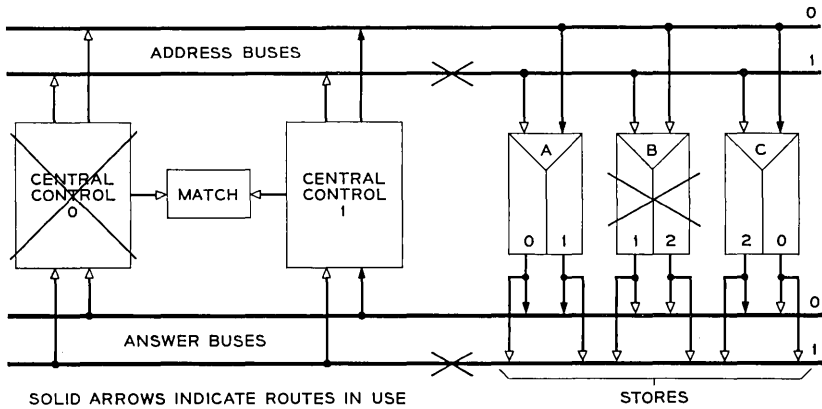


Fig. 6 — Three program store system — CC, PS, and bus out of service.

The four route flip-flops in each store that determine the routing of outputs can be controlled via the store address bus. The flip-flop that determines the address bus used is controlled via the central pulse distributors.

To summarize: since the correct operation of the central control-program store community is vital to the system operation, the central control, the bus and the program store memory are fully duplicated. To allow growth in increments of a single store, split duplication is used for memory. To allow (1) programs to be written independently of central control program store configuration, (2) rapid changes in configuration, and (3) growth without central control changes, route flip-flops are used to determine interconnection configurations, and enabling by name codes is used to select stores. In normal operation the two central controls, when possible, fetch their instructions from different stores via different paths. This provides for: (1) continuous exercising of standby equipment, and (2) an arrangement through which a thorough check of the configuration can be obtained by matching between the central controls.

The switching and duplication plan used in the call store community is nearly identical to that used in the program store community. Split duplication, name enabling, and configuration control by route flip-flops are used. The primary difference is in the size: whereas the number of program stores may vary from 2 to 6, the number of call stores may vary from 2 to 37.



### 3.3 *Duplication and Switching of Peripheral Units*

The peripheral units include the switching network,<sup>11</sup> signal distributors, central pulse distributors and scanners,<sup>7</sup> and the master control center,<sup>8</sup> i.e., all the units connected to the central control via the peripheral unit bus system.

The switching network is organized into frames. Each frame has its own controller, which sets up and takes down connections in the switching matrix associated with that frame. The active central control broadcasts addresses and commands via the common peripheral bus, which is connected to all the peripheral units. An enable pulse which the central control sends via the central pulse distributor gates the data from the peripheral unit address bus to the proper controller. (See Fig. 7a.)

The system time on a macroscopic level is divided into 5-millisecond intervals (see Section 5.1). At the beginning of every fifth interval, the central control sends orders to the controllers. During the following 25 milliseconds the controllers act on the orders and set up the requested paths.

Since failure of a controller would result in loss of the frame, controllers are duplicated (see Fig. 7b). The switching matrix in each frame is divided into two equal groups, each assigned to a controller. Under normal conditions each controller operates with its assigned switch group. Simultaneous path selections can be made within the two switch groups on the same frame. This has the additional advantage of providing continuous exercise of standby equipment. However, when a controller is in trouble its mate can be given control over both switch groups. This mode of operation, called the "combined mode," is established by operating a relay in the fault-free controller.

Since the operation of the entire peripheral system depends on the peripheral unit bus, this bus is duplicated (see Fig. 7c). Each frame is assigned four enable outputs from the central pulse distributors. The central control selects the controller and the bus by selecting one of four enable points. The central pulse distributors are provided in pairs. Two of the four enable points are assigned to one central pulse distributor, while the other two are given identical assignments in the mate central pulse distributor.

The central pulse distributors are connected by a duplicate bus system with the central control. The central pulse distributor address bus used by the active central control is determined by a flip-flop in the central control. To cause a connection to be set up in a network frame, the central control:

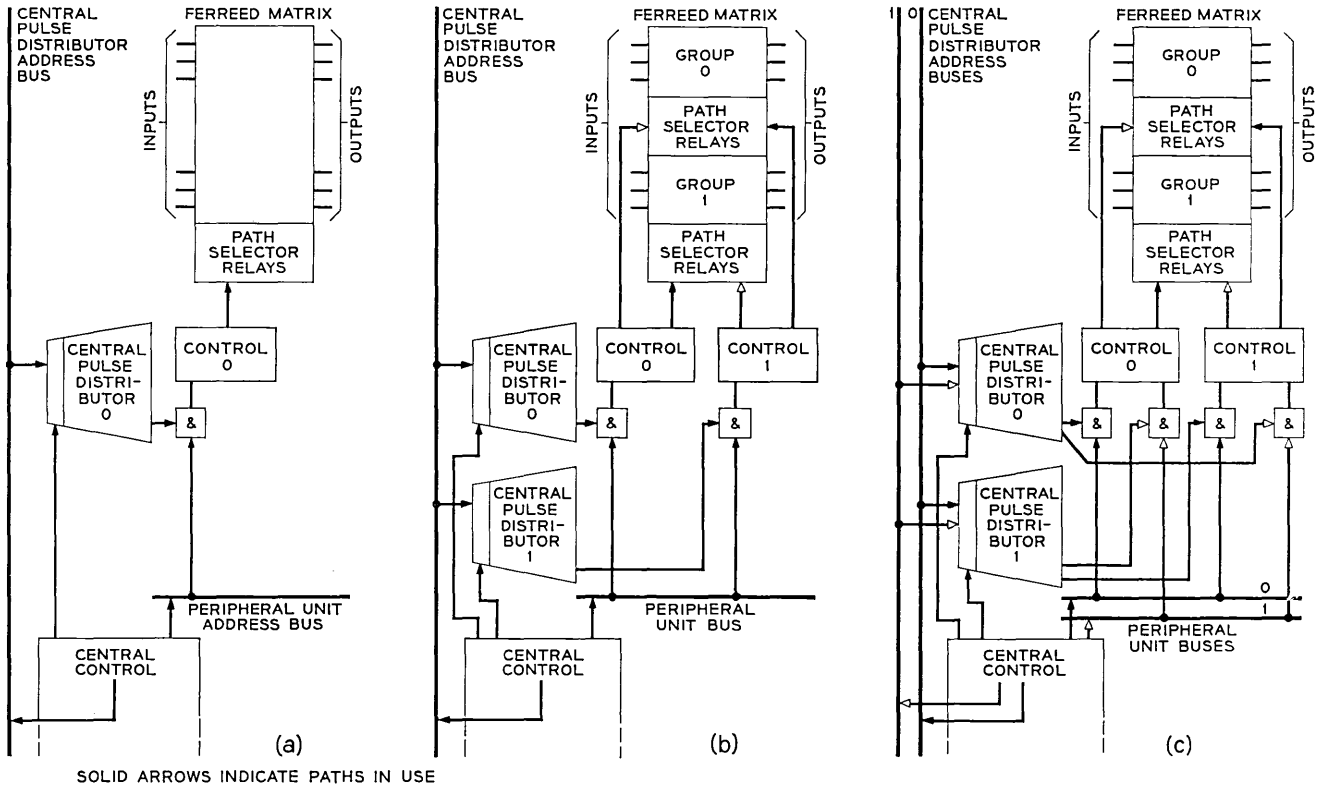


Fig. 7 — Network duplication.

- (1) selects the proper central pulse distributor (1 out of a possible 16) by an individual execute signal to that central pulse distributor, and
- (2) broadcasts the address of the appropriate enable point to the central pulse distributor via one of the central pulse distributor address buses;
- (3) broadcasts the address of the path to be selected over the peripheral unit address bus. The enable signal from the central pulse distributor gates this address from the proper bus and controller.

Thus, unlike the central processor where the configuration is entirely determined by the route flip-flops, the peripheral unit configuration used for peripheral actions is determined by the addresses sent out by the central control. For each peripheral unit operation, the central control consults a table in the call store memory, called the "enable table," to determine the correct peripheral address bus and controller to use. In the case of network and signal distributor controllers, this consultation is also necessary to make certain that only one command is sent to the controller in a given 25-millisecond period. To switch a controller out of service the central processor places the switching frame into the combined mode of operation by operating a relay in the fault-free controller; it then modifies the enable table so that all commands to the frame will be routed to the fault-free controller. To switch a bus out of service, only the enable table needs to be modified.

The network matrix is not duplicated, but it provides redundancy, since alternate paths exist over which a particular call can be placed. This redundancy is provided to insure that during normal operation of the system the probability of calls being blocked is within the system requirements. This redundancy also serves to insure adequate network performance in case of network troubles. For example, a faulty cross-point, after having been located and removed from service by marking associated links "busy" in the memory, will have the same effect upon system performance as a busy link.

The function of the signal distributor is to provide the central processor means for controlling relays in the system, primarily the relays in trunk circuits. A controller is provided which accepts the commands of the central control via the peripheral unit bus and translates these commands to an operation of an output relay. The output relays are in a sense analogous to the network matrix. They are not duplicated as such, but redundancy exists for traffic reasons, and in case of a relay failure an alternate trunk exists to complete a particular call. Since the loss of a controller would result in a loss of access to all 768 output points on the frame, two controllers are provided per frame. As with network con-

trollers during trouble-free conditions, each controller provides access to half the output terminals. If one controller is faulty, the remaining controller can be given access to the entire output field. As with all the peripheral units, the central processor selects the controller and the bus by using the appropriate enable output.

The scanners provide the central processor with means for supervising line and trunk on-hook and off-hook conditions, monitoring dial pulses, and observing various points within the system for administrative and maintenance purposes. Each scanner consists of a matrix of current sensing elements (ferrods) and a controller which allows the central processor to interrogate the ferrods. Since the loss of a ferrod affects only one customer or one trunk circuit, the ferrod matrix is not duplicated; since a fault in the controller may affect the entire matrix, the controller is duplicated. In case of network and signal distributors, both controllers can be active in that they can both be working simultaneously. Only one of the scanner controllers is active in any given 11-microsecond cycle. The active controller, selected by the central control using the appropriate enable output, receives the commands from the central control and transmits back to central control the state of the requested 16 ferrods via both of the peripheral answer buses. The central processor switches controllers or peripheral address buses by selection of enable points. A flip-flop in the central control selects the answer bus used by the central control.

The master control center contains the AMA recorders, maintenance teletypewriters, a magnet card writer, trunk and line test panel, and manual controls and displays. The AMA units are duplicated not only for dependability but to allow data storing while changing the magnetic tape. These units have access to both peripheral unit buses. The unit used currently for recording, designated the active unit, is determined by relays in the AMA control circuits. The central control can control these relays via central pulse distributors. The peripheral unit address bus used in any given transfer of data to the unit is determined by the enable output used.

A number of teletypewriters can be connected to No. 1 ESS. These include maintenance, traffic, and service order teletypewriters. The maintenance teletypewriters provide personnel with means for requesting system action and also means for the system to report diagnostic results or results of requested actions. Since this maintenance teletypewriter is the main communication link between the office personnel and the system, two maintenance teletypewriters are provided. One is always located at the master control center; the second may be at the master control center or may be in a remote location.

Master control center equipment other than the AMA units and maintenance teletypewriters is not duplicated. Continuous operation of this equipment is not vital to the system operation; a fault in the trunk and line test panel, for example, may delay the routine testing or restoration of some trunk but has no direct or immediate effect on system operation.

To summarize: unlike a fault in an unduplicated central processor, which would be likely to cause an office failure, a fault in the peripheral system may have a less serious affect. Therefore, only those parts of the peripheral system whose failure would affect a substantial number of customers (e.g., peripheral bus or network controller) are duplicated. Those parts of the peripheral system where failures would affect a single subscriber, (e.g., line ferrod) or where failures would reduce the traffic handling capacity of the system (e.g., network crosspoint) are not duplicated. The bus and controller used in each operation are selected by an enable point. This selection is determined by enable tables in the call stores. This method can be used because the frequency of use of peripheral unit controllers is lower than the frequency of use of stores in the central processor, making the consultation of enable tables possible. This method is more suitable for peripheral units than the route flip-flop and enable code method used in the stores, since it places minimum equipment in each peripheral unit and requires substantially the same equipment in the central controls regardless of the office size. These factors are important since (1) most controllers are directly associated with a relatively small number of lines and trunks and therefore any additional circuits placed in the controllers will reflect strongly in the over-all cost, and (2) the number of peripheral units is, in general, much greater and is more affected by office growth than the number of units in the central processor.

#### IV. MAINTENANCE CIRCUITS

The No. 1 ESS maintenance plan is implemented in part by circuits and in part by program. Frequently a given function can be performed by either. For such cases, the use of circuits and programs is balanced to yield the most economical solution. In evaluating the cost of a circuit solution, the cost of maintaining the maintenance circuit itself must be included. The over-all cost of the maintenance plan depends to a large extent on finding this correct balance.

The principal functions of maintenance facilities are trouble detection, recovery of an operational configuration after a trouble has been detected, and generation of diagnostic data that the maintenance personnel can use to locate the trouble.

Trouble detection is a function that requires continuous attention, and is therefore much better suited to circuits than programs, so that real time is not used for this function. In some instances, when trouble is detected the best course of action is first to retry the operation under circuit control before program operation is disturbed by an interrupt. Once the circuit has established with some certainty that trouble exists, the system control is transferred, by circuits, to programs that analyze the problem, determine an operational configuration and control the switching to establish the new configurations. Such analysis is logically complex but occurs infrequently. Therefore a program solution is more suitable. In the generation of diagnostic data for pinpointing the trouble, circuits are used to provide intermediate access in addition to normal input-output access, while the program is used to actually perform the tests and to interpret the test results. In diagnostics, the circuit-program balance is particularly important. If adequate test access is not provided, it becomes either costly in terms of program words, or impossible to pinpoint the fault.

Thus, circuits are used for: (1) trouble detection, (2) automatic retries, (3) administration of coarse program priority, (4) switching of duplicate units, and (5) diagnostic access. Of these functions, trouble detection, retries, and diagnostic access are the subjects of this section. The administration of coarse program priority, i.e., the transfer of program control by interrupting current programs, will be discussed in Section V. The switching of duplicate units was discussed in Section III.

#### 4.1 *Central Control*<sup>2,3</sup>

Since the central control is a data processor which continuously manipulates and modifies data, it is difficult to incorporate self-checking features within a single central control. The concepts of parity checks and other checks which can be provided for transmission or storage units are not practical for a unit such as the central control.

The primary tools incorporated for detecting and diagnosing troubles within the central controls are match circuits capable of comparing a number of internal nodes.

To match information between central controls, each central control transmits information describing its internal state to the duplicate central control. The match information is buffered in match registers, and the registers are then compared. Each match operation compares 24 bits in parallel (i.e., one word of data). Each complete match operation requires nearly one 5.5- $\mu$ sec machine cycle. In order to match two words per machine cycle, two match circuits are provided in each central con-

trol. Fig. 8 illustrates the matching scheme. As shown, each matcher has access to 6 internal points, providing access to 12 points or a total of 288 internal leads. These intermediate access points partition the central control into a number of smaller circuit blocks. The ability to match the internal masked and unmasked bus<sup>2</sup> provides indirect access to all of the index registers. Each match cycle can be sampled at any one of three times during a machine cycle.

The match circuits are designed to operate in five different modes. The modes of operation and their uses are:

(1) Routine match mode — In this mode, two points are matched each cycle, one in match circuit zero and one in match circuit one. In this mode a match is always expected. If a mismatch is detected, special

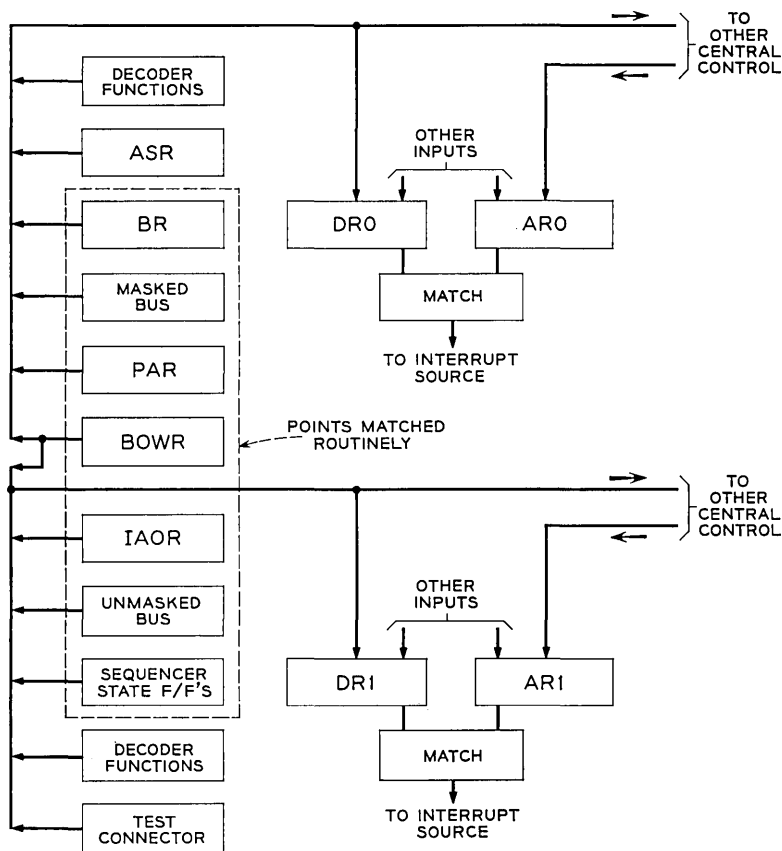


Fig. 8 — Central control matching access.

actions are initiated. The special actions are optional and are specified when the mode is established. The options available (in any combination) are:

- (a) to stop matching and save the contents of the match registers,
- (b) to generate a maintenance interrupt,
- (c) to stop the standby central control.

These options are also available for all the modes to be described below. In the normal use of the routine match mode, options (a) and (b) are used.

In the routine match mode, the points matched each cycle are dependent upon the order being executed. On "write" orders, the index address output register (IAOR) and buffer register (BR) are matched. This compares the address and data generated on the write order. On "read" or "peripheral" orders, the IAOR and the masked bus (MB) are matched. This again compares addresses and data. After a reread (see Section 4.2) or error correction of a program store word, the buffer order word register (BOWR) is matched. On transfers, the program address register (PAR) and unmasked bus (UB) are matched. If none of these special operations is being performed, the program address register, the unmasked bus, the masked bus, and a number of sequencer points are matched in a cyclic manner.

The routine match mode is the normal trouble detection mode for the central control matchers. The two central controls operate in parallel, and as long as they continue to do so, the matchers always detect a match condition. If a mismatch occurs, an interrupt causing a transfer to a fault recognition program occurs.

(2) Directed match mode — In this mode, the points matched and the time of the match in each matcher are determined by the contents of the match control register. This register is set up under program control when the mode is established. In this mode, the specified points are matched once every cycle. The interrupt and other options are also available. This mode is used to determine whether a mismatch occurs at a specified match point during the execution of a specific program.

(3) Sampled match — The sampled match mode is used in conjunction with a mismatch sampling order (EMMS) to perform one sample per matcher at a specified point in a program. During this mode no matching occurs until the EMMS order is executed. The EMMS order specifies the points to be matched, the time of the match, and the number of machine cycles following the EMMS order on which the match is to occur. The information sampled by the matchers is retained in the match registers for examination. This mode is used in the central control



diagnostic program to obtain critical information during the execution of test programs.

(4) Breakpoint mode — In this mode, the active central control can monitor program store addresses, call store addresses, or both, of the standby central control and perform any of the match options when the standby reaches any prespecified program store or call store address. This mode is useful in controlling off-line testing of the standby central control. The standby system can be set up to execute a program independently of the active program to stop the standby system, and to interrupt the active program when the standby program reaches a pre-selected program address.

(5) Preset match — As an extension of the breakpoint mode, it is possible to sample any of the match points of the standby and to compare the point with a prespecified number in the match registers of the active. In this mode, if a match is detected, any of the available options can be executed.

There are additional circuits in the central control for detecting troubles, providing diagnostic access, and providing automatic retrials. Some of these are designed primarily for detecting troubles in units external to the central control. These will be described in the following sections. The remaining circuits provided for central control maintenance are:

(1) Inputs to the emergency-action circuit — Circuits are provided to detect catastrophic failures, such as a faulty clock, loss of power, or locked-up sequencers. If these circuits detect a failure, they signal the emergency-action circuit. The actions performed by the emergency-action circuit will be described in Section 5.6.

(2) Voltage regulator monitoring circuits — When voltages of 4.5 volts are required in the central control, regulators<sup>3</sup> are provided to drop the 24-volt supply to this level. Monitoring circuits are provided to check these voltage regulators. These monitors are connected to scan points for diagnostic testing.

(3) Start-stop and control word write — The active central control has the ability to stop and start the standby central control. It can do this automatically as a function of the matching mode, or under program control. This is accomplished by a sequencer which inhibits the standby decoders and sequencers. The stop action allows the sequencers to complete so that the standby central control can be started at some later time without mutilating the program actions.

While the standby central control is stopped, the active central control can insert data into a limited number of registers and control flip-flops of the standby central control. This feature provides additional input

access for diagnostic testing. The active central control can write into the match control registers of the standby central control and establish match modes without depending upon the standby central control to execute an instruction. The ability to control write into the program address register of the standby provides a simple way for the active central control to force the standby central control into step.

#### 4.2 *Program Store*<sup>10</sup>

Words stored in the program store include seven Hamming error detecting and correcting bits. These bits are computed over the word stored and the 16-bit address. On each reading, the central control performs a check over the address sent and the word received. The central control can detect and correct single errors in the received word, detect double errors in the received word, and detect double or single errors in the address. Single errors in the word are corrected by circuit means at the cost of one 5.5-microsecond cycle but without interrupting the program. When an error is corrected, an error counter in the central control is incremented. This error counter can be read under program control. Detection of a single error in the address or a double error in the word or the address will cause an automatic reread. A failure of the reread will cause an interrupt of the current program and a transfer to the program store fault-recognition program. Rereads will also cause an error counter to be incremented in the central control.

The information block name of the store<sup>10</sup> is transmitted in a two-out-of-four code. The store checks the validity of the code that it receives. Within the store, internal timing checks, waveform checks, and one-out-of- $n$  checks are made. If all these checks pass, the store transmits an all-seems-well pulse back to the central control together with the 44-bit word. If the central control fails to receive the all-seems-well pulse, it will reread. Failure of the reread will cause an interrupt and a transfer to a maintenance program.

Table I summarizes the various maintenance features and circuits in the program stores and in the program store-central control communication facilities. A maintenance read mode allows the central control to read the state of various points within the program store address circuits. The maintenance write mode allows the central control to change the state of the route flip-flops and other control flip-flops within the stores. Both of these modes use the existing communication buses. In addition, the central controls have access to some critical program store test points via the scanners and input access to control flip-flops via the central pulse distributors. Each store has a number of multicontact relays which,

TABLE I — MAINTENANCE CIRCUIT FEATURES OF PROGRAM STORE-CENTRAL CONTROL COMMUNITY

- 
- (1) Split duplication
  - (2) Common duplicate ac bus system
  - (3) Enabling by name code
  - (4) High-speed switching by route and trouble flip-flops
  - (5) Hamming code over word and address
  - (6) Name in 2/4 code (2-6 stores)
  - (7) "All-seems-well" pulse
  - (8) Automatic reread and single-error correction
  - (9) Error counters in central controls
  - (10) Matching of words between central controls
  - (11) Continuous use of standby equipment
  - (12) Maintenance read and write operations
  - (13) Access to program stores via scanners and central pulse distributors
  - (14) Monitor bus
  - (15) Automatic "shut-up."
- 

when operated, connect a number of internal points of the store to the monitor bus which terminates in a scanner. This allows additional access to the store for diagnostic purposes. The automatic "shut-up" isolates the store from the communication bus in the event of a failure which might cause the store which has failed to transmit falsely on the bus.

#### 4.3 Call Store<sup>12</sup>

The interconnecting, duplication and switching plan used for the call store-central control community is similar to that used for the program store-central control community. The main differences between the two plans are italicized in Table II. One difference is the size of the community. The call store community may contain as many as 37 call stores. Another difference is in the information coding. Instead of a Hamming

TABLE II — MAINTENANCE CIRCUIT FEATURES OF CALL STORE-CENTRAL CONTROL COMMUNITY

- 
- (1) Split duplication
  - (2) Common duplicate ac bus system
  - (3) Enabling by name code (*2-36 stores*)
  - (4) High-speed switching by route flip-flops
  - (5) *Parity bit over word, name, and address*
  - (6) *Parity bit over address*
  - (7) "All-seems-well" pulse
  - (8) Automatic reread
  - (9) Error counter in central controls
  - (10) Matching of words between central controls
  - (11) Continuous use of standby units
  - (12) Maintenance read and write operations
  - (13) Access to call stores via scanners and central pulse distributors
  - (14) Monitor bus
  - (15) Automatic "shut-up."
-

code, there are two simple parity checks. One parity bit is over the address, including the store name, and the word. This bit is stored with the word. It is checked in the central control each time a word is read from the call store. The other parity bit is over the address only. This bit is checked by the store each time it accepts an address from the bus. A failure of this check, as well as any failure of the internal store checks, inhibits the all-seems-well pulse. A failure to receive an all-seems-well pulse or a failure of the over-all parity check will cause the central control to reread or rewrite and to increment an error counter. A reread or rewrite failure will cause an interrupt of the current program and a transfer to the call store fault-recognition program.

#### 4.4 *Central Pulse Distributor*<sup>7</sup>

The central pulse distributor, a high-speed electronic translator, provides two types of outputs in response to commands from the central control. The first output is a bipolar pulse used to control trunk circuits and special control circuits located throughout the various subsystems. The second output is a unipolar pulse used to control certain trunk circuits and to enable peripheral subsystems such as the scanner controllers, signal distributor controllers, network controllers, etc. The unipolar enable pulse is used to select one out of  $n$  subsystems connected to a common address bus.

The receiving circuit returns a one-out-of- $n$  acknowledgment signal to the central pulse distributor by way of the same twisted pair over which it received the unipolar pulse. The central pulse distributor translates the one-out-of- $n$  code into the central pulse distributor address form (three one-out-of-8 codes) and returns this address to the central control, where it is matched against the original address.

Within the central pulse distributor, internal checks monitor the operation of the pulse receivers and check that one and only one pulse was received and acted upon from each of the three one-out-of-8 codes. In addition, an internal check is made to assure that current used to drive the matrix is within prescribed limits. If all checks pass, a signal labeled the "all-seems-well pulse" is returned to the central control. A check failure inhibits action on the command (thereby preventing false operations of the central pulse distributor) and inhibits the all-seems-well signal.

The private execute signal, which selects one of a plurality of central pulse distributors, is returned to the central control for verification. This check assures that the correct central pulse distributor, and only the correct one, received and acted upon the address.

Failure of any of the checks causes the central control interrupt sequencer to transfer program control to the peripheral unit fault-recognition programs.

As described in Section III, the central pulse distributors are provided in pairs. One central pulse distributor may be taken out of service by means of a signal from the mate central pulse distributor. This signal controls a flip-flop which will remove power from the execute signal circuit and from circuits which send the verify signals and the all-seems-well signals. This action prevents false generation of outputs. This out-of-service mode is called "quarantine."

To provide diagnostic access to the central pulse distributor, 64 simulated matrix loads are provided. Addresses with good or bad parity can be supplied by the central control to test the final matrix current sampling circuits and access circuitry. The combination of both the simulated matrix load and the trouble detection circuits allows the controller and the common matrix to be diagnosed without placing unnecessary restrictions on the assignments of central pulse distributor points and without generating harmful outputs.

#### 4.5 *Network and Signal Distributor*<sup>6,7,11</sup>

When a network or signal distributor receives a command, a number of dynamic checks are made within the controllers. These include: a check that one and only one path selection relay in each one-out-of- $n$  code group is operated; a check on the continuity of the path through the ferreed control windings; and a check of the amplitude of the pulse which operates the ferreed.

If any of the internal checks fail, the controller sequencing is halted and the controller remains in the trouble state. Each controller is assigned three scan points. The presence of trouble in a controller is detected by program interrogation of these scan points. The controller can be reset by a special common reset command broadcast by the central control.

To prevent erroneous network actions caused by a faulty controller, the mate controller, on command by the central control, can place the faulty controller into a "quarantine" mode. While in this mode, the faulty controller can receive addresses and orders but is prevented from gaining access to the ferreeds.

Diagnostic access to the controllers is provided by a diagnostic bus. Test points are located at key locations within a controller. These test points can be connected to the diagnostic bus by relays operated via the mate controller.

The switching matrix and the signal distributor output matrix include relatively few circuit checks. Most troubles in these circuits are detected by maintenance programs which are integrated into the call programs.

When a path is to be established through the line link network, a partial path is first set up and tested using a detector circuit that is connected to the path momentarily in the fourth switching stage. This detector, labeled "false cross and ground detector," can detect shorts between the tip and ring conductors and the presence of ground or any potential on either conductor.

The customer dial pulse receivers include a detector for foreign voltages. When a line is first connected through the network to a dial pulse receiver, this detector is read via a scan point.

Some of the more complex trunk and service circuits<sup>13</sup> include some check features. For example, seven scan points are connected to multi-frequency transmitters. Some states of these scan points indicate trouble conditions.

The signal distributor contains a circuit which checks whether the selected output relay actually changes its state. A check failure, i.e., no change in state, is reflected in the state of three scan points connected to the controller.

#### 4.6 *Scanner*<sup>7</sup>

The scanners are addressed by the central control via the peripheral unit address bus. The scanners receive commands in groups of one-out-of- $n$  codes. Like other peripheral units, the scanner, upon receiving an enable pulse, returns a verify signal to the central pulse distributor.

Within a scanner controller, the matrix current is sampled and a check is made that one and only one row in the scanner matrix has been selected. The results of these checks are sent back to the central control together with the states of the interrogated ferroids. The check signal is labeled an "all-seems-well scanner pulse." A failure inhibits the all-seems-well pulse, thereby alerting the central control that a trouble has been detected. The central control responds by generating a program interrupt, causing a program transfer to a fault-recognition program.

For purposes of diagnosis, a special maintenance command is provided. This command, broadcast together with an appropriate enable pulse, causes the scanner detector circuits to produce a known output which is sent back to the central control.

#### 4.7 *Alarms and the Master Control Center*<sup>8</sup>

Communication between the No. 1 ESS and the maintenance personnel is achieved by the following means:

- (a) an office alarm system,
- (b) local alarm circuits, display lamps, and power removal switches at the individual system units, and
- (c) a teletypewriter, visual displays, and manual controls at the master control center.

The local alarm system consists of detecting and indicating circuits located in the individual equipment frames. The office alarm system consists of aisle pilot lamps, main aisle lamps, exit lamps and audible alarms.

The main power equipment does not require a series of locating lamps, since it has its own alarm circuit providing both major and minor alarms. Any failure in the central processor or failures of both controllers in a peripheral subsystem will cause a major alarm. Other failures in peripheral subsystems will cause a minor alarm. Failure of a trunk in a group of trunks, of per-line equipment, or of network cross points will be reported via teletypewriter without an audible alarm.

Whenever trouble is detected by a local alarm circuit, the office alarm system alerts the maintenance man; pilot lamps direct him to the faulty equipment unit. Most troubles, however, are detected by the system under program control. In this case, the office alarm system directs the maintenance man to the master control center.

For a locally detected trouble, the alarm is retired at the faulty unit by removing power. For a system-detected trouble, the alarm is retired at the master control center, where a diagnostic printout is given at the teletypewriter. Using this printout, the maintenance man consults a dictionary to obtain the location of the fault.

Teletypewriters are used by the operating personnel to make recent changes of translation information, to request status reports, etc. The system, in turn, uses teletypewriters to print out test results, traffic information, permanent signal conditions, etc.

Additional facilities are provided at the master control center for storing AMA information on magnetic tapes, for updating the translation information contained in the program stores, and for testing lines and trunks. Thus the master control center represents the maintenance and administration center of the office.

## V. MAINTENANCE PROGRAMS

As described in Section IV, circuits are, in general, used for functions such as trouble detection and diagnostic access. On the other hand, the determination of an operational configuration, the control of switching to establish such a configuration, the pinpointing of faults, the conduct-

ing of tests, and recording and interpreting the test results are implemented by program.

The maintenance programs for the No. 1 ESS can be divided into three categories:

- (1) fault-recognition programs and emergency-action programs which determine and switch into service an operational system after a trouble has been detected;
- (2) diagnostic programs which conduct tests on a faulty unit to pinpoint the faulty circuit pack(s); and
- (3) exercise programs which supplement the hardware trouble detection facilities.

The vast majority of these programs are executed on-line\* as opposed to off-line. Some programs can be executed off-line upon request by the maintenance personnel. This facility will be discussed in Section 5.8.

As general design objectives, all of the maintenance programs should be:

- (1) generic — the same programs should be applicable to all offices. Parameters may be used to specify items such as the number of units in a given office.
- (2) uniform — both in their relationship to maintenance personnel and other programs
- (3) simple — for easy design and understanding
- (4) noninterfering — the maintenance programs should interfere with call processing only when absolutely necessary.

Before embarking on a discussion of the three categories of maintenance programs, a brief review of the call program operation is included so that the maintenance-call program relationship can be established.

### 5.1 *Review of Call Program Operation*<sup>14,15</sup>

All system programs are placed into a hierarchy according to their urgency. This hierarchy has many grades or steps. The programs in the upper part of the hierarchic ladder are called “nondeferrable” programs; programs in the lower part are called “deferrable.” The nondeferrable programs are either programs that must be performed in order to secure the system call processing ability, which may be in jeopardy, or programs which are associated with the system’s input-output functions. The input-output programs must be performed punctually and usually

\* Many of the programs are executed by both the active and standby data processor. For the purposes of this paper, the distinction between on-line and off-line is the following: an off-line program is a special program which is executed by the standby data processor while the active data processor executes its normal call programs; all other modes of program execution are termed on-line.



repetitively — otherwise data, e.g., dial pulses, will be lost. The deferrable programs process data already in the system and are not, therefore, as critically synchronized to real time as the nondeferrable programs. Dial pulse scanning, for instance, is a nondeferrable function. Tasks such as network path hunts and processing of traffic data are carried out by deferrable programs.

A control program called the “base-level main program” administers the priority of the deferrable programs. For example, scanning for new originations is given preference over collecting traffic data. In effect, all deferrable programs are placed in one of five work lists of the main program. The main program administers priority by initiating the programs on different lists at different frequencies. For example, jobs on List A are initiated 16 times for each time that programs on List E are initiated.

The main program and the programs it initiates are not “aware” of the passage of real time. To insure punctual performance of the nondeferrable programs without impairing the efficiency of the deferrable programs, an interrupt facility was designed. Every 5 milliseconds, a clock output triggers the interrupt sequencing circuit in the central controls. The interrupt sequencer allows the handling of the current instruction to be completed, inhibits the work on the next instruction, stores in a reserved area of the call store the address of the next instruction and the contents of some other flip-flops, and then transfers to what is called the “interrupt” program (see Fig. 9). The interrupt program stores the contents of the remainder of the central control registers in the call store and then transfers to the interrupt level main program, which initiates the various nondeferrable programs. When these programs have been completed, the central control registers, which at the time of the interrupt were stored in the call store under program control, are restored under program control. A special instruction is then executed which fetches the program store address which was about to be used at the time the interrupt occurred and transfers to the interrupted nondeferrable program at the base level. Thus the interrupt mechanism provides a means for executing the nondeferrable jobs without disturbing the deferrable programs. Under normal conditions, the system operates either on this interrupt level or on the base level. The coarse priority is determined by the circuits, i.e., the 5-millisecond clock and the interrupt sequencers. The finer steps in the priority are administered by the program, i.e., the main program on each interrupt level.

The nondeferrable jobs, in turn, are of two categories: high-priority and low-priority. High-priority programs, such as dial pulse scanning, *must* be performed punctually or information may be lost; low-priority nondeferrable programs, such as sending commands to the network, *should*

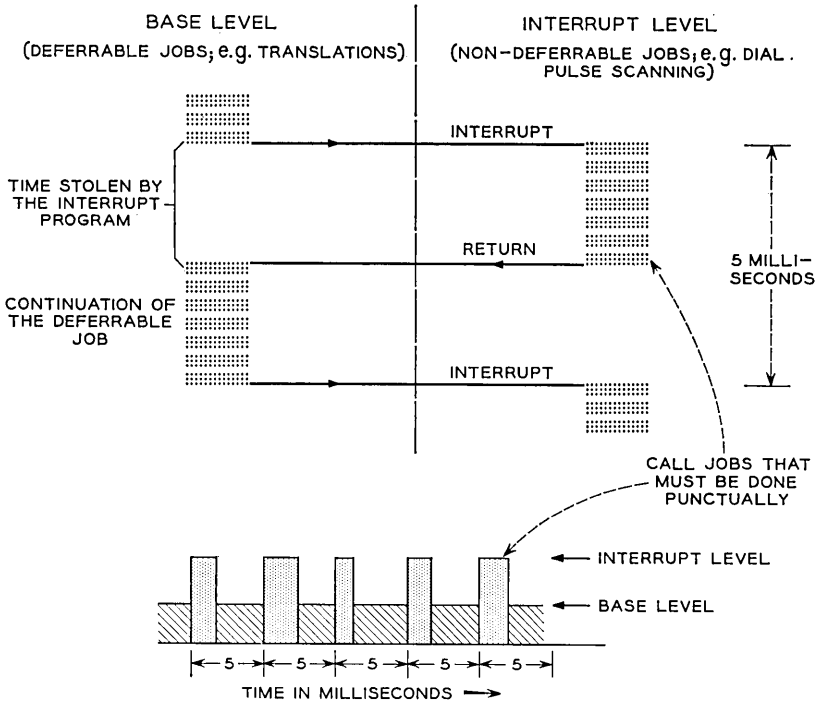


Fig. 9 — Call program operation (1).

be performed punctually. Occasionally, nondeferrable work may not be completed within a 5-millisecond interval. To insure the punctual performance of the high-priority work, a second interrupt level is provided. This interrupt, the H-level interrupt (the first interrupt described above is the J level), will occur if at the end of a 5-millisecond period the high-priority nondeferrable work has been completed but the low-priority has not (see Fig. 10). The H-level interrupt is implemented in much the same way as the J-level interrupt. The interrupt sequencer transfers the program control to the H-level main program, and after carrying out the high-level programs, returns the control to the J-level programs.

As long as the system is trouble-free, it operates in the base level and at H-J interrupt levels as determined by the interrupt hardware.

### 5.2 Fault-Recognition Programs

When circuit troubles occur and are detected by fault-detection circuits, program control must be transferred to programs that recover

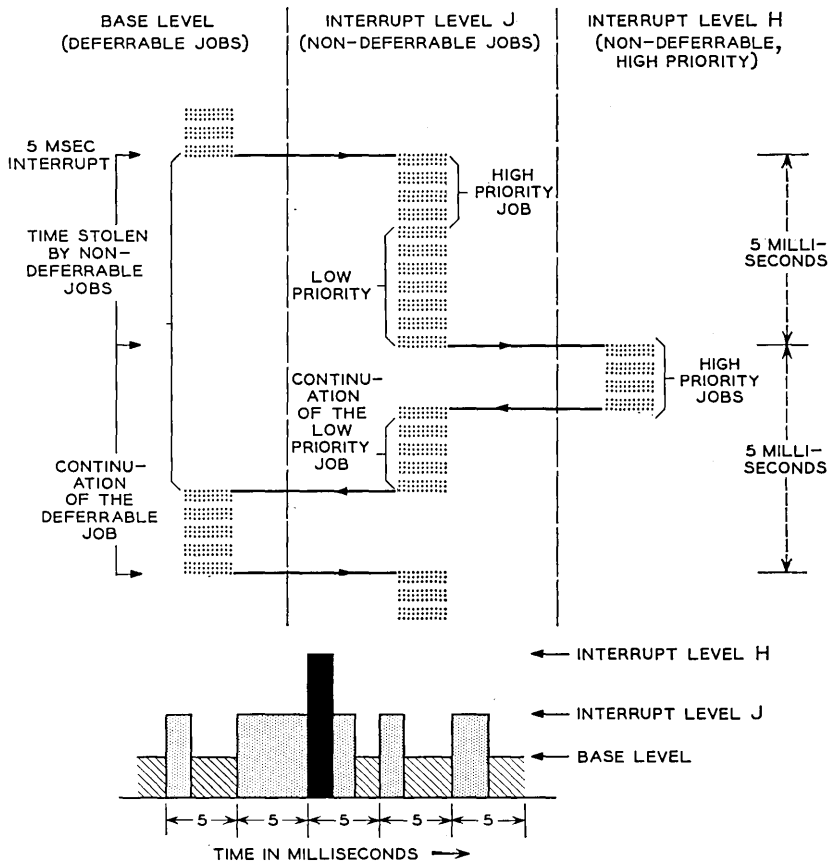


Fig. 10 — Call program operation (2).

the call processing ability of the system. These programs are called "fault-recognition" programs. The transfer of program control is implemented by the central control interrupt circuits. In call program operation, this mechanism is activated by the 5-millisecond clock and is used to interleave the three major priority classes in the call program hierarchy (high-priority nondeferrable, low-priority nondeferrable, and deferrable). In maintenance program operation, the interrupt circuit is in general activated by fault-detection circuits.

There are a total of ten interrupt levels, designated level A, level B, . . . , level K (I omitted) in descending order of priority. Levels A through G are associated with the master control center and with the fault-detection circuits listed in Fig. 11. The base-level programs may be inter-

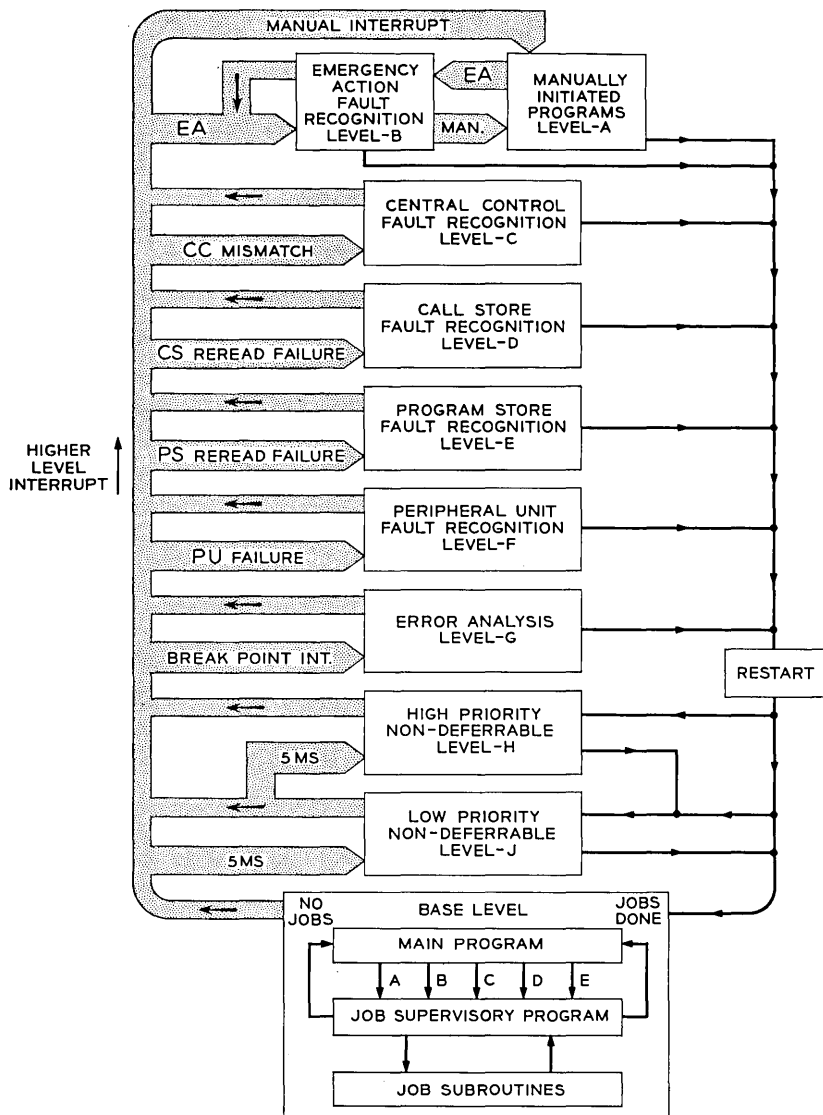


Fig. 11 — Interrupt levels and priorities.

rupted in the manner described previously to begin the execution of one of the ten interrupt programs. Once one of these programs is initiated, it in turn may be interrupted to permit performance of higher-level interrupt functions. However, with the exception of levels A and B, a given interrupt program may not be interrupted to perform a function at the same or lower level. The high-priority programs are assigned to the levels in the interrupt hierarchy according to the relative urgency of the action to be taken.

In order of descending urgency, there are five broad classes of programs:

- (1) Programs that recover the system's data processing ability; these are programs operating on interrupt levels A through E.
- (2) Programs that recover proper operation of the peripheral system; these are interrupt level F programs.
- (3) Programs that handle inputs and outputs; these programs operate on interrupt levels H and J interrupt programs.
- (4) Special test programs that operate on interrupt levels G and K.
- (5) Programs that process information within the system; these are the base-level programs.

Thus a mismatch between central controls generates a C-level interrupt, whereas detection of trouble in a peripheral system generates a level-F interrupt. In case of a central control mismatch, the data processing ability of the entire system is in jeopardy; in case of a peripheral unit malfunction the data processing ability is intact, although service to some or all subscribers may be affected.

In Section 5.5, the fault-recognition programs associated with levels C, D and E are discussed in detail. This section will describe the design considerations and general characteristics common to all fault-recognition programs.

The function of the fault-recognition programs is to recover the data and/or call processing ability of the system. The design objectives in decreasing order of importance are:

- (1) *The data processing ability must be recovered as long as a sufficient number of fault-free subsystems exist to form an operational configuration.*
- (2) *The malfunction that caused the entry to the fault-recognition program should not be allowed to interfere with the calls being processed.* Such interference could occur in a number of ways: (a) the call being handled in the central processor at the time of interrupt might be mutilated; (b) the malfunction may cause mutilation of temporary memory or alter the state of network connections in such a way that calls (both current and future) are affected; (c) the call processing ability may be lost for a

sufficient time to cause input-output information to be lost. For example, if the call store fault-recognition program that operates on interrupt level D takes 50 milliseconds to recover the system, no dial pulse scanning will take place during this interval and dial pulses from lines in the process of dialing may be lost.

(3) *The faulty subsystem should be located and switched out of service.* In some situations, it may be possible to recover an operational system without isolating the faulty subsystem. For example, it may be known that a fault exists in either a program store, a program store bus, or a central control. By switching out all these units, an operational active configuration can be established. The isolation of the faulty subsystem is also considered a function of the fault-recognition program.

(4) *The fault-recognition program should be able to distinguish between errors and faults.* An error is defined as a malfunction, the symptoms of which cannot be reproduced under program control. A fault is defined as a malfunction, the symptoms of which the program can reproduce at will. If the fault-recognition program determines that an error caused the interrupt, it will make a record of the interrupt. These records are utilized by error analysis programs to recognize abnormally high error rates and to determine the cause of such error rates.

It is not feasible to design a fault-recognition program which will always recover an operational system, isolate the faulty subsystem, and separate errors from faults rapidly enough so that there is no interference to call processing. For example, if the level-D fault-recognition program were to check thoroughly all call stores in a large system before returning the system to call processing, many calls might be lost, since such a check would take several hundred milliseconds. Therefore the fault-recognition programs were designed to minimize the *average* recovery time by doing the following:

(1) The programs are designed to recover an operational system rapidly (within 5 milliseconds) from the great majority of interrupts, i.e., from interrupts caused by errors and most faults. (Errors are assumed to be far more prevalent than faults.)

(2) Where the recovery is not simple or where subsequent interrupts indicate that the first attempt to recover was not successful, whatever time is necessary to recover an operational system will be taken.

(3) To expedite the return to call processing once an operational configuration is recovered, the isolation of the faulty subsystem and the analysis of errors are postponed and initiated later as a base-level program.

The fault-recognition programs can be divided into a main program, tests, service programs, and a restart program. The main program con-

trols the general course of action taken. The tests, as the name indicates, are programmed questioning of circuits to determine whether or not the circuits respond properly. Service routines include programs such as a program store configuration change routine, which calculates and establishes a program store configuration that fulfils constraints given as inputs to the routine. The restart program determines at which program point data and call processing should resume, restores the memory and central controls to the appropriate state, and returns the program control to the interrupted level.

### 5.3 *Diagnostic Programs*

The function of diagnostic programs is to generate test data to isolate the fault to a small number of plug-in circuit packs within the subsystem that has been taken out of service by a fault-recognition program.

Typically, a diagnostic program carries out a fixed sequence of tests. These tests are performed by observing the normal outputs of a unit or monitoring some special test points strategically located in the unit. The test points may be observed via a scanner, via normal communication routes (such as used by the control read operation of the stores), or via special communication buses (such as the match buses of the central controls). The test results are recorded in the call store and then printed out via a maintenance teletypewriter. The combinational pattern of which tests passed and which tests failed defines for the maintenance man the circuit pack(s) to be replaced. The translation from test results to the faulty circuit is done with the aid of a maintenance dictionary. The techniques employed to derive the dictionaries will be described later.

The diagnostic programs are normally requested by the fault-recognition programs at a time when an operational configuration already exists. The fault-recognition programs are high in program hierarchy, and hence the length of the fault-recognition programs adds directly to the system down time. The diagnostic programs, on the other hand, add only to the repair time.

Repair time is defined as the interval of time from the occurrence of a fault to its repair. The repair time affects both the dependability and maintainability of the system: when the repair time increases, the probability of the mate unit failing goes up. The repair time includes the time to detect the fault, recover an operational system, inform the personnel, get someone into the office, analyze and repair the fault. In this chain of events, the actions taken by personnel, particularly in an unattended office, may stretch to hours. Thus the diagnostic programs can be and

are assigned to the lowest step in the ladder of program hierarchy without substantially affecting the repair time.

The length of a diagnostic program may vary from a few milliseconds to several hundred milliseconds, depending on the unit being diagnosed. To prevent the diagnostic programs from interfering with call processing, they are divided into 10-millisecond segments. When a fault-recognition program discovers a faulty unit and switches it out of service, it also records the incident in the call store memory reserved for maintenance programs. This memory area is labeled the "maintenance control register." A program called the "maintenance control program" administers and controls this register.

Periodically, the base-level main program calls in the maintenance control program, which in turn discovers the need for the diagnostic action and initiates the first segment of the diagnostic program (see Fig. 12). At the end of a segment, the diagnostic program returns the

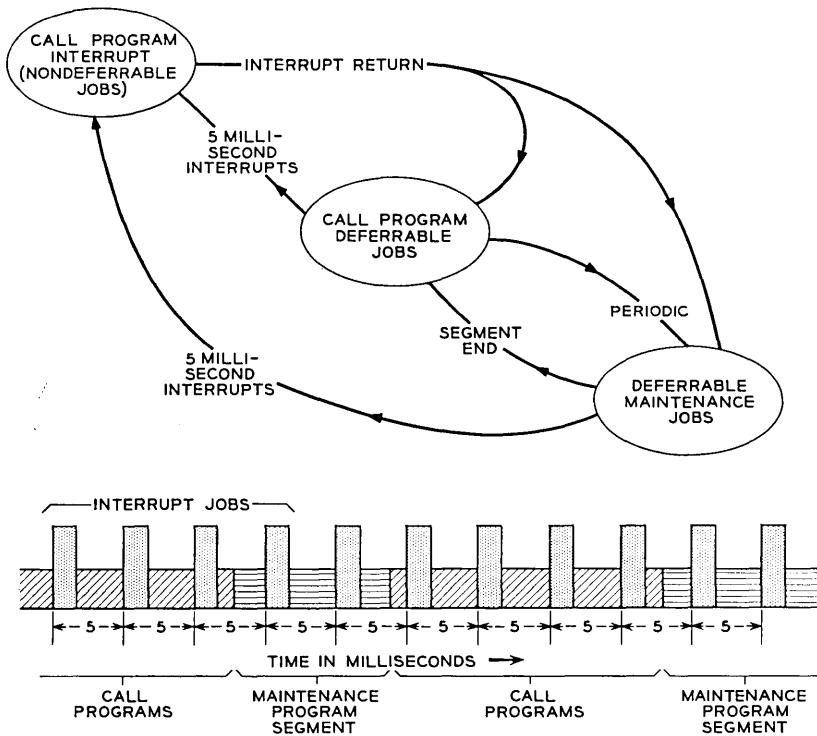


Fig. 12 — Maintenance-call processing interleaving.



control to the maintenance control program, which in turn returns to the base-level main program. On the subsequent main program visits to the maintenance control program, the other segments of the diagnostic program are initiated until the diagnostic program is completed. Each of the diagnostic program segments is interrupted by the J and possibly higher-level interrupts.

The maintenance control program also administers priority among diagnostic programs (central processor diagnostic programs are given higher priority than the peripheral unit programs), insures that no program holds the maintenance control register for more than 10 minutes, and performs tasks that are common to many of the maintenance programs (tasks such as timing, recording error information, control of common maintenance facilities, etc.).

There are a number of problems common to all diagnostic programs. One problem is that of ensuring consistent test results for the same fault every time that fault occurs. One possible cause of inconsistent results is intermittent troubles, i.e., troubles which cause a system failure at some times but not at others. However, if the diagnostic programs are repeated a number of times, it is likely that during some pass of the program the fault will become apparent. By repeatedly performing the diagnostic programs, intermittent faults should be located.

Initial conditions can also affect the consistency of diagnostic results. A fault can occur randomly in time, with the system memory elements in any possible state. The diagnostic tests must be so designed that the same test results are obtained (for a given fault) regardless of the state of the memory elements at the time the fault occurs. This implies that a unit under test must be properly initialized. This problem also affects the choice of the information to be recorded as pertinent test results, and it affects the order in which tests must be performed.

The effect on the recording of test results can best be illustrated by an example. Assume that we are attempting to test a flip-flop controlled by a single input gate. One "obvious" approach would be to write a zero into the flip-flop and record one bit of information indicating whether or not the flip-flop read a zero after the write operation. Next, one could write a one in the flip-flop and record an additional bit indicating whether or not this was successful. However, if this is done, inconsistent results will be obtained for certain types of troubles. Assume a fault in which the gating function controlling the inputs to the flip-flop is inoperative, so that it is impossible to write anything into the flip-flop. Then the information in the flip-flop is variable, depending upon the state of the flip-flop when the fault occurred. If at one occurrence of the

fault a zero was in the flip-flop, the zero test would pass, but the one test would fail. If at another time a one was in the flip-flop when a fault occurred, the zero test would fail, but the one test would pass. If both of these results are recorded as independent test results, then the results may differ from one fault occurrence to another. This problem can be avoided if only one result is recorded for the above two tests. This result should be the union of the results of the zero test and the one test.

This same problem affects the order in which tests must be performed on a unit. To test a unit, one must apply test inputs to the unit and observe the outputs. As described in a previous section, for nearly all units test points are provided to give input-output access in addition to the normal input-output access. Even so, it is not possible to provide independent input-output access to all circuits within each unit. Consequently, in testing a specific circuit it is sometimes necessary to use another circuit within that unit as an input or output device. However, if this circuit, used as a tool to test another, contains memory elements which may suffer from the type of fault described above, inconsistent results may occur if the problem is not handled properly.

Generally, a circuit is tested before that circuit is used to test another; the diagnosis is terminated if a failure is found within the original circuit. In most cases this can be done with little loss in fault resolution. Where this cannot be done, one must design the test in such a way that the results obtained are independent of the memory state of the circuit.

Normally, when a diagnosis is requested after a fault is discovered by some program, a complete diagnosis is performed, and the results are printed out in a convenient reduced form. In addition, the maintenance man will be able to request a diagnosis from the system teletypewriter. He will be able to request a complete diagnosis, in which the results are printed out in reduced or in unprocessed form, or he can request that only certain parts or phases of the diagnosis be performed.

A diagnostic program is automatically initiated whenever power is restored to a unit which previously had power removed. When a unit is being repaired, power is normally removed. When power is restored, a diagnostic program will be automatically requested and, if it passes, the unit will be restored to service. If a failure is detected, the unit will be left out of service and the test results printed out on the teletypewriter.

#### 5.4 *Exercise Programs*

The third category of maintenance programs consists of exercise programs which exist for the following reasons:

(a) *Supplementing trouble-detection facilities*: test calls, for instance, are initiated periodically to detect troubles that might otherwise go undetected.

(b) *Searching for uncorrected errors*: some programs, for example, look for discrepancies between the network hardware and the network map stored in call store.

(c) *Checking trouble detection circuits*: mismatches, for instance, are intentionally introduced to check the response of the system.

(d) *Exercising infrequently used hardware*: the program store configuration, for example, is periodically changed to ascertain that it can be changed when needed.

The exercise programs may be initiated automatically and periodically by the system. They may also be initiated on demand by other programs or by maintenance personnel.

The exercise programs, like diagnostic programs, are of low priority and operate on the base program level under the control of the maintenance control programs. As with the diagnostic programs, the prime design consideration is to minimize the program length.

### 5.5 *Implementation of Fault-Recognition Programs*

In the interests of brevity the following discussion is limited to those programs concerned with maintaining the central processor.

#### 5.5.1 *Central Control Fault Recognition*

The match circuits of the central control are the primary tools for detecting central control troubles. These circuits normally operate in the routine match mode discussed previously. When the system operates in this mode, a mismatch results in a level-C interrupt source being set. Provided that no higher interrupt level is active, the setting of this source results in a C-level interrupt program being entered by the interrupt sequencer. The C-level interrupt program is the central control fault-recognition program.

When this program is entered, the only fact that is readily apparent is that there has been some disagreement between the two central controls. This disagreement may have been caused by a random error which affected one of the central controls, by a fault in the active central control, a fault in the standby central control, or by a fault in some external unit which affected only one central control. The basic function of this program is to determine which of these possibilities exists. If it is determined that the mismatch was caused by an error, the two central

controls are put back into step and routine matching restored. If it is determined that one of the units is faulty, that unit is removed from service and the appropriate diagnostic program is requested.

As will be described later, when troubles are detected in some units external to the central control (such as the stores), information (such as an address) is often saved within the central control. This aids the fault recognition program in locating the suspect unit. This is not true in the case of the central controls, however. Since the matching is not instantaneous, nor are all internal points matched continuously, and since trouble can occur randomly within any program, no information is available when the central control fault recognition is entered to give any indication as to which central control contains incorrect information. Information is available within the match control register (MACR) and mode control register (MOCR) of the central control which defines the internal point where the mismatch was detected. However, this information gives no prior knowledge as to which central control contains incorrect data.

The central control fault-recognition program strives to determine which unit (if either) is faulty by attempting to reproduce the trouble symptoms under controlled conditions by logically exercising the central control hardware. If the trouble symptom cannot be reproduced, the trouble is classified as an error and the central controls are returned to parallel data processing. If the trouble symptom is reproduced, the faulty central control is removed from service.

Fig. 13 shows the basic program actions performed by the central control fault recognition program. First the central controls are forced into step and a directed match mode of the program address register (PAR) and index adder output register (IAOR) is established in both central controls. In this mode, the PAR and IAOR are matched once per machine cycle, and if a mismatch occurs at any time, this fact is retained in central control memory.

With these conditions established, the testing begins. The tests are divided into routines which exercise specific hardware areas of the central control (for example, the index adder and its associated registers). These test routines consist of data manipulating operations which expect to find known answers if all operations are performed successfully. The expected answers are checked using conditional transfer orders. For example, a very simple test of the accumulator adder would be to add zero to zero and transfer to a failure routine if the answer is not zero.

If all conditional transfers pass, the program checks to see if the two central controls are still in step by examining whether a mismatch has

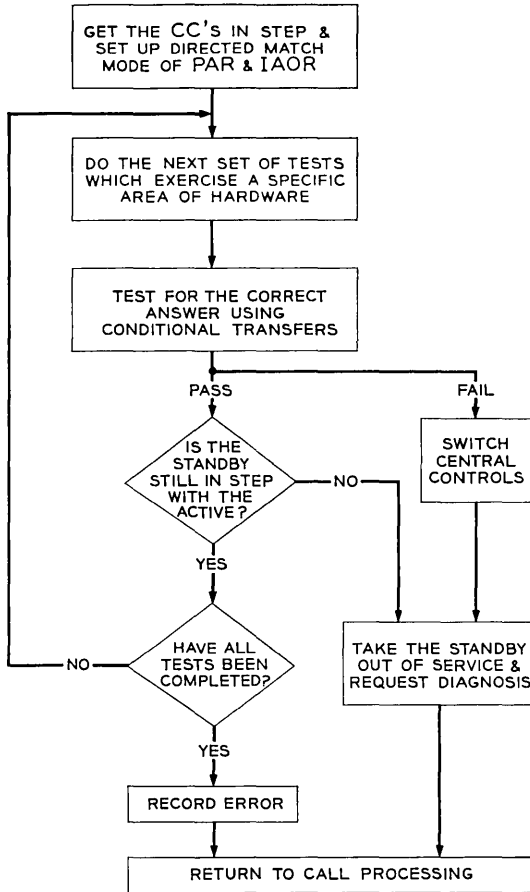


Fig. 13 — Central control fault-recognition test mode.

occurred somewhere in the test program. If they are in step, the testing continues until all tests have been completed successfully or a faulty unit is found and taken out of service. If all tests pass, the error is recorded and the restart program returns control to the call processing programs.

Let us review briefly what is happening within the two central controls during the fault-recognition program. First the two central controls are brought into step and the directed match mode established. With this accomplished, the test exercises are executed. During these exercises each central control is testing itself via the mechanism of

conditional transfers. If the active central control fails a test, it will transfer to a routine for switching central controls (i.e., interchanging the active and standby central controls) and presumably will switch them by changing the activity flip-flop via a central pulse distributor operation. (The possibility of the active central control being incapable of performing the switch will be discussed below.) Switching of central controls generates a B-level interrupt which will lead back to the central control fault-recognition program, which should find the standby central control faulty (as described below) and take it out of service. Thus the active central control will have been switched out of service.

If the standby central control is faulty and fails a conditional transfer test, it will also transfer to the routine for switching central controls. However, the standby unit is incapable of switching central controls because of its restricted access to the peripheral system. After attempting to perform a switch it will get stuck in a program loop or attempt to follow the actions of the active central control. If the active unit passes its tests (and we assume a fault in only one central control), then when the active unit later examines the match circuits it will see the mismatch generated by the fact that the standby unit transferred to the switch routine. Upon detecting this condition, it will take the standby central control out of service, request a central control diagnosis and return to call processing. The standby central control is taken out of service by modifying the bus control flip-flops so that the standby unit transmits to no external equipment, disjoining the central controls, and setting the trouble flip-flop to inform the emergency-action circuit of the faulty status of the standby.

If a trouble is found in the standby central control, it can be readily taken out of service by the active central control. If the active unit detects trouble within itself it may switch itself out of service. For some very basic troubles the active unit will be incapable of performing this switch operation. For these troubles the emergency-action circuit is relied upon. When the level-C interrupt is generated, a flip-flop is set by the interrupt sequencer, which will activate an emergency-action timer. This circuit will time out in 40 milliseconds if the fault-recognition program does not return to call processing within the 40-millisecond interval. If the central control fault-recognition program is capable of locating the faulty unit and removing it from service, the 40-millisecond timer will be stopped. If, however, the program gets "lost" because of a very basic trouble in the active central control, the emergency action circuit is activated after 40 milliseconds. This circuit will switch the central controls. In addition, if the fault-recognition program recognizes

that it is incapable of performing the switch it will attempt to induce an emergency-action cycle before the 40-millisecond timeout.

To exercise the central controls completely would require at least 25 milliseconds. To avoid taking this much time for each interrupt, the central control fault-recognition program is divided into two parts, a first-look program and a complete check program. Fig. 14 shows how these two programs are used to perform the over-all fault recognition function. As shown, a mismatch causes an interrupt to the first-look control which exercises only those portions of the central control which are directly associated with this mismatch point. For example if the mismatch was detected at the index adder output register, the first-look program will exercise the index adder and its input regis-

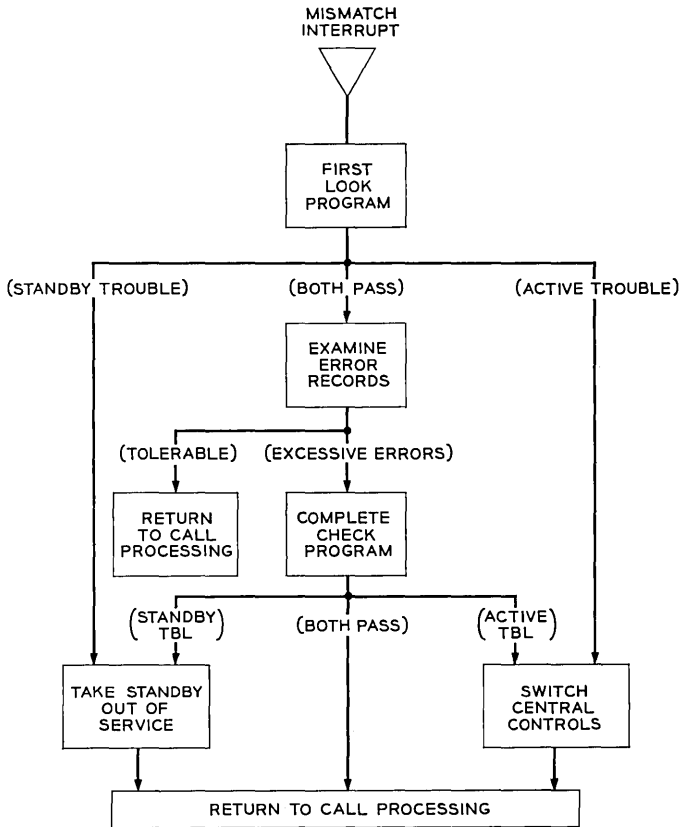


Fig. 14 — Central control fault-recognition program.

ters, and the index registers. If no fault is found in this limited area, it is assumed that the mismatch was caused by an error; the error is recorded and past error records are examined. As long as the error rate is low, the first-look program will return to call processing.

If the error rate is found to be excessive, the complete check program is entered. This program is a fairly complete test of the central controls. The complete check program is also designed to find and classify faults in the circuits interconnecting central control and other units. When necessary, this program will link to the call store, program store and peripheral unit fault recognition routines to recover a complete system. If the complete check program finds a faulty unit, that unit is taken out of service and a diagnostic program requested. If the complete check program is performed a number of times to no avail, additional maintenance actions, such as a diagnosis of the standby central control, will be requested.

#### *5.5.2 Program Store Fault Recognition*

The program store fault-recognition program is entered by a level-E interrupt following a reread failure of a program store word. A reread is performed when the program store fails to return an all-seems-well signal or when the central control detects a double error or an address error in a program store word.

Since the trouble causing the interrupt may be in the program store containing test programs (the base program store), the initial portion of the level-E interrupt program is located in the call store. Thus a level-E interrupt results in a wired transfer to a call store controlled program.

The principal function of this call store controlled program is to establish a "base" program store (i.e., a program store containing the remaining test programs) with which the active central control can communicate. This program first examines flip-flops within each store which indicate whether that program store had an all-seems-well failure. These flip-flops are examined by scanning. If it is determined that a single program store is faulty, the call store controlled program will take the store out of service, establish connections to a usable base program store, and transfer to a program within this store to complete the fault recognition function.

Not all program store troubles are detected as all-seems-well failures. Troubles in readout channels of the program store or buses will be detected by the central control as double errors, or address errors (or as



an excessive number of single errors, which will be discussed later) and will not be registered in the all-seems-well flip-flops within the program stores. In this case, the faulty store is not immediately identifiable.

To handle troubles of this type a more exhaustive back-up program is required. If the call store controlled program does not find a store with its all-seems-well flip-flop set, it uses an "establish base" routine to connect the active central control to a base program store. If this is successful, control is transferred to a "bootstrap" program located in the base store. This program will test the remaining program stores until it has found a sufficient number of working stores to be able to connect a full copy of memory to the active program store bus. A deferred fault-recognition program is then requested, and control is returned to call processing.

If the establish base and bootstrap programs are unable to establish an active system of program stores, they will induce the emergency-action circuit to take over and alter central control-program store interconnections until a workable configuration is obtained. (See Section 5.6.)

The deferred fault-recognition program which may be requested by the above programs is the highest priority deferrable maintenance task. The function of the deferred fault-recognition program for the program stores is to test all standby program stores and access to all stores from the standby bus, the standby program store bus, and the standby central control. Whenever a trouble occurs which is not a clear-cut single fault in one program store, the interrupt program described above does sufficient testing and switching to recover a complete copy of memory for the active system. However, the status of the standby stores, bus and central controls is left in doubt. The deferred fault-recognition program tests the standby stores, buses and central control. If any of these units is found faulty, it is removed from service and the appropriate diagnosis requested.

The program store fault-recognition program is entered whenever a program store reread failure occurs as a result of a repeated double error, address error, or an all-seems-well failure. Single errors in program store words can be detected and corrected, but will not result in a level-E interrupt. Thus a permanent fault in a readout channel would not initiate the fault recognition program. Routine maintenance programs are provided for troubles of this type.

Whenever a single error is detected, a hardware counter in the central control is incremented. A routine exercise program examines this error counter periodically. As long as the error count remains low, the counter

is reset and call programming is reentered in the normal manner. If the error count is found to be excessive, special error analysis programs are initiated. These programs attempt to locate the unit producing the high error rate by modifying the interconnection configuration of central control, program stores and buses, and again monitoring the error rate. With the error information of the initial configuration plus that obtained by changing the configuration twice, it is possible to locate the error source, provided it is consistently producing errors. When the suspected unit is located, it is taken out of service and a diagnosis for that unit is requested.

### 5.5.3 *Call Store Fault Recognition*

The duplication and switching plan for the call stores is nearly identical to that for the program stores. Consequently, many of the problems encountered are similar. The program is complicated by having to deal with a larger number of units and by having to cope with the fact that for some troubles it may not have any temporary memory to depend upon, with the exception of the internal central control registers. In addition, all testing of call stores must be performed while protecting the information stored in the temporary memories.

The call store fault recognition is called in as a level-D interrupt program when a reread failure or rewrite failure of a call store word is encountered. When this failure is detected and the interrupt request is made, the address at which the failure occurred is saved in the central control match registers. This program is again divided into a first-look program, which handles most simple troubles, and a bootstrap program to handle the more difficult troubles in the call store community.

The first-look program uses the failing address to determine which of two (recalling duplication) call stores failed. It then examines the failure indications retained in the central controls to determine which central control detected the trouble. Knowing this and the call store interconnection configuration, it determines which call store responded improperly and removes it from active use if a duplicate is available. It then performs an access test to ensure that the new call store configuration is set up properly. If this is performed successfully, a deferred fault-recognition program is requested and call processing is reentered. The deferred fault-recognition program will, at a later time, check to see that the call store removed from service is truly faulty and, if so, mark it in trouble and request a diagnosis. If it determines that the call store is fault-free, it will update its memory and return it to normal operation.

If any of the conditions assumed above are not met, the call store fault-recognition employs a bootstrap program to restore a complete copy of temporary memory to the active system. This program assumes all call stores are faulty until proven otherwise. It tests call stores and buses until it has a complete copy which the active central control can use. It then requests a deferred fault-recognition program to check out the remaining call store units to determine their operational status. Any faulty units are removed from service.

There are also error programs associated with the call stores. These programs use an error counter in the central control plus selected configuration changes to identify the unit generating the errors.

### 5.6 *Emergency-Action Functions*

System troubles are normally detected by trouble-detection circuits, and the call processing ability of the system is recovered by means of fault-recognition programs as described above. This approach requires a reasonably good central processor. Even though each of the central processor subsystems is duplicated and only one of the duplicated subsystems may be faulty, the fault-recognition programs just described depend upon the active central processor to recover a working system.

To recover from situations where a "sane" central processor is not available, a combined circuit-program facility, labeled "emergency action," has been designed. When trouble is detected within the central control, program store or call store, a 40-millisecond emergency action timer is started. The fault-recognition programs should be able to recover the call processing ability of the system within this interval. However, if the fault-recognition program is not successful, the 40-millisecond emergency-action timer will time out and activate the emergency-action circuit.

The emergency-action circuit establishes various combinations of data processor subsystems without reliance on program instructions. Program instructions are used to determine whether or not the assembled central processor is sane. This program performs a series of tests on the central processor subsystems involved in the new configuration. The program is designed as a maze. To qualify, the central processor must proceed through the maze via one and only one correct path. The rearrangement of subsystems is accomplished by a logic circuit which selects, one at a time, all combinations of central control, program store and program store bus systems. For each selected configuration, the maze program is started and a sanity timer is activated. As long as the maze

program proceeds through the predetermined course, the sanity timer is program reset. When the maze program is completed, the selected configuration is considered sane. On the other hand, if the program strays off course, the sanity timer is not reset and will time out after 0.704 millisecond (128 cycles). The timeout produces an input signal to the emergency action circuit which selects a new program store-bus-central control configuration. This procedure is repeated until the maze program qualifies a configuration as sane.

Since the emergency-action facility is provided as a back-up to the fault-recognition programs, this facility must be made as independent of normal central processor operation as possible. For example, the emergency-action facility is not dependent on the system clock. The emergency-action hardware generates its own pulses that sequence the emergency-action circuit through its actions.

The initial activation of the emergency-action circuit begins a cycle. Subsequent input signals produce state changes advancing a four-stage state counter. This counter records the successive enable signals within a cycle and directs the specific actions which are to be carried out for each timeout.

As the emergency-action sequencer advances through its various states it switches through all possible combinations of central controls, base program stores, and buses. The configurations established by the emergency-action sequencer during each state are summarized in Table III. Note that during the initial state 0000 and during states 0111, 1000, and 1111 no switching is performed and only an interrupt is generated.

TABLE III — CONFIGURATIONS ESTABLISHED BY EMERGENCY-ACTION SWITCHING  
Status of Units after a Switch Performed by the Indicated State

EA State	CC0	CC1	PS0	PS1	Bus0	Bus1	Other Stores
X000	U	U	U	U	U	U	U
X001	C	C	U	U	U	U	U
X010	U	U	U	U	C	C	U
X011	U	U	A	S	A	S	T
X100	U	U	A	S	S	A	T
X101	U	U	S	A	S	A	T
X110	U	U	S	A	A	S	T
X111	U	U	S	A	A	S	T

X: don't care

A: this unit is switched active

S: this unit is switched standby

U: the status of this unit is unchanged

C: the status of this unit is complemented

T: this unit is marked in trouble.

With each activation of the emergency-action sequencer, a B-level interrupt is initiated. The configuration changes and the actions listed above are completed before the interrupt program is started. The maze program, if successful, is followed by an emergency-action recovery program which is designed to recover the call processing ability of the system.

The emergency-action interrupt program is divided into five phases. These are as follows:

- (1) basic sanity maze program,
- (2) operational checks of the central pulse distributor,
- (3) operational checks of the call stores,
- (4) bootstrap recovery of program stores, and
- (5) emergency-action evaluation.

The basic sanity testing is not designed to isolate trouble, but instead to test the ability of the active central processor to process program instructions properly. The program with the aid of a sanity timer determines if the active central processor is operable (sane).

The check of the central pulse distributor is made to ascertain that the central pulse distributor can be used successfully in the program store recovery program. The operational check of the call stores is made for similar reasons. A bootstrap recovery of the program stores is carried out only after the emergency-action sequencer has advanced to or beyond the state where the program store complex was forced into a special configuration.

The emergency-action evaluation program determines the rate at which emergency actions are occurring and attempts to determine the cause of any recurring cycles. It also requests subsequent maintenance program actions to isolate faulty units and to return the fault-free units to service.

The emergency-action circuit is activated initially by one of the following conditions (see Fig. 15):

- (a) a clock check circuit detects trouble in the microsecond clock,
- (b) a sequencer check circuit finds a locked up central control sequencing circuit, or
- (c) a "real-time" check determines that the program is out of step with a time reference provided by the emergency action.

The real-time check is a test of the normalcy of call processing by both circuits and programs. This check assures that call processing has not been limited by the exclusion of some program functions. In addition, this check compares the passage of "real time" as counted by the program to that counted by the hardware. A record of time is kept by the program on the basis of the 5-millisecond interrupts generated by a

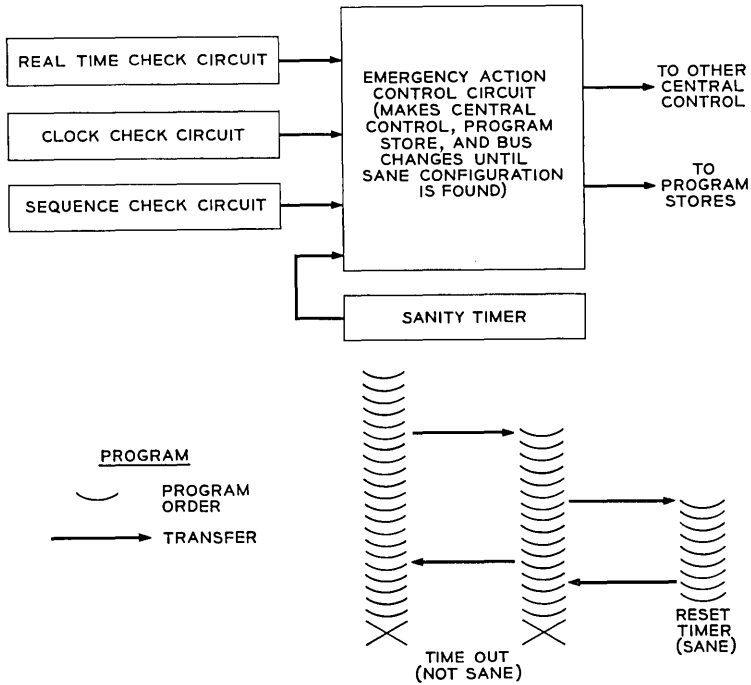


Fig. 15 — Emergency action.

5-millisecond clock. The same clock circuit supplies a signal once every 10 milliseconds to the emergency action counter. This counter, which can count up to 640 milliseconds (see Fig. 16), has two other inputs (enable and reset) which are program controlled. If the program progresses properly through its various tasks and if it stays in step with the emergency action counter, it will first reach a point (T2) where it must generate an enable signal and later a point (T3) where it must generate a reset signal. If the program goes astray and fails to generate either or both signals, the emergency-action counter will time out and activate the emergency-action sequencer within 640 msec.

During the 640-millisecond period, call programs operate on base level and on interrupt levels H and J. The proper recurrence of the interrupt level is reflected in the program count of time and in proper operation of enable and reset inputs to the emergency-action real-time counter.

Associated with each priority level (A through E) of the base level is a reserved location within the call store which is set to the "1" state after the priority level has been visited and each of the jobs at that

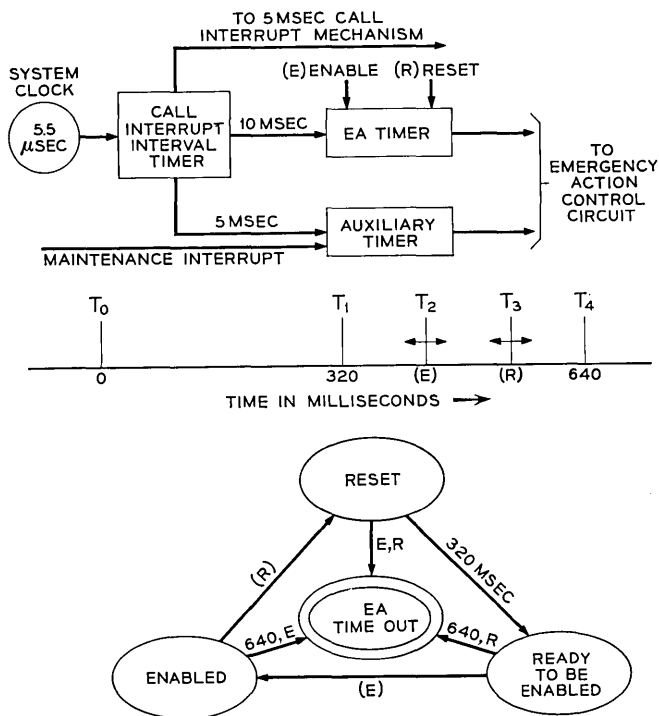


Fig. 16 — Emergency-action timers.

level has been performed. The real-time check program examines these call store locations to determine whether the base-level work is proceeding normally. If these locations indicate the failure to complete the normal amount of work, the real-time control program initiates an overload control. The over-load control slows down the acceptance of new work to allow the assumed backlog of work to be completed. If the overload control fails to recover the normal visits to the base-level jobs, the real-time check assumes that the failure to cycle through base level is due to mutilated data in call store. A reinitialization of vital call store constants is made. If the trouble condition continues, further reinitialization of data is carried out.

### 5.7 Implementation of Diagnostic Programs

#### 5.7.1 Central Control Diagnosis

The object of the central control diagnosis is to isolate faults in the standby central control to a small number of replaceable circuit packs.

This is accomplished by performing a series of tests on the standby central control, recording the results of these tests and printing them out on the teletypewriter. The faulty packages are identified by locating the printout in the maintenance dictionary.

In the fault-recognition program, each central control in effect tests itself by performing logical operations and conditional transfers. This technique was employed since it was not known which central control, if either, would be faulty. However, in the diagnosis it is known that the standby central control is faulty and that the active central control is fault-free (assuming only one central control is faulty). Knowing this, a different (more reliable) testing technique can be employed. In this environment the active system can be used to test the standby central control.

To perform a test, one wishes to apply certain inputs to the circuit under test, observe the outputs, compare them with expected outputs, and record results indicating which tests passed and which failed. The facts that the central controls are complete duplicates, are capable of running in synchronism, and that we have a way of comparing their operations (the match circuits) can be used to advantage in testing the standby central control.

The principal testing technique employed in the central control diagnosis is to force the two central controls to execute the same test program and to compare certain critical points using the match circuits. A test result or results are then recorded for each match operation. A mismatch indicates a failure for the operation being tested, and a match indicates that the test passed. In this testing mode, the mismatch sampling order is used to sample the desired information at the desired time. Thus the most frequent type of test in the central control diagnosis is of the following type:

- (1) the central controls are forced into step;
- (2) a test program is executed which completely exercises a specific hardware function of the central control;
- (3) at a number of times during this program, the output (or the nearest available output) of the circuit being tested is sampled using the match circuits and the mismatch sampling order;\*
- (4) as the sampled matches are executed, the results of these matches are recorded; and
- (5) at various points, the central controls are forced back into step to ensure meaningful match results.

---

\* This can be done without depending upon the standby central control to execute the mismatch sampling order properly by operating the standby central control in the directed match mode and the active central control in the mismatch sampling mode.



The programs which actually test the standby central control are divided into subroutines labeled "test phases." A test phase exercises a specific hardware area of central control. For the central control there are 20 test phases. The test phases, defined by the circuits which they test, are:

- (1) power and clock circuits,
- (2) start, stop and control word reception,
- (3) alternate routes to hardware in phase 2,
- (4) buffer order word register — error detection and correction,
- (5) index adder and index adder registers,
- (6) index registers and bus circuits,
- (7) decoders,
- (8) homogeneity and transfer logic,
- (9) program address incrementing,
- (10) arithmetic and logic circuitry of the accumulator register,
- (11) memory operations,
- (12) sequencing circuits,
- (13) buffer bus registers,
- (14) parity generators and checkers,
- (15) maintenance circuits (matchers, etc.),
- (16) call store address circuits,
- (17) program store address circuits,
- (18) scanner answer circuits,
- (19) enable control, and
- (20) peripheral address circuits.

The phases are performed in the order indicated, except that at certain points the diagnosis is terminated if failures have been detected. The first phase consists of scanner operations which check the power state of the standby central control to determine whether all voltage regulators are functioning normally and whether the microsecond clock appears normal. The above circuits have dc trouble detectors associated with them which are in turn connected to scan points. If a failure is detected in phase 1 no further tests are performed. Phase 2 tests the ability of the standby central control to receive control words from the active central control. The control write facility is used throughout the diagnosis to set up the match control circuits of the standby central control and to get the two central controls in step by control writing into the program address register (PAR). This phase is performed using both call store buses if they are available. If a failure is detected in phase 2, phase 3 is performed and the diagnosis is terminated, with the remaining test results being recorded as all-tests-pass.

Phase 4 tests the buffer order word register (BOWR) and the error

detection circuits. This phase is performed from both program store answer buses if they are available. If a failure is detected, the diagnosis is terminated.

Phase 5 tests the index adder circuits, and phase 6 tests the index registers and bus logic circuitry. If a failure is detected in either of these phases, phase 7 (which tests the decoders) is performed, and the diagnosis is terminated. These phases test registers which must be used in testing all circuits which follow; the diagnosis is terminated to avoid inconsistencies created by initial conditions described earlier.

If the above phases find no trouble, phases 8 through 15 are performed. These phases test all internal circuits which are not associated with external buses. If any failures are detected in these phases, the diagnosis is terminated after phase 15.

If the fault has not been located by any of the preceding phases, phases 16 through 20 are executed. These phases are again concerned with circuits associated with external buses. Each of these phases is performed from both of the duplicate buses associated with the circuits being tested, unless one is not available. If a failure is detected in a phase, the diagnosis is terminated at the completion of that phase.

It is estimated that the central control diagnostic program requires approximately 6,000 program words, and generates approximately 2,000 bits of test results. The 2,000 bits of information will normally not be printed out on the teletypewriter. Instead, a number generation program (to be described later) will operate upon this data and print out a much smaller, easier to handle, number.

### *5.7.2 Program Store and Call Store Diagnostic Programs*

The program store and call store diagnostic programs are similar in function and in design. Both of these programs attempt to locate a fault within a store which has previously been found faulty. This objective is accomplished by performing a series of exercises on the faulty store and recording the results of these exercises.

Since it is necessary to be able to diagnose a store with one bus out of service and with one central control out of service, the exercises are performed using the active central control and active bus. Most of the testing is performed using special maintenance orders provided for this purpose. Using the maintenance orders, only the store specified by the order responds, and it sends its answers back on the bus from which it receives, regardless of the state of the answer routing flip-flops. When the faulty store is being tested it is normally connected to receive from the active bus but to send on neither bus for normal orders. Thus, for the

case of program stores, the active central control receives its instructions from a fault-free store connected to the active bus; yet it can also interrogate the faulty store by reading data words using the maintenance orders. Similarly, for the call stores the active central control uses a good set of call stores connected to the active bus for storing and reading temporary information, and it gains access to the faulty store for test purposes by using the maintenance orders.

As in the central control tests, the store tests are divided into meaningful blocks called "phases" which test various blocks of the store circuits. If both buses are available, all of these phases are performed from both buses.

### *5.8 Routine Exercise Programs*

The routine exercise programs are basically of two types: those which check to see that various memory items (both call store and flip-flop) are updated, and those which exercise hardware which is not used in normal system operation.

The automatic programs have periodic schedules. There are three classes of automatic programs, where the class is determined by the scheduling technique. Class I programs are rigorously scheduled at a relatively high frequency. These programs are entered from the high-priority main program regardless of the office traffic. They must of necessity be fairly short, since they are performed religiously, even during the busy hour. An example of a program assigned to this class is that program which interrogates and resets the store error counters. This must be done on a strict schedule to ensure meaningful interpretation of the contents of the counters.

Class II programs are also rigorously scheduled, but at a much lower frequency. Programs which must be performed every hour, or at some specific time during the day, are assigned to this class. An example of a program in this class is a program for testing the emergency action circuit. This test should be performed only when traffic is low and consequently would be scheduled daily at 2 a.m. or some other nonbusy hour.

Class III routine exercise programs are the lowest-priority programs in the system. These exercises are performed in system spare time when no other jobs are waiting. These exercises are ordered in a circular list, so that when one is completed, the next one in the list is initiated. Most of the routine exercises are assigned to this class. Some examples of routine exercises in this class are:

- (1) a program to exercise the match circuits of the central control to

insure that they are capable of detecting a mismatch and that they will operate correctly in all available modes

(2) a program to exercise the error detection and correction circuits of the central control

(3) programs to check that the stores will respond properly to all maintenance orders

(4) programs to change the interconnection configurations of duplicate units

(5) programs to verify and update status words in temporary memory to insure that they agree with the actual system status.

The demand exercise programs are initiated upon request. The request can be initiated by the teletypewriter or by some other program. All automatic programs can be requested as demand programs. Some examples of demand programs are:

(1) a program to remove a unit from service

(2) a program to restore a unit to service

(3) a program to print out the status of a particular unit

(4) a program to print out the contents of a specified call store location, etc., and

(5) programs that audit the network memory.

All of the maintenance programs are normally executed by the active data processor or by both the active and standby systems. Tools in the form of demand exercise programs will also be provided to allow the execution of almost all of these programs on a repeated basis by the standby system. The need for this ability may arise if a marginal trouble develops which is not detected by the diagnostic programs. For troubles of this type it may be desirable to execute some maintenance programs continuously in the standby system while maintenance personnel make observations with an oscilloscope or some other manual tool.

Circuit tools are available, and program tools will be provided, for this purpose. To run the standby central processor independently of the active central processor (i.e., off-line), all that is required is to interconnect the central controls, call stores, and program stores so they operate as two independent systems. As described in Section III, with complete duplication this ability exists. The start-stop control, control write facility, and the breakpoint match mode also provide circuit tools which allow the active system to start the standby system in any program and to stop it at any desired address.

For example, if it is desired to execute some program continuously, starting at address A and ending at address B, this could be accomplished in the following manner. An input message would request that the

standby system execute the program from address A to address B. This demand exercise would modify call store and program store configurations to set up two independent systems. The active central control would stop the standby central control and control write the start address A into the standby program address register. It would then set up the breakpoint match mode to monitor address B with the interrupt and stop standby options specified. Next, it would start the standby central control and return to call processing.

The standby system would then begin executing program at address A while the active system ran its normal call programs. When the standby system reached address B, the active matchers would be alerted, stop the standby system and interrupt the active system with a G-level interrupt. This interrupt program could restart the standby system at A and return to call processing, etc.

## VI. MAN-MACHINE RELATIONSHIP

### 6.1 *Reaction to Trouble*

Let us now review briefly the facilities through which the maintenance man normally communicates with the system. Most of these facilities are included in the four frames which are called the "master control center." When a failure occurs in the system it is most frequently detected by circuits. Programs are immediately brought in to remove the faulty unit from the system and to establish a working system configuration. At a later time, diagnostic programs are run on the faulty unit. For example, assume that a scanner controller has failed, has been switched out of service, and has been diagnosed. At this point, the office alarm system will sound an audible minor alarm and light lamps that direct the maintenance man to the master control center. At the master control center a maintenance teletypewriter prints out the identity of the faulty controller and also a code which the maintenance man, with the aid of a dictionary, translates to the location and identity of the faulty package. The man, using this information, will locate the frame containing the faulty controller. At the frame a red trouble lamp will indicate which controller within the frame is out of service. By pushbutton he will remove the power from the controller and replace the faulty package. With another pushbutton he reapplies power. This action signals the system to start a diagnosis on the controller. If the diagnosis passes, the system will extinguish the trouble light at the controller as a signal that the controller is fault-free. Fig. 17 illustrates this flow of actions.

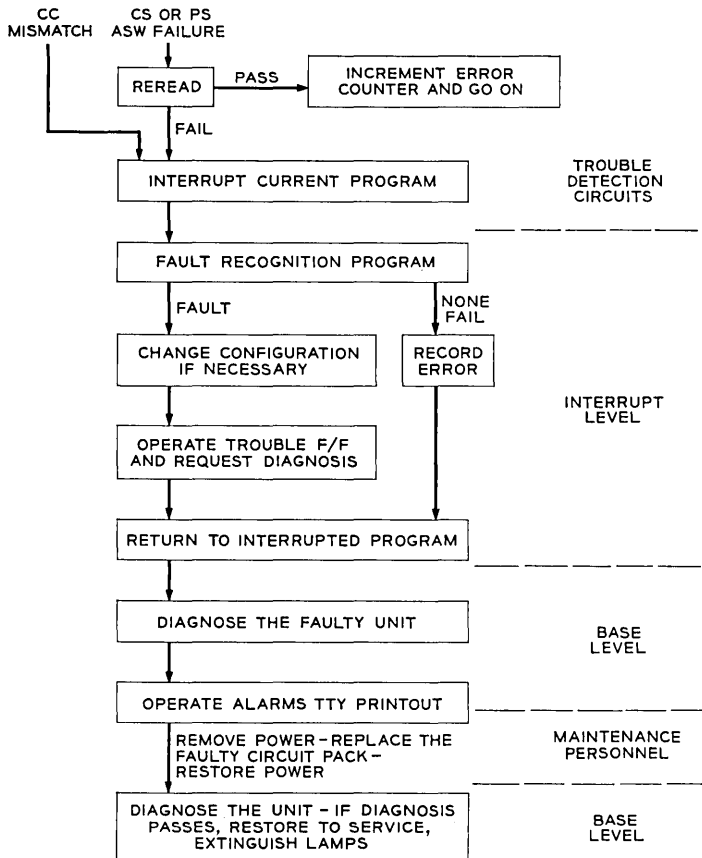


Fig. 17 — Maintenance reaction to trouble.

### 6.2 Maintenance Dictionary Production

The maintenance dictionaries are used for translating diagnostic program output, as printed out on maintenance teletypewriters, to specific package locations. Because of the complexity of No. 1 ESS and because of the number of diagnostic tests, it is considered infeasible to produce complete and accurate dictionaries by logical reasoning alone, i.e., to predict the reaction of each test to every possible fault. The method that will be used to produce dictionaries is: each plug-in circuit pack will be replaced by a fault simulator which will introduce every possible type of single fault on the replaced package one at a time and

then record the system reaction on a high-speed output tape.\* From this record, the dictionaries can be produced by sorting and ordering the test results using auxiliary data processing equipment. In addition to the relatively complete and accurate dictionaries which will result, this method also provides early feedback for evaluation of the maintenance plan — feedback which otherwise would take years to collect from operational experience. At the time diagnostic data for dictionaries are collected, additional data will be collected on key program reactions and decisions. The kinds of data which can be collected are: the system configuration changes, the length of time required by the various phases of the program, program interactions, and interrupts generated. These data can be evaluated to find any weaknesses of the maintenance programs and also to find redundant or troublesome programs.

The data generated by diagnostic programs consist of a binary word of  $n$  bits. Each bit may represent the result of a test (pass or fail), or it may take several bits of information to represent the result of a test. A simple way of organizing the dictionaries is to convert these binary words to decimal numbers, place them in numerical order, and list next to each number the faulty packages that generated that number. This is essentially what was done for the Morris ESS central control dictionary.<sup>16</sup> This dictionary, although somewhat bulky (it consisted of some 1200 double-thickness pages), was quite satisfactory when the printout in the field exactly matched a dictionary entry. When faults were introduced into the Morris central control, it was found that approximately 60 per cent produced test results that exactly matched the dictionary entries. Another 16 per cent of the faults produced entries that could be located by using some relatively awkward interpretative rules. There are many factors that can cause the diagnostic results for a given fault to differ from one machine to another or from one occurrence to another. Test results may depend on the memory state of the machine at the time the trouble occurs,† and variations in component and voltage values in two systems may be sufficient to produce different results. It is expected that a similar situation may exist in No. 1 ESS, although to a lesser degree. Consequently, it is desirable to present diagnostic data in the dictionaries in a form such that if an exact match cannot be found, the dictionary entry (or entries) most nearly resembling the diagnostic printout can be easily located.

---

\* This process has to be carried out only once unless changes are made in the maintenance program or in the circuits.

† As stated previously, in designing diagnostic tests extensive efforts are being made to avoid this problem.

The diagnostic data can be given a geometric interpretation by considering the given test result as a point in a multidimensional space where the binary word that represents the test results gives the coordinates of the point. The method that will be used in No. 1 ESS to generate diagnostic dictionaries takes an advantage of the fact that (1) of the many possible test patterns relatively few will occur\* (the space is sparsely populated) and (2) that the significance of individual tests varies. The dictionary entries will express the weighted (according to the significance of the tests) distance of each point from preselected reference points, rather than specifying the exact coordinate of the test result. This method will yield (1) a more compact dictionary, since the distances can be expressed more concisely than the actual coordinates, and (2) a dictionary in which similar patterns can be easily located when an exact match cannot be found. The methods of diagnostic data collection and dictionary generation will be subjects of a future article.

#### VII. CONCLUSIONS

The design of maintenance programs has been based on the logical analysis of circuit diagrams rather than on actual experience with the system. Because of this and the complexity of the system, it is expected that the first design will not completely meet all of the objectives. The data produced for the dictionaries will provide early feedback on the strengths and weaknesses of the maintenance plan. This feedback should enable us to evolve a design which will meet all of the design objectives. It is also hoped that with adequate feedback the 47,000-word maintenance program can be substantially reduced.

#### VIII. ACKNOWLEDGMENTS

To design a central office which is both reliable and maintainable has required the complete cooperation and awareness of everyone connected with the design of No. 1 ESS. The authors would like to pay special tribute to their coworkers in the No. 1 ESS maintenance planning department whose work is summarized herein.

#### REFERENCES

1. Keister, W., Ketchledge, R. W., and Vaughan, H. E., No. 1 ESS System Organization and Objectives, B.S.T.J., this issue, p. 1831.

---

\* For example, in the No. 1 ESS central control approximately  $2 \times 10^5$  faults will be introduced; the diagnostic program consists of approximately 2000 tests. Thus, assuming all tests are independent, there are  $2^{2000}$  possible test patterns, of which only  $2 \times 10^5$  will occur.



2. Harr, J. A., Taylor, F. F., and Ulrich, W., Organization of No. 1 ESS Central Processor, B.S.T.J., this issue, p. 1845.
3. Cagle, W. B., Menne, R. S., Skinner, R. S., Staehler, R. E., and Underwood, M. D., No. 1 ESS Logic Circuits and Their Application to the Design of the Central Control, B.S.T.J., this issue, p. 2055.
4. Ferguson, J. G., Grutzner, W. E., Koehler, D. C., Skinner, R. S., Skubiak, M. T., and Wetherell, D. H., No. 1 ESS Apparatus and Equipment, B.S.T.J., this issue, Part 2.
5. Chevalier, J. G., and Eisenhart, R. K., No. 1 ESS Circuit Packs and Connectors, B.S.T.J., this issue, Part 2.
6. Danielson, D., Dunlap, K. S., and Hofmann, H. R., No. 1 ESS Switching Network Frames and Circuits, B.S.T.J., this issue, Part 2.
7. Freimanis, L., Guercio, A. M., and May, H. F., No. 1 ESS Scanner, Signal Distributor, and Central Pulse Distributor, B.S.T.J., this issue, Part 2.
8. Dougherty, H. J., Raag, H., Ridinger, P. G., and Stockert, A. A., No. 1 ESS Master Control Center, B.S.T.J., this issue, Part 2.
9. Connell, J. B., Hussey, L. W., and Ketchledge, R. W., No. 1 ESS Bus System, B.S.T.J., this issue, p. 2021.
10. Ault, C. F., Gallaher, L. E., Greenwood, T. S., and Koehler, D. C., No. 1 ESS Program Store, B.S.T.J., this issue, p. 2097.
11. Feiner, A., and Hayward, W. S., No. 1 ESS Switching Network Plan, B.S.T.J., this issue, Part 2.
12. Genke, R. M., Harding, P. A., and Staehler, R. E., No. 1 ESS Program Store — A 0.2-Megabit Ferrite Sheet Memory, B.S.T.J., this issue, p. 2147.
13. Biddulph, R., Budlong, A. H., Casterline, R. C., Funk, D. L., and Goeller, L. F., Line, Trunk, Junctor and Service Circuits for No. 1 ESS, B.S.T.J., this issue, Part 2.
14. Harr, J. H., Hoover, Mrs. E. S., and Smith, R. B., Organization of the No. 1 ESS Stored Program, B.S.T.J., this issue, 1923.
15. Carbaugh, D. H., Drew, G. G., Ghiron, H., and Hoover, Mrs. E. S., No. 1 ESS Call Processing, B.S.T.J., this issue, Part 2.
16. Tsiang, S. H., and Ulrich, W., Automatic Trouble Diagnosis of Complex Logic Circuits, B.S.T.J., 41, July, 1962, p. 1177.



# No. 1 ESS Bus System

By J. B. CONNELL, L. W. HUSSEY and R. W. KETCHLEDGE

(Manuscript received January 15, 1964)

*Communication between the various units within No. 1 ESS demands a complex network of transmission lines. The requirements are radically different from previous offices because a large amount of information must be transmitted at high speed, in digital form, between a multiplicity of locations with a high degree of reliability.*

*A description is given of objectives and of organization and operational modes for the network. The problems involved are discussed together with the hardware by which the objectives were successfully realized.*

## I. INTRODUCTION

The function of the bus system of No. 1 ESS is to provide intramachine data and control communication. The need for large amounts of such communication is a natural outgrowth of the functional unit structure of the over-all system. At the same time the interconnection method must be highly flexible to permit easy growth and to adapt to large variations between different installations.

Existing electromechanical switching systems send most of their intramachine data by dc ground-return signaling. Most relays used in telephony are relatively easy to control at a distance because of their limited frequency response and high signal threshold. Semiconductor logic circuits, on the other hand, have a wide frequency response and can be disturbed by relatively small noise signals.

It became apparent quite early in the development that conventional interconnection techniques would not be satisfactory. The No. 1 ESS bus system described in this paper has proved to be a successful solution of the interconnection problem.

### 1.1 *Size*

Telephone switching systems tend to be relatively large assemblies of equipment consisting of many separate functional units. Compared to

previous systems, No. 1 ESS is physically smaller and is composed of fewer separate functional units. However, even a small 5000-line No. 1 ESS might typically contain 3 temporary memories, 2 permanent memories, 2 central controls, 2 master scanners, 2 central pulse distributors and a master control center plus 19 switching network, trunk or junctor units — a total of over 30 separate functional units. A very large office would contain hundreds of such units. Similarly, while distances between units may be only a few tens of feet in a very small office, a very large office will require interconnecting leads hundreds of feet long.

To assemble the functional units into a working system requires interconnections which can transmit data and control information between units with the utmost dependability. In most cases a multiplicity of sources and sinks is involved. For example, either central control must be capable of sending orders to hundreds of network controllers.

### 1.2 *Duplication*

All system units required to provide continuous service are duplicated. This includes the bus system. When one or more functional units are out of service the effective interconnection pattern must be modified appropriately.

This requires that the buses provide suitable interconnections both normally and under trouble conditions. The bus structure must permit assembly of a fully operational system in the presence of any pattern of faults that does not include simultaneous failures of both units of a duplicate pair.

Through the use of duplicated buses and provision of multiple-source, multiple-sink capability, the buses become a means for achieving the necessary duplicate switching.

The Morris, Illinois, trial<sup>1</sup> demonstrated that, to avoid any disturbance of telephone service when a functional unit fails, the duplicate switching must be performed rapidly. Relays used in the Morris trial equipment to switch service to duplicates were found to be too slow. Thus duplicate switching at electronic speeds appeared necessary. To achieve this goal, the bus system provides all interconnection facilities that the system will require under any conditions. At any instant only a limited set of these possibilities will be in use. Each functional unit can be instructed to receive on a particular bus or its duplicate, to send on a particular bus or its duplicate, or in some cases to send on both. Means are also provided to disable the sending circuits so that false pulses due to trouble conditions cannot destroy the usefulness of the bus to the other units it serves in multiple.

### 1.3 *Speed*

Cycle times for the control actions of No. 1 ESS are 5.5 microseconds. In a typical cycle, a read operation in the program store can occur simultaneously with a read or write operation in a call store, and both of these may be simultaneous with an instruction to a peripheral unit. To keep the time wasted in propagation to a minimum, the bus lengths are kept as short as possible and the buses are word organized, with parallel transmission of bits. Maximum lengths of 125 feet between any central control and any store keep propagation times to a small but not inconsequential fraction of a cycle. For peripheral units, longer bus lengths of up to 450 feet are required in large offices.

Considerations of speed are also involved in the choice of a 0.5-microsecond pulse as the basic bus signal. Shorter pulses would be more difficult to generate, transmit and detect, while longer pulses would cost time. Also, the 0.5-microsecond pulse is short compared to the repetition time of 5.5 microseconds, and dc restoration is therefore not necessary.

### 1.4 *Environment*

A number of environmental factors complicate the bus problem. These include the use of common storage battery power and the large physical size of the system, which together cause significant dc potential differences to exist between the grounds of various units. This ground potential problem is avoided by making the bus system ac-coupled. This takes care of dc noise, but other noise sources are also important. Relay circuits in No. 1 ESS are carefully protected to minimize noise from this source. However, a No. 1 ESS may be adjacent to an electromechanical central office whose relays might cause interference. Balanced transmission, shielding where necessary, and separation of bus conductors from other leads minimize such noise pickup. In addition, buses generally include synchronizing or enabling signals that reduce noise effects by time discrimination. An enable pulse is used to activate the other bus sensing circuits. Occasional noise pulses occurring in the absence of a pulse on the enable lead will not normally be sensed.

### 1.5 *Standardization with Flexibility*

A strong effort towards standardization and code minimization has been made in the No. 1 ESS development. Thus the same techniques, circuits and components have been used for all of the high-speed interconnections of the system. However, flexibility of system arrangements has been retained. This is largely due to the multiple-source, multiple-

sink capability of the bus system and to the use of separate program store, call store and peripheral unit buses. This flexibility is particularly important when additions are made to a working office. When a new equipment unit is added, the appropriate buses are easily extended to the new unit. The bus duplication is used to make the extensions without service interruption.

## II. INTERFRAME COMMUNICATION SYSTEM ORGANIZATION AND OPERATION

### 2.1 General Organization and Growth

A simplified diagram of the No. 1 ESS System is shown in Fig. 1. The input-output units represent a wide range of physical units: the space-division network controllers, ferrod scanner units, trunk and junctor signal distributor units, message accounting tape units, and teletypewriters. In its simplest form, the central control complex is capable of only one operation with these equipments during any particular input-output cycle. As a result, all such units are designed to work from one master bus system called the "peripheral bus system." This peripheral bus system is expandable to a very large number of input-output units.

Fig. 1 shows duplicate central control units connected to the above mentioned peripheral bus system. It also shows them connected to a group of program stores via a program store bus system and to a group of call stores via a call store bus system.

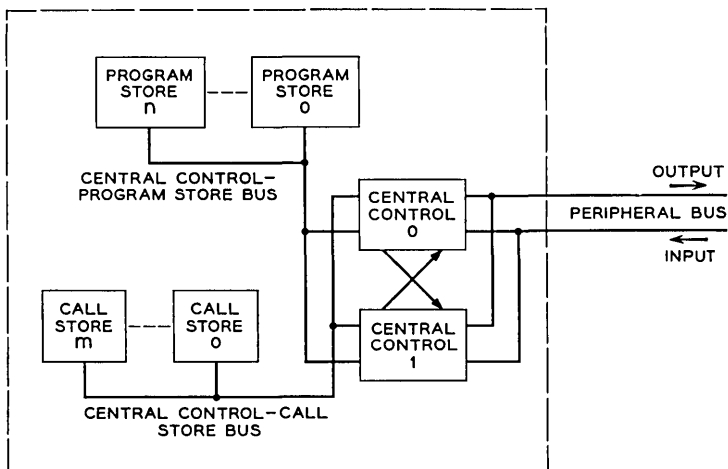


Fig. 1 — Simplified system diagram.

All of the central control operating programs and some rarely changed data (translation data, primarily) are maintained in the program stores. The call store contains data normally modified as the result of processing. By using, with some exceptions, separate memory systems for instructions and for data it is possible to utilize parallel memory operations and instruction overlap operation. This permits more data processing per unit of real time, which in turn increases the call handling capacity of the system.

There are groups of leads interconnecting the central controls shown in Fig. 1. The central controls operate in parallel, and these leads are required for synchronization and for maintenance matching.

2.2 *Bus Control*

Each of the three major bus systems can be thought of as a separate subsystem. The major difference between a bus subsystem and any of the other subsystem units, for example a program store, is that the bus is physically distributed over many equipment frames and its control, although conceptually centralized, is similarly distributed over many units. All bus operations are initiated by the central control. It is the central control that requests information from the program store, reads or writes in the call store, and requests input information or transmits instructions to units on the peripheral bus. Fig. 2 depicts the basic problem for a typical bus system. The central controls are shown on the left of the figure. The buses are shown duplicated, and each consists of two one-way groups of twisted wire pairs (designated "address" and "answer" groups). Each one-way group contains one or more functional groupings, although no such breakdown is shown on this figure. The operation of the bus is such that only one unit (source) can transmit on any bus

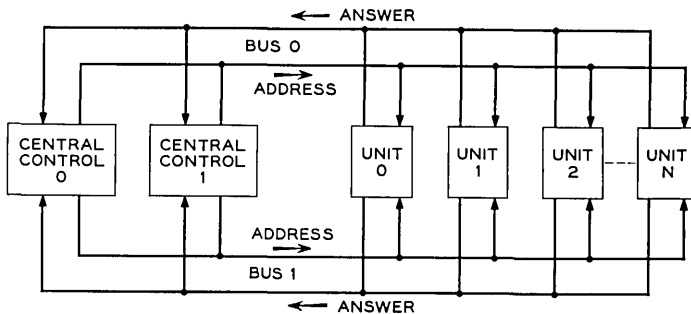


Fig. 2 — Typical bus configuration.

group (e.g., bus 0 address group). However, all the sink units can receive from the same bus. The control problems common to all the No. 1 ESS bus systems are the following:

- (1) control of the configuration between central controls and the buses for both transmission and reception,
- (2) selection of the unit on the bus for which the transmission is intended, and
- (3) control of the configuration between the units on the buses and the buses for both transmission and reception.

The solution to these three problems depends very heavily on the duplication scheme used with the equipment served by the bus system.

### *2.3 Program Store Interconnections and Operation<sup>2</sup>*

Due to two major factors, the frequency of use of the memory units and the memory address limitations of the central control, a code select method is used with the memory bus systems. Each transmission from the central control to either the program store or call store bus system is accompanied by a group of code bits (4 bits with the program store bus system and 6 bits with the call store bus system). These code bits are received by all memory units on the bus system. Each store has preset into it the codes for the two information blocks it contains. Only if a store contains the block requested in either its left or right half will it respond to the address bits. The address bits are used to specify the location in the memory block. Normally, two stores will respond to each request.

The addresses for both the program stores and the call stores are generated by the central control. These addresses are formed from a 21-bit memory address field. Table I lists the memory address spectrum assignments. If a program store is being addressed, 20 bits of the memory address spectrum are required. The 21st bit is used to determine which part of the program store word is to be used by the central control. This bit is not sent to the program stores, and it is used only when receiving data from the program store. Of the 20 bits that are transmitted as a program store address, 4 are used as information block code bits and 16 are used to specify an address within the information block.

The program store is a read-only type of memory unit. As a result, there is no data bus for writing into the program store. However, there are other bits of information sent to the program stores. Four bits are used to specify one of five program store operating modes. A synchronizing bit is also sent to the program stores. It is used to control the gating of the information bits into the program store receiving circuits. The



TABLE I — MEMORY ADDRESS SPECTRUM

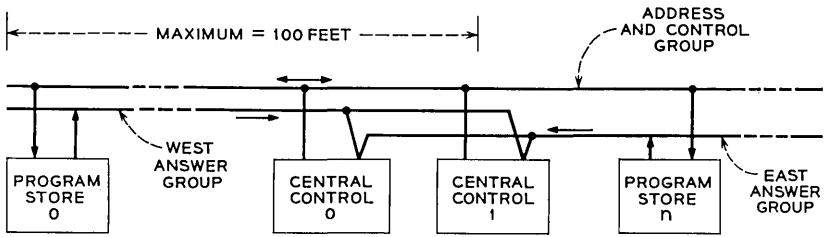
		P.S. Bus Code Bits				C.S. Bus Code Bits						
20	19	18	17	16	15	14	13	12	11-0			
0	0	0	0	0	0	0	0	0	all 0's		} buffer reg operations	
0	0	0	0	0	0	0	0	0	all 1's			
0	0	0	0	0	0	0	0	0	all 0's		} central control call stores	
0	0	0	1	0	1	1	1	1	all 1's			
0	0	0	1	1	0	0	0	0	all 0's		} reserved for future allocation	
0	0	0	1	1	0	1	1	1	all 1's			
0	0	0	1	1	1	0	0	0	0's-1's			
0	0	0	1	1	1	0	0	1	0's-1's			
0	0	0	1	1	1	0	1	0	0's-1's			
0	0	0	1	1	1	0	1	1	0's-1's			
0	0	0	1	1	1	1	d	d	0's-1's			
0	0	1	0	0	0	0	0	0	all 0's			
0	1	1	1	1	1	1	1	1	all 1's		} right half program store words	
1	0	0	0	0	0	0	0	0	all 0's		} unused	
1	0	0	1	1	1	1	1	1	all 1's			
1	0	1	0	0	0	0	0	0	all 0's		} left half program store words	
1	1	1	1	1	1	1	1	1	all 1's			

use of synchronizing pulses reduces the time during which the program stores are exposed to noise on the buses.

The central control receives 44 information bits and an all-seems-well signal from the program store. The all-seems-well signal is returned by the program store if certain conditions are satisfied during the execution of the central control request. A synchronizing signal is also sent to the central control together with the program store readout and all-seems-well signal. This is used to reduce the time that the central controls are exposed to noise on the bus system. Fig. 3 depicts the bus system and points out its ability to expand. Also shown in Fig. 3 are two separate answer buses, an east and a west. The need for two separate answer buses arose because of timing considerations and because of the directionality of the cable receivers used in the bus system. The address bus does not need to be separate, since the cable drivers transmit along the bus leads in both directions.

Fig. 4 presents a timing diagram for program store bus operations. This timing diagram shows the relative time between transmissions and receptions. A diagram showing the relative timing between program store bus operation, call store bus operation and peripheral bus operations will be presented later in this article.

Route control flip-flops located in the central controls and program



<u>ADDRESS &amp; CONTROL GROUP CONTAINS</u>	<u>ANSWER GROUP CONTAINS</u>
4 ENABLE CODE BITS	44 BITS—CONTENTS OF PROGRAM STORE
3 MODE CONTROL BITS	1 ALL SEEMS WELL BIT
1 CONTROL WRITE (CW) BIT	1 SYNC BIT
16 ADDRESS BITS	46 TOTAL
1 SYNC BIT	
25 TOTAL	

Fig. 3 — Program store bus system.

stores allow a multiplicity of program store bus configurations. Three flip-flops are used in the central controls for normal control. Seven flip-flops within each program store control the configuration between the buses and the store.

2.4 *Call Store Interconnections and Operation*<sup>3</sup>

A central control can address its own call stores, the other central control, or a group of special control flip-flop locations termed “buffer control registers.” These are all addressed via the central control index adder output register, receive data from the central control data buffer register, and transmit data to the central control data buffer register. Choice among these actions is dictated by mutually exclusive addresses

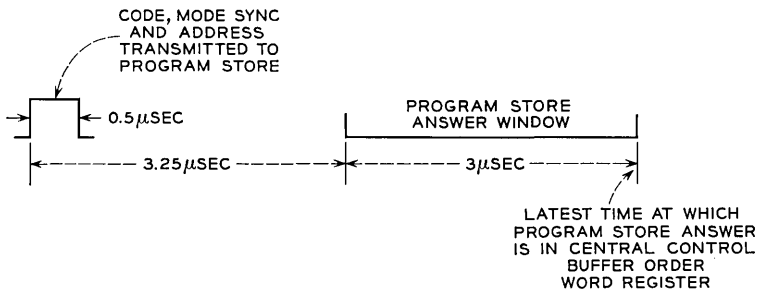
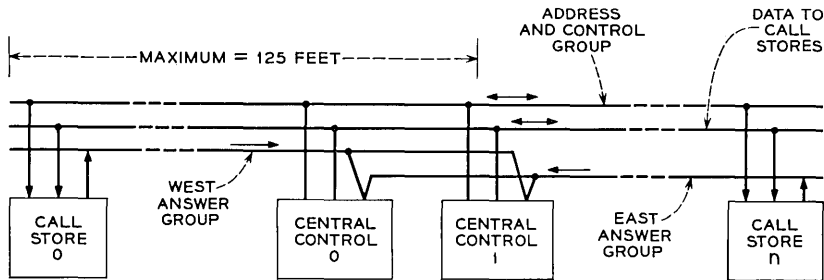


Fig. 4 — Program store system timing.

recognized at the output of the index adder output register in the central control. Table I contains the address assignments for these operations. The buffer control registers are internal to the central control and are not accessed via the call store bus system except when one central control is writing in one of these registers in the other central control. They are mentioned here for completeness.

The call store bus is duplicated. Only one bus is shown in Fig. 5. As shown in this figure, this bus contains an address and control group, an information group for writing in a unit on the call store bus and an answer group for reading from a unit on the call store bus. The address and control group transmits two synchronizing pulses, six enable code bits, three mode-control signals (C, G, H), a 12-bit address, read or write signal and a parity bit which is computed over the address and code. Two synchronizing pulses are required, since the information is sent during two separate time periods. The information group transmits a synchronizing pulse, a 23-bit information word, and a parity signal which is computed over the information, address, and code. This parity bit is stored with the word and is checked later when the word is read out of the call store. Fig. 6 shows a timing diagram for call store write and read operations. The answer bus returns a synchronizing pulse, the stored



ADDRESS & CONTROL GROUP CONTAINS

- 6 ENABLE CODE BITS
- 3 MODE CONTROL SIGNALS
- 12 ADDRESS BITS
- 1 READ BIT
- 1 WRITE BIT
- 1 PARITY BIT
- 2 SYNC BITS
- 26 TOTAL

ANSWER GROUP CONTAINS

- 24 DATA BITS
- 1 ALL SEEMS WELL BIT
- 1 SYNC BIT
- 26 TOTAL

DATA TO CALL STORE GROUP CONTAINS

- 24 DATA BITS
- 1 SYNC BIT
- 25 TOTAL

Fig. 5 — Call store bus system.

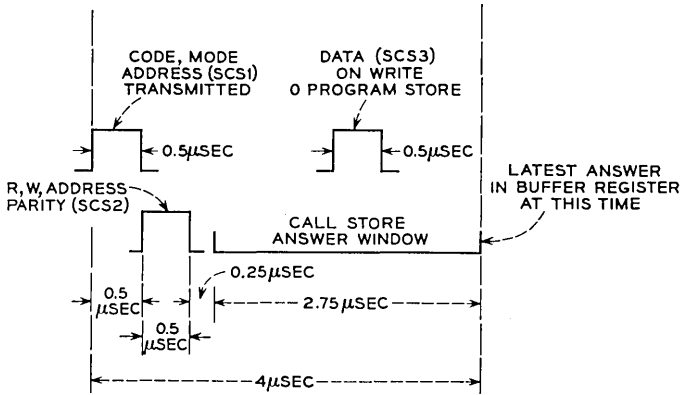


Fig. 6 — Call store system timing.

parity bit, 23 bits of information, and an all-seems-well signal. The all-seems-well signal returns only if certain conditions are met while the call store is executing a central control requested operation. There are east and west answer buses similar to those discussed with the program store answer bus. They are required for the same reasons.

Route control flip-flops located in the central controls and call stores allow a multiplicity of call store bus configurations. Three flip-flops are used in the central controls for normal control. Seven flip-flops within each call store control the configuration between the buses and the store.

### 2.5 Peripheral System Interconnections and Operation

All peripheral units will receive either data or instructions from the duplicated peripheral address bus. Peripheral units such as scanners, signal distributors, and network switch bays have duplicated controllers. Each of the two controllers can be connected to either address bus. The basic bus-to-controller logic for all such units is shown in Fig. 7. The four leads marked  $E_0$ ,  $E_1$ ,  $E_2$  and  $E_3$  are the enable leads. They activate the unit and simultaneously select the bus-to-controller path. These four signals are supplied from central pulse distributor units, which are duplicated. Two enable leads are supplied from each of the duplicate central pulse distributors. There are some units on the peripheral address bus which have unduplicated controllers. These controllers will have access to both address buses. For such units, two enable paths, one from each central pulse distributor, will be used to activate the unit and select the bus-to-controller path.

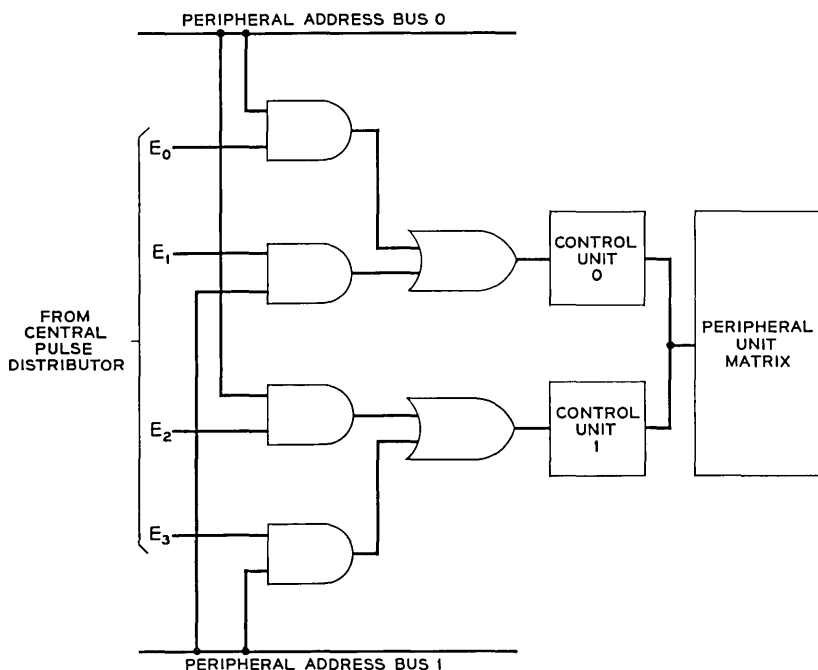


Fig. 7 — Peripheral unit enabling.

In addition to the address bus and the enable facilities for peripheral units, there is also an answer bus for units which transmit information to the central control.

The central pulse distributor outputs are used to perform one of two major functions. The first of these is the peripheral unit enable function and the second is that of providing high-speed discrete control over various flip-flops that may be located throughout the system. The enabling function is performed concurrently with peripheral address operations. As a result, a separate bus system is needed between the central controls and the central pulse distributors. When a non-enable operation is performed using the central pulse distributor, only its bus system is used. Although the central pulse distributors have a separate bus system, it is thought of as being nested into the peripheral bus system and is considered for purposes of operation as well as discussion part of the peripheral bus system.

Fig. 8 is a diagram of the interconnections between the central controls and the peripheral system, including the central pulse distributors. Shown in this figure are:

- (1) peripheral unit address bus
- (2) peripheral unit answer bus
- (3) central pulse distributor address bus
- (4) central pulse distributor verify-answer bus
- (5) central pulse distributor execute leads
- (6) central pulse distributor echo leads
- (7) central pulse distributor unipolar outputs (enable leads)
- (8) central pulse distributor bipolar outputs.

The central pulse distributor is a matrix of 1024 high-speed, low-level pulsing sources. It can provide two types of outputs; bipolar pulses and unipolar pulses. A bipolar pulse uses both polarities on a single twisted wire pair, whereas a unipolar pulse uses a single polarity on a twisted wire pair. Although the bipolar output is on a single twisted wire pair, it

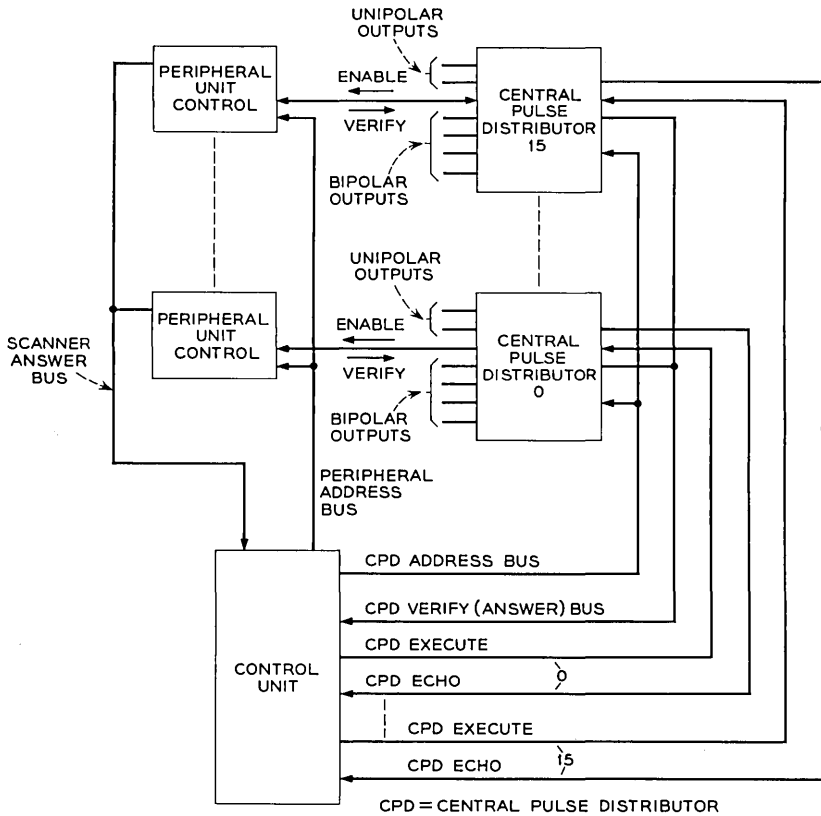


Fig. 8 — Peripheral bus system.

requires two central pulse distributor matrix points to produce the two polarity pulses. One point in the matrix is required for each unipolar output.

The central pulse distributor is effectively a one-out-of- $n$  translator for enabling purposes. As shown in Fig. 8, there can be as many as 16 central pulse distributor units. They are all accessed over a common address bus and they reply over a common answer bus. An execute translator located in the central control is used to select a single central pulse distributor. There are 16 outputs in this translator, one for each central pulse distributor. Upon reception of an execute pulse, the central pulse distributor will transmit an execute-verify signal to the central control over a separate lead. This signal is called the "central pulse distributor echo." Sixteen echo inputs, one from each central pulse distributor, connect to the central control. These signals are received and compared against the output of the execute translator whenever a central pulse distributor is used.

A central pulse distributor has a variable range of combinations of unipolar and bipolar outputs. A maximum of 512 unipolar points can be served by one central pulse distributor. This uses one half of its output matrix. The other half must be used for bipolar pulsing only and can produce 256 such outputs. The unipolar half can be adapted for bipolar use, thus affording up to 512 bipolar outputs per central pulse distributor.

Unipolar pulses are the only type used for peripheral unit enabling. The half of the central pulse distributor matrix that can produce unipolar outputs is equipped with a verification feature. Whenever a peripheral unit is being addressed via the peripheral address bus, an enable address corresponding to that unit must be sent to the central pulse distributor. This address chooses a unipolar output point, and a pulse is transmitted by the central pulse distributor on a single twisted wire pair to a peripheral unit. It is this pulse which enables a peripheral unit to receive information from the peripheral bus. After the information is received from the peripheral bus and checked for proper address, a verify pulse is returned to the central pulse distributor on the same twisted wire pair. The central pulse distributor receives this verify pulse and codes the matrix point on which it was received into the same address code format initially sent to the central pulse distributor from the central control. This is then transmitted to the central control on the central pulse distributor verify answer bus. The central control compares this answer to the address it had sent out. If a unipolar output is used for other than peripheral unit enabling and does not return a verify pulse, it is up to the central control to recognize this as the case and ignore the

central pulse distributor verify answer. The verification feature does not exist for the bipolar half of the matrix nor for those normally unipolar half outputs which are connected to provide bipolar outputs.

Binary information is used to select the proper central pulse distributor, address the central pulse distributor and address peripheral units. However, it is not transmitted to the peripheral units or the central pulse distributors in this form. It is pretranslated in the central control.

There are 14 binary bits used to address central pulse distributors. Four of these bits are used to select which central pulse distributor will be used. The ten binary-coded address bits are pretranslated by the central control into two coded groups of one-out-of-eight and one group of one-out-of-sixteen. These require 32 address bus leads. The one-out-of-sixteen group can be thought of as two groups of one-of-eight where one and only one of these groups is used for an operation. It is the division of this group that determines which half of the output matrix is being used (e.g., verification half or not). Thus only three groups of one-out-of-eight are required to specify a point in the verification half of the output matrix. The answer bus then consists of 24 leads treated as three groups of one-out-of-eight. Although more address bus leads are required with this pretranslation technique, the saving of translation equipment at the central pulse distributors is sufficient to warrant this operation.

A multiplicity of units share the peripheral address bus system. The major units — scanners, signal distributors and network switch frames — have their information pretranslated at the central control before being placed on the peripheral bus. The information for these units (addresses and instructions) is maintained in the central control in binary form. One register location is used for generating peripheral addresses in the central control; this is the addend K register. A group of translators connect between this register and the peripheral address bus. Selection of the proper translator must be made with every peripheral operation. Fig. 9 shows the information groupings for the units on the peripheral bus. Fig. 10 shows the binary form as placed in the addend K register. Information can also be placed on the peripheral address bus in untranslated binary form. Choice among translators will be discussed below. The size of the peripheral bus in terms of the number of leads is dictated by the translator coding for the network switch bays. As seen from Fig. 9, 36 leads are needed in the peripheral address bus for these units. There are two additional leads in the peripheral address bus; these are the network reset lead and the false cross and ground test lead.

The peripheral answer bus is made up of 17 leads; 16 of these carry answer information, while the 17th is an all-seems-well scanner lead.

The control of the peripheral bus system must be flexible, because of





	ADDEND K REGISTER BITS																								
	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
EXPANDED SCANNER (SINGLE WORD SCANNER)	$F_{22}$	$F_{21}$	$F_{20}$	$F_{19}$	$F_{18}$	$F_{17}$	$F_{16}$	$F_{15}$	$F_{14}$	$F_{13}$	$F_{12}$	$F_{11}$	$F_{10}$	$A_2$	$A_1$	$A_0$	$B_2$	$B_1$	$B_0$	$L_3$	$L_2$	$L_1$	$L_0$		
	ENABLE ADDRESS												SCANNER ROW ADDRESS			ANSWER BIT POSITION									
LINE SWITCH BAY 4:1	$I_2$	$I_1$	$I_0$							$\phi_3$	$\phi_2$	$\phi_1$	$\phi_0$			$C_3$	$C_2$	$C_1$	$C_0$	$S_1$	$S_0$	$L_3$	$L_2$	$L_1$	$L_0$
	INSTRUCTION									OUTPUT LEVEL						EQUIPMENT DESIGNATION			SIDE SWITCH		LEVEL				
LINE SWITCH BAY 2:1	$I_2$	$I_1$	$I_0$							$\phi_3$	$\phi_2$	$\phi_1$	$\phi_0$			$C_3$	$C_2$	$C_1$	$C_0$	$S_2$	$S_1$	$S_0$	$L_1$	$L_0$	
	INSTRUCTION									OUTPUT LEVEL						EQUIPMENT DESIGNATION			SIDE SWITCH		LEVEL				
JUNCTOR SWITCH BAY	$I_2$	$I_1$	$I_0$							$G_1$	$G_0$	$BS_2$	$BS_1$	$BS_0$	$BL_2$	$BL_1$	$BL_0$	$JS_2$	$JS_1$	$JS_0$	$JL_2$	$JL_1$	$JL_0$		
	INSTRUCTION									GRID		B-LINK SIDE SWITCH			B-LINK LEVEL			JUNCTOR SIDE SWITCH		JUNCTOR LEVEL					
TRUNK SWITCH BAY	$I_2$	$I_1$	$I_0$							$G_1$	$G_0$	$TS_2$	$TS_1$	$TS_0$	$TL_2$	$TL_1$	$TL_0$	$BS_2$	$BS_1$	$BS_0$	$BL_2$	$BL_1$	$BL_0$		
	INSTRUCTION									GRID		TRUNK SIDE SWITCH			TRUNK LEVEL			B-LINK SIDE SWITCH		B-LINK LEVEL					
SIGNAL DISTRIBUTOR												$I$	$H$	$A_2$	$A_1$	$A_0$	$B_2$	$B_1$	$B_0$	$C$	$D_1$	$D_2$			
												INSTRUC-TION		EQUIPMENT DESIGNATION			SIGNAL DISTRIBUTOR POINT								

NOTE: BLANK SPACES ARE DON'T CARES

Fig. 10 — Addend K register bit assignment.

by the central control. One is the internal instruction for the peripheral unit and is placed in the central control addend K register; the other word is the peripheral unit enable word and is placed in the central control F register. On nonperipheral unit operations, when only a central pulse distributor output is to be activated, the address of the point is placed in the central control F register and the addend K register is not used.

The F register word contains 23 bits. Bits  $F_{10}$  through  $F_{13}$  control the execute translator and select the proper central pulse distributor. Bits  $F_{14}$  through  $F_{22}$  combined with  $F_9$  are the ten binary bits from which the central pulse distributor point address is generated. The use of the  $F_9$  bit is not direct. It becomes part of the address only for non-enable operations. For enabling operations,  $F_9$  is considered equal to zero regardless of its actual value. For non-enable operations, it assumes its actual value. This bit is the one used to select which half of the central pulse distributor matrix is used.

Those central control instructions used for peripheral operations are treated as enable or non-enable instructions. The central control instruction decoder must determine which it is and whether or not a peripheral answer is expected if it is an enable-type instruction. Whenever a peripheral answer is expected, a special flip-flop designated scanner request flip-flop (SCR-FF) is set. When no answer is expected, it is reset. Whenever a non-enable operation is to be performed, another special flip-flop designated the F inhibit flip-flop (FINH-FF) is set. It is also set for a class of single-word scanner instructions (see Fig. 10). It is reset for all other peripheral instructions. The  $F_9$ ,  $F_8$  and  $F_7$  bits are used to select the peripheral address translator for all enable operations except the single-word scanner instruction. The codes for translator selection are given in Table II. The  $F_6$  bit is used to determine whether or not a verify address is expected from the central pulse distributor. The  $F_5$  bit is used

TABLE II — TRANSLATOR SELECTION CODING

$F_9$	$F_8$	$F_7$	Translator
0	0	0	short binary to peripheral bus
0	0	1	long binary to peripheral bus
0	1	0	line switch bay, 4:1 concentration
0	1	1	line switch bay, 2:1 concentration
1	0	0	junctor or trunk switch bay
1	0	1	long binary to CPD address bus
1	1	0	signal distributor
1	1	1	scanner

to determine whether or not to expect an all-seems-well scanner response. The  $F_4$  through  $F_0$  bits are reserved for maintenance use and exert no control over peripheral operations. Upon detecting a peripheral operation, the central control instruction decoder either sets or resets the SCR and FINH flip-flops and starts a peripheral control sequence circuit. This sequence circuit will control all peripheral operations. This sequence circuit uses the SCR and FINH flip-flops as well as many of the F register bits to control peripheral operations. The use of the F register as functions of the SCR and FINH flip-flops is as follows:

(1) FINH = 0 and SCR = 0 or 1, use the  $F_9$ ,  $F_8$  and  $F_7$  bits to select the peripheral translator,

(2) FINH = 1 and SCR = 1, use the scanner translator for the peripheral address bus, ignore the condition of  $F_9$ ,  $F_8$  and  $F_7$  bits with regard to selection of the translator. For this particular case, as can be seen from Fig. 10, these bits are part of the scanner address.

(3) FINH = 1 and SCR = 0, no use is made of the peripheral address bus, and the  $F_9$  bit is used as part of the central pulse distributor address selection.

The bus-to-controller path for a peripheral unit is selected by the enable pulse that it receives. A typical two-controller unit has four possible enable pulse inputs, two from each central pulse distributor of a duplicate pair (see Fig. 7). Bit position  $F_{10}$  of the central control F register is the least significant of the four central pulse distributor select bits. It is used to select between a duplicate pair of central pulse distributors. The  $F_{14}$  bit of the same register is used to select between halves of the enable portion of the central pulse distributor output matrix. Bits  $F_{10}$  and  $F_{14}$  control the selection of the one-out-of-four unipolar enabling distributor output points, two of which are located in each central pulse distributor of a duplicate pair. Table III shows the route selection for all such peripheral units. From this table it is clear that  $F_{14}$  is also used to select the proper address bus for transmission.

The F register word is called the "enable address," since it chooses the address bus and indirectly the proper controller. This word can differ

TABLE III — PERIPHERAL UNIT ROUTE SELECTION

$F_{14}$	$F_{10}$	CPD of a Pair	Peripheral Unit Controller	Address Bus
0	0	A	A	A
0	1	B	B	A
1	0	A	B	B
1	1	B	A	B

from unit to unit. For this reason, the normal procedure is to “look up” the enable address in a call store memory table. This table is termed the “enable” table.

When performing non-enable operations, the  $F_{14}$  bit is still used to control the selection between halves of whichever portion of the central pulse distributor matrix the  $F_9$  bit selects. As such, the  $F_{14}$  bit is used to determine whether or not a bipolar output pulse is the set or reset polarity. If a pair of unipolar outputs is used for such control, the  $F_{14}$  bit is used to choose between them.

Although the major peripheral bus configuration control takes place via the same mechanism as the unit selection, there are several operating modes which affect this bus configuration. Mode-control flip-flops are used to allow additional operating modes. There are two mode flip-flops in each central control for central pulse distributor address bus control (see Table IV) and two for peripheral address bus control (see Table V).

In order to properly diagnose certain classes of faults, it is necessary to provide independent operation of central controls. This independent mode is referred to as “off-line” operation. The two central controls can work with independent memory units. However, many of the peripheral units are common matrix units, and therefore two nonsynchronized machines cannot be given simultaneous control of the peripheral system. It is still necessary, however, to allow off-line operation of the peripheral system. This is implemented using two flip-flops. These are located in each central control. The central controls react to these flip-flops as shown in Table VI.

Fig. 11 is a timing diagram for peripheral operation. All transmissions in the program store and call store system were accompanied by sync pulses. These are used to reduce the time the receivers on the buses are

TABLE IV — CENTRAL PULSE DISTRIBUTOR MODE CONTROL

CPD Bus Mode Control Flip-Flops		Mode
CDMA	CDMB	
0	0	normal <sup>1</sup>
1	0	mode A <sup>2</sup>
0	1	mode B <sup>3</sup>
1	1	not used

<sup>1</sup> Normal mode: only active control unit transmits over bus designated by CPDB flip-flop.

<sup>2</sup> Mode A: only active unit transmits over both buses.

<sup>3</sup> Mode B: active unit transmits on bus designated by CPDB flip-flop — standby unit transmits on other bus.

TABLE V — PERIPHERAL BUS MODE CONTROL

Mode Control Flip-Flops		Mode
PBMA	PBMB	
0	0	normal <sup>1</sup>
1	0	mode A <sup>2</sup>
0	1	mode B <sup>3</sup>
1	1	not used

<sup>1</sup> Normal mode: only active central control transmits over bus designated by bit 14 of F register.

<sup>2</sup> Mode A: active unit transmits on both buses. Standby does not transmit on either bus.

<sup>3</sup> Mode B: active unit transmits on bus designated by bit 14 of F register; standby unit transmits on other bus.

open and susceptible to noise pulses. The peripheral bus system uses no sync pulses. Some noise protection is afforded, however, through the use of other techniques. When a peripheral unit is enabled by a central pulse distributor unipolar pulse, the mechanism that recognizes the enabling is also used to open a “window.” The address for this unit must arrive on the peripheral address bus during the time this “window” is open. The window interval is 2.5 microseconds maximum. Thus, the address and enable signals must arrive at the peripheral unit within a narrow band of time coincidence. Two factors control this coincidence. One is the time of transmission from the central control; this can be strictly controlled. Another is the cabling of the address bus and the enable leads throughout the office. To control the differential between the address and enable route in all sizes of offices, the central pulse distributor frame is used as a distribution center for the peripheral address bus. In this way the distance, and therefore the delay, between the point of transmission, the central control, through the central pulse distributor frame to the peripheral unit is the same for both the address and enable signals. Fig. 12 shows

TABLE VI — OFF-LINE OPERATION

Off-Line Control Flip-Flops		Active Unit	Standby Unit
OL1	OL2		
0	0	E & A	nothing
0	1	E	A
1	0	A	E
1	1	nothing	E & A

E = Enable operation.

A = Address operation.

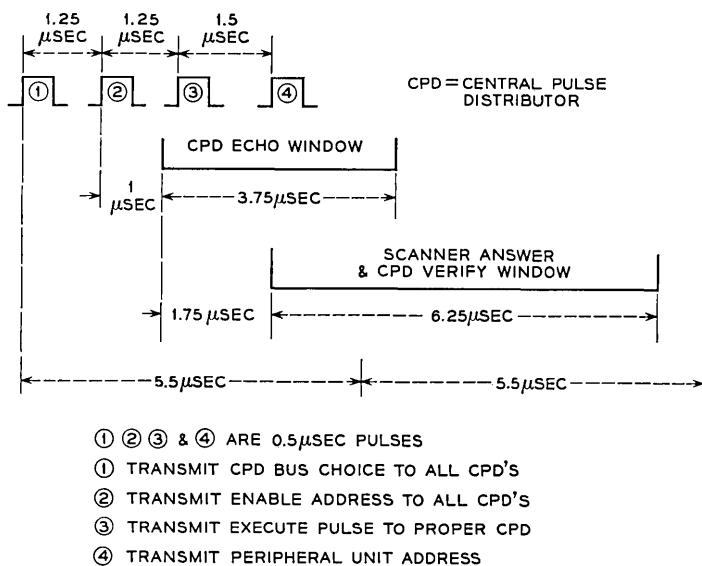


Fig. 11 — Peripheral system timing.

the arrangement in a central pulse distributor. There is provision for four branch address buses in each central pulse distributor. If a second central pulse distributor pair is required for enabling purposes, then the branch address bus used for the units enabled by the additional central pulse distributor pair must emanate from the new central pulse distributor frame.

The peripheral answer bus also passes through the central pulse distributor frames. Within these frames, the answer buses are “fanned in” from a maximum of four to one main answer bus. The main answer bus connects to the central controls. The branch address and branch answer buses give the peripheral system greater flexibility than was required with the program store and call store systems. This flexibility was not needed in these cases since there was a limited number of units on each bus system. With the peripheral bus system, the number of physical frames can run as high as 400 or 500.

With both the address and answer buses, the 0 bus is both fanned in or fanned out at the 0 central pulse distributor of a pair and the 1 bus at the 1 central pulse distributor of the pair.

In the central pulse distributor, there are many output points associated with functions other than the enabling of peripheral units. This was mentioned earlier. Some of these output points control critical

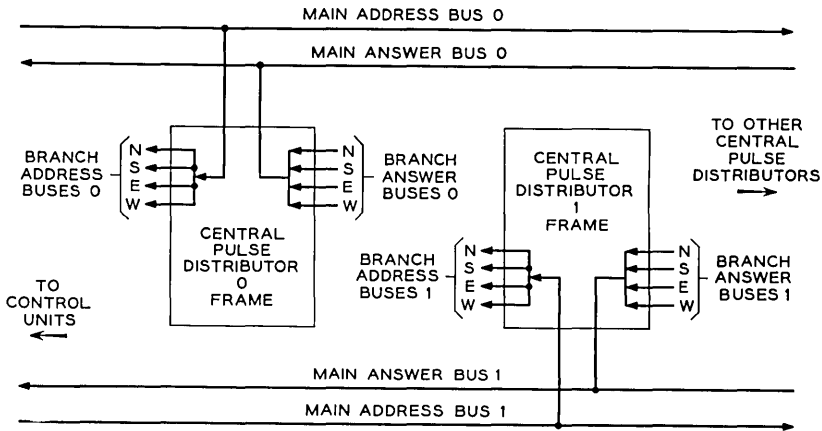


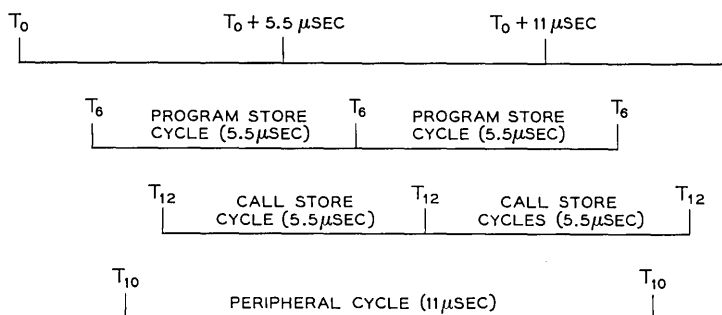
Fig. 12 — Address bus fan-out and answer bus fan-in.

system flip-flops — for example, which central control is active and which is standby — as well as the call store and program store route flip-flops. An unwanted change of these flip-flops could be caused by random noise which may appear on the wires connecting them to the central pulse distributor. A means of protecting such flip-flops has been incorporated into the system. This consists of a common synchronizing signal sent to all such flip-flops simultaneously with the central pulse distributor output pulse. Only the active central control can transmit this pulse. This common signal is designated WRMI (“we really mean it”). It is transmitted from the central control over two twisted wire pairs for duplication and synchronizes in the central pulse distributor frames with the central pulse distributor execute signal. It is fanned out much the same as the address bus.

Fig. 13 shows the relative occurrence of operations on the three bus systems. All call store and peripheral bus operations are controlled by program instructions. The program store bus system is controlled by program instructions only for transfer instructions and data readings.

Because of the necessity for speed and the use of each bus system for round-trip communications, the propagation time, and hence the length, of the buses becomes a critical factor. In the No. 1 ESS system the program store bus is limited to 100 feet and the call store bus to 125 feet from the farthest central control to the farthest memory unit. It was mentioned earlier that two bus cables are provided in both the program and call store bus systems. This is accomplished by combining two unidirectional answer buses at the central controls. The address bus is bidirectional.





$T_n = n(0.25\mu\text{SEC})$  FROM  $T_0$

e.g.  $T_6 = 6(0.25\mu\text{SEC}) = 1.5\mu\text{SEC}$  AFTER  $T_0$

THERE ARE 22 SUCH  $0.25\mu\text{SEC}$  INTERVALS IN ONE  $5.5\mu\text{SEC}$  CYCLE

Fig. 13 — Basic communication timing.

The peripheral bus system is used to communicate with many more units than either of the memory bus systems. As a result, longer bus lengths must be used. In addition, two cycles, or 11 microseconds, are used for peripheral bus operations. The distance from the farthest central control through the bus fan-out points to the farthest peripheral unit can be no more than 450 feet. An equally critical factor in the peripheral system is the differential distance between the address bus and the enabling path for the peripheral unit. This differential is minimized by controlling the cable routes.

### III. BUS CIRCUITRY

#### 3.1 *The Peripheral Address Bus*

##### 3.1.1 *Typical Pair*

Consider a typical pair in the peripheral address system, starting at one of the central controls. The twisted pair is in a standard switchboard cable and is made of 26-gauge wire with approximately one twist every 2.5 inches.

In a small office one end will go up from a central control to the cable rack, into a special shielded compartment, and over the row of frames to an end guard at the end of the row. Within this end guard are banks of resistors. The address pair terminates in one of these resistors.

The other end goes to the other central control, also by way of the cable rack. After going into the second central control and out again, it

goes, via the cable rack (as always), to a central pulse distributor. Here it has an option. In a large office it would go in and out of a CPD and might go on to do the same at several others. In the small office it is simply connected to a pair of terminals of the CPD, but does not actually go in (see Section 3.7), and then continues in turn to all the peripheral unit frames which contain scanners and network controllers. Beyond the last controller or scanner the other end terminates in a resistor on a nearby end guard.

Before considering the associated hardware it may be noted why a twisted pair was chosen. From the previous discussion, it is evident that this is a transmission line with primary considerations of transmission loss and speed, immunity to noise, reflections and crosstalk, and low cost. One obvious possibility, because of its excellent transmission properties and shielding, is coaxial cabling. However, the cost of coaxial cable is high\* and the costs of terminating coaxial wiring are worse. Fortunately, twisted pair, carefully balanced and properly terminated in 100 ohms, has characteristics which, for our purposes, are satisfactory. At the frequencies of interest, the effective delay is  $2 \mu\text{sec}/1000$  feet. The loss varies with frequency (6 db/1000 feet at 1 mc), producing the distortion illustrated in Fig. 14. As will be noted, tolerable delay limits the length to less than 450 feet where the loss and distortion are relatively small. As will be discussed later, reflections, noise, and crosstalk can be held within satisfactory limits.

### 3.2 *Grounding Inductance*

Precise balancing to ground is essential to the good noise and crosstalk characteristics required.

The balanced grounding cannot be done satisfactorily by means of the terminating resistors. This would require an expensive pair of matched resistors at the terminals. Instead, a 0.25 mh inductor is used, connected across the pair and grounded at the midpoint. A much better balance is obtained than is practical with resistors, and the ground can be located at the most desirable point near the middle of the longest buses.

The inductor is built into terminal blocks where the pairs are brought in or out of a particular frame.<sup>4</sup>

### 3.3 *Cable Receivers*

Since this pair transmits a signal from a central control to a large number of receivers in the frames which contain network controllers,

\* Even in a relatively small office, the 38 pairs in the address bus alone may add up to about 3 miles of pairs and some 3000 terminations, so interconnections are an appreciable part of the cost of an office.

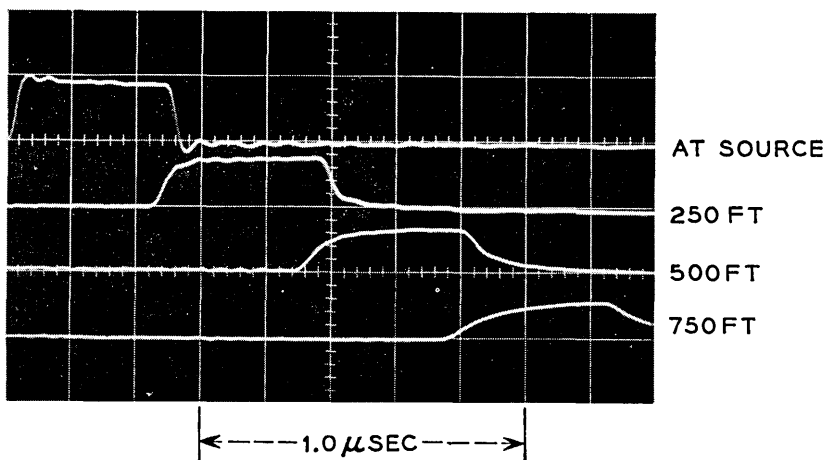


Fig. 14 — Pulse distortion by transmission.

scanners, and signal distributors, it is essential that the individual receivers should not seriously degrade the transmission characteristics of the pair. In addition, a failure in any one receiver should not interrupt transmission to the other locations.

To meet these requirements, a small current transformer is used for pick-off. The primary winding is essentially the bus pair passing through a toroidal core in such a manner as to constitute a balanced, two-turn primary winding. The secondary is a 50-turn winding.

As shown in Fig. 15, the secondary winding goes to a simple grounded-emitter transistor amplifier.

This receiver picks off not more than one per cent of the power being transmitted in the pair and amplifies it to a useful level for use in the frame. This circuit adds a small loss in the balanced pair when operating properly, and nothing which can happen in the output side of one or two pick-offs will seriously damage transmission to other frames. Under normal conditions each pick-off inserts a balanced impedance, less than one ohm, into the line. If the secondary of one transformer is either shorted or opened, the effect on transmissions is a small change in the inserted impedance. In the worst case (open secondary) the inserted impedance in the line is  $3 \mu\text{h}$ . This is harmless if only one or two are open circuited.

### 3.4 Cable Drivers

The standard pulse on the pair is produced by a cable driver. As may be noted in Fig. 16, this is a two-input logic circuit.<sup>5</sup> It is designed to

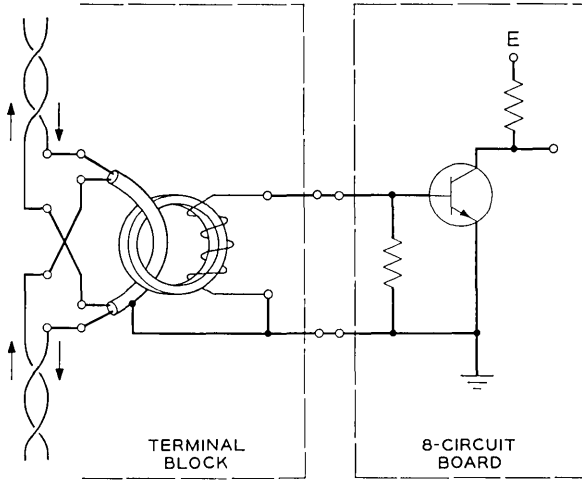


Fig. 15 — Pick-off transformer and cable receiver amplifier.

operate at a higher level than the normal logic circuitry. The two inputs are driven in coincidence by a pulse from standard logic circuits and a timing, or gate, pulse. The output transformer, designed to work into 50 ohms, is connected across the transmission line, driving the bus in both directions.

The requirement that two or more drivers be connected to the same pair and drive at a level adequate for operation of many receivers made the shunt connection the only practical one. The transformer design was the most exacting in this circuit. It was necessary that a physically small

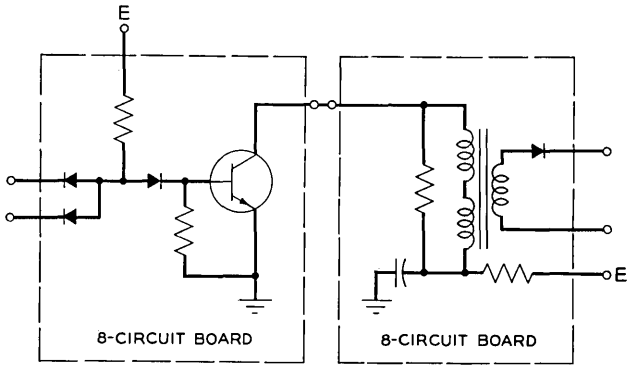


Fig. 16 — Cable driver amplifier and output transformer.

unit (mounted on a standard printed wiring board) meet the requirements: drive a 200-ma, 0.5- $\mu$ sec pulse into a 50-ohm load on a 10 per cent duty cycle. A 2:1 turn ratio is used, with 3 equal windings, to limit the transistor current to 100 ma. The inductance per winding is 1.35 mh. There were two serious problems. The impedance looking back into the transformer was not large enough to prevent deterioration of the transmission when several circuits were across the same line. This was solved by means of a series isolating diode. A second problem was a sharp reverse spike, caused by leakage inductance, at the end of an output pulse. This spike temporarily increased the transistor collector voltage above breakdown and, in extreme cases, produced noise in other channels where amplifiers were on the same board. These defects could not be cured by decreasing the magnitude of the primary damping resistor without increasing the transistor load and lengthening the transformer duty cycle. The problem was solved by trifilar winding. The result was a slightly increased capacitance (which was desirable here), a leakage inductance cut to about one-third its previous value, and a unit which could be much more readily manufactured to specifications.

### 3.5 *Noise, Distortion, and Crosstalk*

Before going to more specialized circuitry which is associated with the bus system, the situation on distortion, noise, and crosstalk may be summarized.

As noted previously, delay requirements limit cables to less than 450 feet long. Over that distance the transmission distortion is small. There is one difficulty. While the amplifiers regenerate and transmit a good pulse, they also tend to lengthen it. This becomes serious in a large office where more amplification is used. It will be discussed later.

Putting the buses in a separate section of the cable rack, shielded from the rest of the world, is, in a sense, extra insurance against noise as far as the balanced line itself is concerned. Extensive tests have been made simulating noise (such as that from unprotected relays) which might be encountered. No trouble was detected from the transmission line itself. Longitudinal noise did, however, leak into the cable receiver. The cable receiver transformer transforms from balanced to grounded transmission. At high frequencies the parasitic capacitance ruins the balance, and so a high-level, sharp spike of longitudinal noise could be transmitted through the cable receiver. This problem was solved by shielding the primary winding as shown on Fig. 15.

The greatest remaining source of noise and crosstalk is within terminal

circuitry rather than the bus itself. The circuitry is closely packed, eight circuits to a board on closely packed boards. All circuits under such conditions are prone to noise and crosstalk because of coupling and common ground connections. The amplifier circuits, because of the large currents involved, are particularly so. The solution required careful design (and redesign!) to keep common ground leads short and isolate noisy output leads from sensitive input ones.

### 3.6 *Grouping of Drivers and Receivers*

Closely analogous to noise problems are those concerned with multiple drive points and multiple reception. As described, an address pair has two or more cable drivers, grouped in the control area, any one of which may drive a multiplicity of receivers in the peripheral area. In a large office there may be up to 50 receivers scattered along each half of the pair with the drivers grouped together in the middle.

There are also answer buses which transmit from a multiplicity of peripheral points to a few receivers in the control area. Here a one-sided bus must be used. Drivers cannot be put on both sides of a receiver. The necessary polarities are such that a pulse from a driver on one side passes through the isolating diode of a driver on the other side in the forward direction. It can temporarily break down the corresponding transistor, or turn it on, producing noise and distortion on the pair. In the case where the address bus goes two directions, two answer buses must be used, going to separate receivers.

### 3.7 *Central Pulse Distributor*

There are two distinct parts to the central pulse distributor. One<sup>6</sup> decodes an enable address and sends an enable signal on private pairs to other frames. The other part is the one to which the peripheral address pair goes. This section contains a group of amplifiers which regenerate the address pulse and can transmit it simultaneously to a maximum of four outputs. In a small office this amplifier is not needed and this section is wired but not equipped. The address bus is brought only to the CPD terminals so that it will be available if the office grows.

If the expansion in the number of receivers or length of bus exceeds the capacity of one bus, then the CPD is equipped. The pulse is picked off through a standard cable receiver which drives a bus fan-out circuit (see Fig. 17). This circuit simply transformer-couples the input of two transistor amplifiers in parallel to the output of a receiver amplifier. These amplifiers, through standard transformer circuitry, drive two pairs.

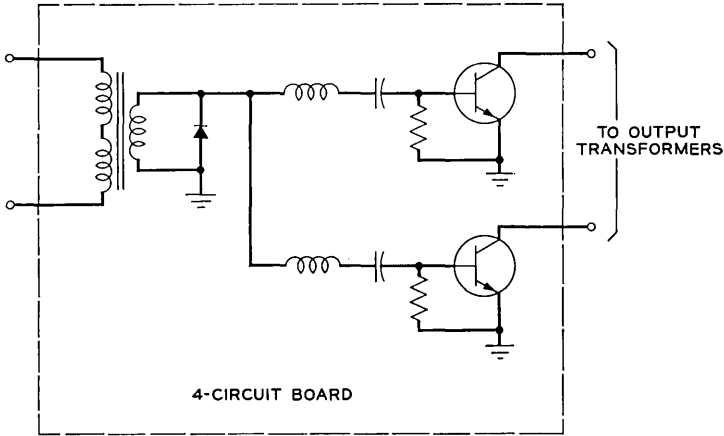


Fig. 17 — Bus fan-out circuit.

Since each end of each bus can be used, this gives fan-out from one bus to four in parallel.

The addition of this amplification results in a difficulty previously mentioned. Each time the pulse is regenerated, the output lengthens a little. At this point there is danger that the accumulated stretching will give a pulse longer than the  $0.75\text{-}\mu\text{sec}$  maximum permitted in the peripheral area. To avoid this, the pulse width as well as amplitude is regenerated. *LC* circuits in series with the transistor inputs convert the input pulses to a half sine wave with  $0.5\text{-}\mu\text{sec}$  duration. The amplifier clips this, producing a new  $0.5\text{-}\mu\text{sec}$  pulse.

One further note may be made on bus length. The delay which may be tolerated is measured from the first cable driver in the control area to the farthest cable receiver. The 450-foot maximum length must be measured in the same way from the control center, not the CPD. Since the corresponding enable pulse must coincide with the address at the receiver, the buses involved must be of equal length, within 50 feet. To insure this, they follow the same path, as far as possible.

#### IV. OTHER BUSES

##### 4.1 Answer

The answer bus sends signals between the same units as does the address bus, but in the reverse direction, so it has many cable drivers scattered along it and a few receivers in the control area. As noted, the

characteristics of the cable drivers make this a one-sided bus — drivers cannot be located on both sides of the receivers. It uses exactly the same circuitry as the address bus, with one exception. In the large office the answer bus must fan in four-to-one. This is done using a combination (Fig. 18) of receivers, logic circuits and drivers in an OR configuration.

#### 4.2 Enable Address

The enable address is decoded in its section of the CPD, and the terminating circuitry is somewhat different. The input from the enable address bus to the CPD's is picked off in the manner already described, but the outputs are on a one-to-one basis, each enable pair going to one receiver. These pairs are transformer-coupled in special circuitry<sup>6</sup> within the particular frames. As discussed in the reference, there are several types of these pairs and they may transmit the standard pulses or a bipolar pulse. The enable verify is an exception to the rule that buses are

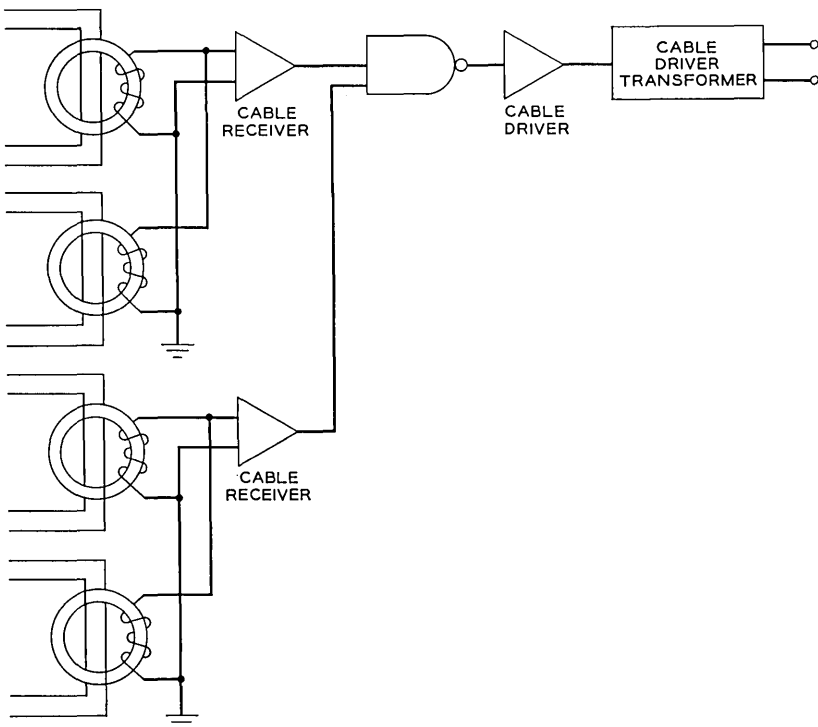


Fig. 18 — Bus fan-in circuit.



all one-way. When an enable pulse is transmitted, a verify is returned (after a brief delay) to the CPD over the same pair, using special circuitry.<sup>6</sup> From the CPD the verify pulse gets to the control center over an enable verify bus in the standard manner.

4.3 *We Really Mean It and Set Manual Pulse Bus*

There are many private pairs, similar to the enable, which are used to enable or disable various sections of the office — generally by operating flip-flops. These are operated by pulses generated in the enable section of the CPD's and the coincident gate (WRMI), which is a simultaneous pulse produced synchronously from a special fan-out circuit. Three of the four outputs are always used even in a small office:

WRMIA goes to master control center, master scanner and central control

WRMIC goes to call stores

WRMID goes to program stores.

All use standard drivers and receivers.

A special fan-out circuit is used because the WRMI pulse must be stretched when it arrives at the CPD before being gated out.

The pulse is received by means of a cable receiver pick-off and amplifier (Fig. 19). This drives a dynamic register<sup>6</sup> which stretches the pulse to 2  $\mu$ sec. During this interval the four WRMI output pairs may be pulsed coincidentally with the pulse from the enable section of the CPD. This is done by standard cable driver circuitry. The additional cable driver shown on Fig. 19 is needed to correct the polarity.

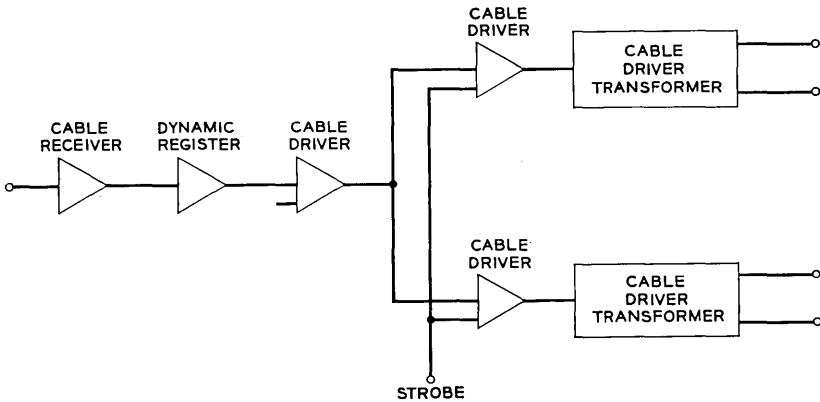


Fig. 19 — WRMI fan-out circuit.

The SMPB pair is discussed with WRMI only because it also is a lone pair — a gate or execute driven from the master control center and having receivers at central controls and program stores.

#### 4.4 *Stores*

The individual enable and status connections from the CPD's to the stores have been mentioned. The other major interconnections in this area are the call store or program store address and answer buses. For timing reasons these must be kept short — not more than 100 feet in length or more than 24 receivers per bus. There is the same sort of limitation as on the peripheral address and answer. Drivers in the control area can operate both ends of the address bus, but the answer bus must be one-way. Drivers at the stores must all be on the same side of the receivers in the control area. Standard cable drivers are used for these buses. Standard receivers are used at the central control. At the stores the cable receiver current transformers are standard, but the amplifiers are part of a more complex circuit.

#### 4.5 *Private Buses in the Control Area*

Most of the pairs between the central controls for maintenance and diagnostic purposes use standard drivers and receivers. There are also timing (clock pulse) pairs which have carefully controlled transmission characteristics. These connect directly to special circuitry within the central controls.

### V. DC CONNECTIONS

There are three types of these:

#### 5.1 *Master Scanner*

This collection of connecting pairs is used to observe the status of test points scattered throughout the office. The scanner itself is not very sensitive to noise but, because the leads are long and are exposed to many sources of noise, there is a danger that they might transmit noise to another sensitive circuit. For this reason, twisted pairs are used and the leads are carried in their own shielded section in the cable rack. They are of course isolated from the frame ground, the ground return being brought back to the power ground for that circuit.

### 5.2 *Trunk Circuitry*

The interconnections for the trunk circuit relays are also twisted pairs grounded in the same manner as the master scanner pairs. They are also, as far as practical, balanced to ac to suppress relay noise.

### 5.3 *Maintenance and Control Circuitry*

There are a number of relays used for maintenance and control purposes. These are operated by dc connections similar to those of the trunk circuitry.

## VI. OPERATING EXPERIENCE

As previously noted, extensive laboratory tests were made, simulating the worst predictable conditions. Troubles were detected (such as the need for shields on the pick-off transformers) and eliminated.

The central control and associated buses were operated in and close to an operating, electromechanical central office (even with protection taken off adjacent relays) without failure.

The most extensive and significant experience is with the actual offices — with the Holmdel, N. J., systems laboratory, which is a small office with a complete interconnecting bus system, and with the first commercial office, installed and now under test at Succasunna, N. J.

Testing of an assembled office was begun in Holmdel in April, 1963. It has progressed to the state that calls have been set up through it. Testing of the Succasunna office started in August, 1963, and is continuing. Most of this testing uses the interconnecting buses, so this part of the office has been thoroughly tested.

These tests prove that this is a working system which should operate satisfactorily in commercial service.

## VII. SUMMARY

The complex network of transmission lines described here is organized to meet the objective — the rapid exchange of a large amount of information (in digital form) between many functional units. The organization and operational modes result in flexible, dependable and efficient use of hardware. Simple, reliable hardware is used with a high degree of standardization, resulting in over-all economy and reliability.

## REFERENCES

1. Keister, W., Ketchledge, R. W., and Lovell, C. A., Proc. I.E.E., **107**, Part B, Supplement, Nov. 20, 1960.

2. Ault, C. F., Gallaher, L. E., Greenwood, T. S., and Koehler, D. C., No. 1 ESS Program Store, B.S.T.J., this issue, p. 2097.
3. Genke, R. M., Harding, P. A., and Staehler, R. E., No. 1 ESS Call Store — A 0.2-Megabit Ferrite Sheet Memory, B.S.T.J., this issue, p. 2147.
4. Chevalier, J. G., and Eisenhart, R. K., No. 1 ESS Circuit Packs and Connectors, B.S.T.J., this issue, Part 2.
5. Cagle, W. B., Menne, R. S., Skinner, R. S., Staehler, R. E., and Underwood, M. D., No. 1 ESS Logic Circuits and Their Application to the Central Control, B.S.T.J., this issue, p. 2055.
6. Freimanis, L., Guercio, A. M., and May, H. F., No. 1 ESS Scanner, Signal Distributor, and Central Pulse Distributor, B.S.T.J., this issue, Part 2.

# No. 1 ESS Logic Circuits and Their Application to the Design of the Central Control

By W. B. CAGLE, R. S. MENNE, R. S. SKINNER, R. E. STAEHLER and M. D. UNDERWOOD

(Manuscript received January 9, 1964)

*The central control is composed of thousands of relatively simple logic circuits, intricately interconnected to perform the primary data processing functions of the No. 1 electronic switching system. This paper discusses both the circuit development of the basic modules and their application to the logic design of some of the high-speed arithmetic circuits required in the central control. Special electrical design features, wiring rules, and circuit pack assignment procedures are also described.*

## I. INTRODUCTION

Companion articles in this issue<sup>1,2,3</sup> discuss the logical organization and diverse responsibilities of the central control (CC) as the primary information processing unit of the No. 1 electronic switching system.

This paper has the objectives of describing the basic building-block circuits which constitute the framework of the central control and illustrating their logic design applications in some of the major functional portions of the unit.

The presentation is divided into three parts:

Sections II–VI deal with the parameters of the semiconductor devices and the characteristics of the basic circuit in which they are integrated.

The logical effectiveness of the resultant configuration is demonstrated in Sections VII and VIII, which trace its applications from fundamental designs to some of the more complex, specialized functions required in the CC.

Section IX discusses circuit packaging techniques and over-all equipment development as applied to the central control.

## II. LOGIC CIRCUITS

### 2.1 *General*

Numerous factors ranging from system philosophy to device physics were considered in the process of selecting the basic semiconductor logic circuits for No. 1 ESS. The paramount objective in this process was to achieve versatile and reliable circuit performance at an over-all minimal cost. In such a large, complex information processor this can be realized only by a concurrent minimization of costs in the broad areas of design, manufacture and maintenance.

### 2.2 *Building-Block Approach*

The search for the common denominator of minimal cost in these broad areas highlighted the need for a standardized set of building-block logic circuits, not only for use in the central controls, which have the largest concentration of logic circuitry, but also for use throughout the entire system.

Standardization provides the system logic designer with a set of "black boxes" that can be easily interconnected to perform any complex logic function. This facilitates design by reducing the time and cost involved in specifying the logical fabrication of the system.

The minimization of the varieties of building blocks, with their correspondingly higher levels of production, translates to reduced manufacturing costs. Concurrent characterization of devices suitable for many different applications within the system reduces the number of device codes and further benefits the economics of production.

Finally, the advantages to the system user in terms of trouble diagnosis and reduced maintenance costs, particularly in the stockpiling of replacement packages and devices, is considerably improved.

### 2.3 *Reliability*

The electronic circuits in No. 1 ESS must be intrinsically reliable if the over-all system is to provide uninterrupted telephone service. This means that the components used in the logic circuits must be inherently reliable of themselves and used within conservative limits.

Duplication of major system units is the most important safeguard of reliable, continuous service. To prevent the loss of system control, the central control is fully duplicated. Both units of a duplicated pair contain a large number of circuits whose functions are devoted to continually monitoring all internal data processing and cross-comparing

their operations. Even so, the requirement<sup>1</sup> of a 40-year mean time to the simultaneous failure of duplicated units imposes stringent upper bounds on the allowable failure rate of the basic logic circuits.

To help visualize the circuit failure rates implied by these requirements, the required mean repair time in hours has been plotted against the number of failures per  $10^9$  circuit hours in Fig. 1.

To achieve what has been established as a reasonable mean repair time for a CC of from one to three hours (including both fault location and replacement times) it is apparent that there must be in the order of 50 to 100 failures or less per  $10^9$  circuit hours. For individual logic circuits, the semiconductor failure rates must then fall in the range of 1 to 10 failures per  $10^9$  device hours to meet these objectives.

#### 2.4 Switching Speed

The No. 1 ESS system organization requires a number of units to function in real time and share a single control unit. This necessitates the use of fast switching circuits in the control unit in order not to limit the system capability. In switching circuit design, trade-offs are often made between long logic chains which require a small number of logic elements and larger, more complex circuits which have less delay.

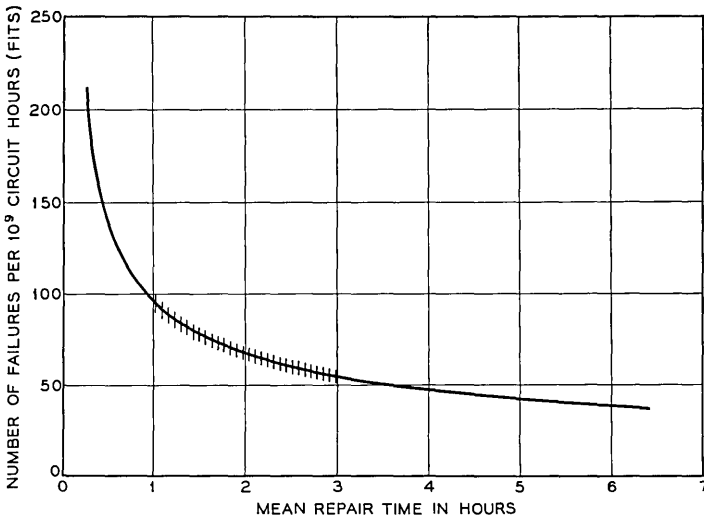


Fig. 1 — Logic circuit failure rate vs mean repair time. Assumptions: duplicated control systems, each containing 12,500 circuits; 40-year mean time to simultaneous failure of duplicated units. Typical circuit elements: 1 transistor, 4 diodes, 4 resistors, 16 soldered connections, and 4 pressure contacts.

Faster switching speeds in the individual devices alleviate the problem considerably, but this is countered by the increasing cost of the devices. These factors are all weighed in deciding upon a satisfactory operating speed.

### 2.5 *Environment*

For over-all system economy, the logic circuits are designed to utilize the common central-office battery power sources. Since these voltages can decrease by as much as 20 per cent during commercial ac power failure, the logic circuits must be relatively insensitive to supply voltage variations.

The logic circuits are designed to operate reliably over the temperature range of 0–55°C.

### 2.6 *Silicon Semiconductors*

The low leakage currents and high allowable junction temperatures of silicon devices as compared to germanium have resulted in the specification of silicon semiconductors for all circuits in No. 1 ESS. This eliminates the necessity for an air-conditioned environment. The greater tolerance to temperature variations results in smaller devices for the same power rating, allows higher packing densities for the equipment with a corresponding reduction in lead lengths and floor space, and allows the use of wave soldering on printed wiring boards. Low storage times permit high-speed operation with relatively simple circuit designs. The slightly higher voltage drop characteristic of silicon devices is easily circumvented.

## III. SELECTION OF A BASIC GATE

### 3.1 *Type of Function*

There are many different types of semiconductor logic gates which fulfil in varying degrees the basic requirements outlined in the preceding sections. The selection of the optimum gate for a particular application depends on the relative weight given to factors such as switching speed, component and assembly costs, reliability, allowable tolerances of components and voltages, minimization of device and package codes, logical flexibility, noise margins, and many others.

The building-block logic gates for a system must, first of all, be capable of being interconnected to perform any complex logic function which may be required by the logic designers. This requires that a basic set



of functions be provided. Historically, AND and OR gates together with both inverting and noninverting amplifiers and memory cells (flip-flops) were first used. This was the case in the experimental Morris, Illinois, electronic switching system.<sup>4</sup>

Recently, more complex functions have been combined in the basic block — for example, the various forms of AND-NOT and OR-NOT. These are more efficient in that any one can be used exclusively to construct a complete logic system. A further benefit is that these circuits usually insert voltage or current gain in every logic stage, reducing the variations in impedance levels in a complex circuit and thereby improving speed and crosstalk performance.

### 3.2 Low-Level Logic

After studying the characteristics of many of the widely used gate circuits, the diode-coupled AND-NOT gate<sup>5</sup> was chosen for use in No. 1 ESS. The circuit is generally referred to as low-level logic (LLL). As shown in Fig. 2, the LLL gate consists of a diode AND gate, a biasing circuit to overcome the combined voltage drop of an input diode and the driving transistor's saturated  $V_{CE}$ , and an inverting amplifier. The combination of AND plus INVERSION derives the logical AND-NOT function. This circuit has an individual source of base current to avoid the unequal base current problems present in other logic configurations. The diodes are poled in such a manner that large reverse base drive currents can be obtained. These properties result in full utilization of the switching speed capabilities of the transistor. The voltage margins are designable and do not depend solely on device characteristics. All in all, LLL is a high-performance gate yielding a

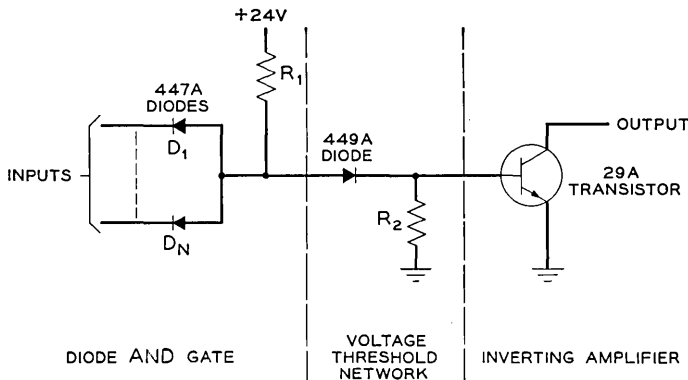


Fig. 2 — LLL circuit.

combination of fast switching speeds, large fan-in and fan-out, and excellent margins.

#### IV. LLL CIRCUIT DESIGN

##### 4.1 *Worst-Case Design*

The worst-case design philosophy was used for the LLL circuit. This means that the expected characteristics of the gates were predicted on the basis of all circuit parameters being simultaneously at their extreme limits in such a manner as to degrade performance. This approach enhances circuit reliability by insuring that the margins of typical circuits are well within the design limits. For the LLL gate, worst-case design still yields very fast operation with moderately high fan-in and fan-out for relatively wide resistor and supply voltage tolerances.

##### 4.2 *Saturated Operation*

An important consideration in the design of the LLL circuit is the use of a saturating or nonsaturating amplifier design. The primary advantage of the nonsaturating design is the elimination of the delay caused by the storage time of minority carriers in the transistor during turn-off. The significance of storage time is very much a function of the particular transistor employed and the need for speed in the system. Nonsaturating designs are complex, costly circuits which result in larger output voltages when the transistor is conducting. Also, biasing of the transistor in its active region not only increases power dissipation in the device, but also causes the circuit to be more sensitive to noise and prone to oscillation.

Part of the apparent speed advantage of the nonsaturating circuit is lost because of the necessarily higher threshold voltages. For many high-speed silicon switching transistors the storage times are so short relative to their rise and fall times as to be almost negligible in an LLL-type circuit. Furthermore, since approximately half of the stages in a logic chain are turning ON and half are turning OFF at any one time, only half the improvement in turn-off delay can be deducted from the average delay per stage. These considerations led to the selection of a saturating amplifier design.

##### 4.3 *Biasing Network*

The major item for design in the LLL gate is the biasing network. Once a method for performing this function is selected and the element

is properly designed, the characteristics of the gate are largely determined by the inherent capabilities of the semiconductor devices, ambient operating conditions, and the tolerances of the resistors and power supply voltages.

The primary function of the biasing network in an LLL gate is to insure that its transistor will not conduct when any input to that LLL is held low by a conducting transistor in the prior stage. This requires that the biasing circuit keep the base-emitter voltage below the minimum threshold of the transistor for the maximum sum of the diode voltage drop and the  $V_{CE}$  of the saturated transistor.

The biasing network has three other important functions. First, the design should cause relatively large reverse base current flow when the transistor is switched from ON to OFF. This is necessary to obtain minimum turn-off time. Second, the biasing network should provide a noise threshold to the amplifier to prevent small transient voltage perturbations from falsely turning on the transistor. Third, the network should be designed so that the voltage across it remains as low as possible when current flows from  $R_1$  into the base of the transistor (see Fig. 2). This minimizes the voltage swing required at the gate input.

The voltage biasing element in the LLL circuit consists of three silicon junctions in series. The three junctions provide approximately a two-volt threshold to the amplifier without the need for a second power supply. Resistor  $R_2$  across the base-emitter junction of the transistor provides a relatively low-impedance shunt for the small forward current in the multijunction diode when the input to the circuit is low. This improves the noise margin by insuring a low potential at the base of an OFF transistor. The signal voltage swing is minimized since the difference between the threshold voltage and the forward-biased voltage drop of the multijunction diode is small. The reverse breakdown voltage of the three junctions is very high to prevent any reverse dc current. The necessary reverse transient drive is obtained by designing the multijunction diode to have a relatively long recovery time. Thus if an input to an LLL circuit whose transistor has been conducting suddenly goes low, the voltage across the diode cannot change instantaneously. For a brief time the base of the transistor is made negative and reverse base current is drawn through the temporary low reverse impedance of the multijunction diode. After recovery the diode reverts to a high impedance and prevents noise from being transmitted into the base node. To further insure this, the net capacitance of the three junctions in series is kept small.

#### 4.4 *Dummy Load Resistor*

The LLL circuit shown in Fig. 2 has one serious deficiency. In the OFF state the collector node is at an indeterminate potential between two high impedances which are presented by the OFF collector and the nonconducting input diodes in the driven circuits. Not only does this make the node sensitive to noise, but it also slows down the turn-on of the following stages, since collector node stray capacitance must be charged by the internal gate resistors of the driven LLL circuits. The addition of a dummy load resistor from the collector node to a voltage above the threshold of the driven inputs provides a charging current for the capacitance, defines the OFF state voltage and provides a noise margin in this state, and limits the depth of saturation by establishing a lower bound for collector current.

#### 4.5 *Operating Levels*

The choice of current and voltage levels for the LLL circuit is influenced by many of the parameters discussed earlier and ideally should be made for each system and device selection. In general, the selection of low impedance levels improves speed and capacitive crosstalk performance while higher levels result in lower power dissipation (allowing higher packing densities), improved device reliability, and fewer inductive interaction problems. In high-speed circuits the mechanical embodiment of individual circuits and of whole equipment units is significant because of its primary effect upon stray capacitance and inductance.

The impedance level chosen for the logic circuits in No. 1 ESS was largely determined by the stray capacitance anticipated at collector nodes and by estimates of wiring inductance. To charge an estimated maximum node capacity of 150 pf to the threshold of an LLL circuit in 50 nsec implies a charging current of about 10 ma. In the worst case (fan-out of one), approximately 75 per cent of this current is assigned to the dummy load and 25 per cent to the internal gate of the driven circuit. This guarantees the 10 ma for the minimum fan-out case and keeps the ratio of maximum to minimum collector current low. The choice of a collector supply voltage in the 4- to 5-volt range provides sufficient margin for noise and establishes the OFF collector impedance at about 500 ohms. The current and voltage levels chosen assure fast switching speeds for the low-power semiconductor devices described in Section V. In addition to these considerations, the effects of stray capacitance and inductance on interaction between circuits (crosstalk) had to be evaluated for the packaging and wiring arrangements used in No. 1 ESS.

#### 4.6 *L and C Crosstalk Effects*

In addition to degrading rise times, stray capacitance between signal leads can cause interference between circuits. At the impedance level chosen, these effects do not significantly alter performance in this system.

Wiring inductance can affect circuit performance in two ways. Series lead inductance causes longitudinal voltage drops due to the rapid change of signal current. The allowable length of wire depends upon its inductance, the noise thresholds of the circuits, the operating current levels, and the required switching speeds. The wire inductance can be reduced by using twisted-pair wire for long interconnections. In addition to the series inductance problems, the mutual inductance occurring between parallel wire runs can cause interference between circuits.

The interactions of the  $L$  and  $C$  effects, together with the complex wiring patterns and nonlinear waveforms occurring in logic circuits, make an exact analysis of the net effects very difficult.<sup>6</sup> Analytic studies of simplified models in conjunction with laboratory measurements were utilized to formulate the following set of wiring rules to insure that the physical wiring patterns would limit the crosstalk to acceptable levels.

#### 4.7 *Wiring Rules for Interconnecting Logic Circuits*

- (1) Leads must be as short as possible, using horizontal and vertical paths
- (2) wires following the same path should be loosely constrained, i.e., not laced into a cable
- (3) a single wire should be used to tie the points of a node together and to a dummy load resistor where:
  - (a) the total node wiring is less than 6 feet and
  - (b) parallel runs do not exceed 3 feet
- (4) twisted pairs must connect points of a node together and to three dummy loads in parallel at the driven end where:
  - (a) the total node wiring exceeds 6 feet and
  - (b) parallel runs exceed 3 feet but in no case exceed 10 feet
- (5) when two LLL circuits are interconnected to form a flip-flop, they must be on the same circuit pack.

### V. DEVICE CHARACTERISTICS

#### 5.1 *General*

The characteristics of each of the semiconductors used in the LLL circuit are shown in Tables I–III. Both the input diode and the multi-

junction diode are specifically characterized for the LLL circuit. The transistor, on the other hand, is characterized for use in many different applications in a variety of circuits in No. 1 ESS.

The reliability of these devices when used in the LLL gate has been predicted on the basis of step-stress aging experiments.<sup>7</sup> The studies indicate that failure rates will be less than 10 in  $10^9$  device-hours for the transistor and less than 5 in  $10^9$  device-hours for the diodes. These rates are adequate to meet the reliability objectives discussed in Section 2.3.

The variations of specific parameters over the temperature range of 0–55°C were considered in establishing specific limits, although all limits are specified at 25°C to facilitate testing.

### 5.2 29A Transistor

The most important parameters of the transistor employed in the LLL circuit are presented in Table I.

The choice of operating level in the LLL circuit restricts the maximum collector current, collector voltage, and power dissipation to well below the maximums allowable. In general, these margins have been utilized to improve reliability in the logic circuits. In a number of cases the extra capability has been utilized in the design of scaled-up versions of the LLL circuit which satisfy the need to drive high fan-out situations within the logic and to drive cables to remote units.

Although the specified limits are important in determining worst-case performance, a knowledge of the distribution of parameters is necessary to predict typical behavior. Distributions of some of the more significant parameters are given in Figs. 3, 4, and 5. From the figures it is obvious that all of the limits specified are well within the device capability, which assures a high yield in production with resultant economies.

TABLE I — 29A TRANSISTOR SPECIFICATIONS

$BV_{CEO} \geq$	30 v	$I_C = 5$ ma, $I_B = 0$
$BV_{EBO} \geq$	7 v	$I_C = 0$ , $I_E = 10$ $\mu$ a
$h_{FE} \geq$	30	$I_C = 5$ ma, $V_{CE} = 1$ v
$V_{CE} \leq$	0.5 v	$I_C = 50$ ma, $I_B = 2$ ma
$V_{BE} \leq$	1.1 v	$I_C = 50$ ma, $I_B = 2$ ma
$C_{ob} \leq$	6 pf	$I_E = 0$ , $V_{CB} = 5$ v
$C_{ib} \leq$	10 pf	$I_C = 0$ , $V_{EB} = 1.5$ v
$T_{on} \leq$	65 nsec	$I_C = 35$ ma, $I_B = 1.5$ ma
$T_{off} \leq$	65 nsec	$I_C = 10$ ma, $I_B = 2.5$ ma, $V_{BER} = -0.9$ v
$\theta_{J-C} \leq$	0.33°C/mw	

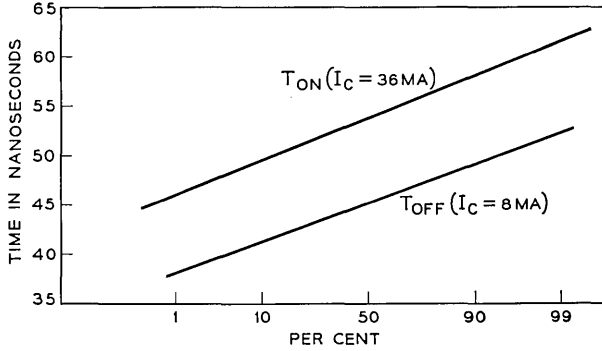


Fig. 3 — Typical switching speed distributions for worst-case conditions for 29A transistor.

5.3 449A Multijunction Diode

Table II lists the parameters of the multijunction diode used in the biasing network of the LLL circuit. The minimum forward voltage specification guarantees an adequate noise margin under the worst (high-temperature) conditions for the OFF state. The maximum limit assures a low drop in the conducting state, which provides an efficient transfer of current into the base of the transistor when it is turned ON.

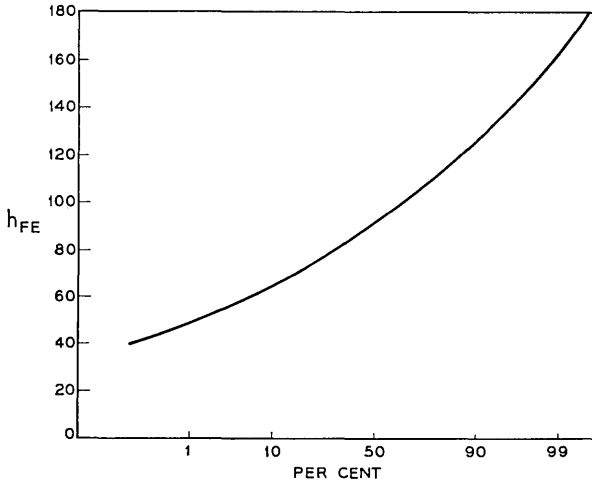


Fig. 4 — Typical  $h_{FE}$  distribution for 29A transistor:  $I_C = 5 \text{ ma}$ ,  $V_{CE} = 1 \text{ v}$ , circuit design limit  $h_{FE} \geq 22$ .

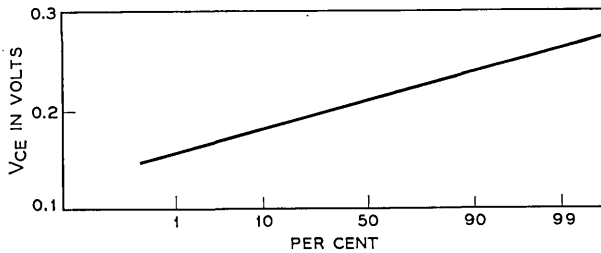


Fig. 5 — Typical  $V_{CE}$  distribution for 29A transistor:  $I_C = 50$  ma,  $I_B = 2$  ma specification limit  $\leq 0.5$  v.

Specifying the minimum stored charge guarantees that sufficient base current can be drawn during turn-off of the slowest 29A transistor. The upper limit on the total capacity across the three junctions limits noise transmission to a satisfactory level.

#### 5.4 447A Logic Diode

The parameters of the 447A diode are given in Table III. The forward voltage drop, typical for a small silicon computer diode, is easily accommodated by the threshold created by the multijunction diode and the transistor base-emitter junction. The reverse breakdown is higher than necessary for the circuit, but improves reliability and allows for other applications. The reverse leakage current is so small as to be negligible for any reasonable fan-in.

The upper limits on capacitance and reverse recovery time are necessary to assure obtaining the full speed capability of the transistor. Consider the circuit in Fig. 6. If both  $Q_0$  and  $Q_1$  are conducting, current through  $R_1$  will be shunted through  $Q_0$  to ground and the current in  $R_2$  passes through  $Q_0$  and  $Q_1$  in parallel; therefore  $Q_2$  and  $Q_3$  are non-conducting. If  $Q_0$  is suddenly turned OFF, the shunt path to ground for  $I_1$  will be removed, allowing  $Q_2$  to turn ON. On a transient basis, however, diode  $D_2$ , which serves to isolate the collector nodes, can have a

TABLE II — 449A MULTIJUNCTION DIODE SPECIFICATIONS

$V_F \geq$	1.53 v	$I_F =$	70 $\mu$ a
$V_F \leq$	2.3 v	$I_F =$	2.5 ma
$V_F \leq$	2.85 v	$I_F =$	20 ma
$C \leq$	15 pf	$V_F =$	1 v
Stored charge $\geq$	$0.4 \times 10^{-9}$ coul	$I_F =$	2 ma, $I_R =$ 10 ma



TABLE III — 447A LOGIC DIODE SPECIFICATIONS

$BV \geq$	40 v	$I_R =$	$5 \mu a$
$I_s \leq$	25 na	$V_R =$	20 v
$V_f \leq$	1 v	$I_f =$	10 ma
$C \leq$	3.5 pf	$V_R = 0, f_0 =$	100 kc
$t_{rr} \leq$	4.0 nsec		

low reverse impedance for a time equal to its recovery time. As long as the low reverse impedance condition persists, the current from  $R_1$  can "leak" through  $D_2$  as indicated by  $I_3$  in the figure. This interaction between nodes causes no logical error but does introduce an absolute delay in  $Q_2$ 's turning on. In a fast system such delay is not tolerable. The capacitance of the diode allows a similar interaction. In addition, the capacitance can transmit noise into the internal AND gate node and also degrade rise times at this node by its shunt effect on the current from  $R_1$ . The specifications on the 447A diode are such as to make these effects almost negligible.

VI. SUMMARY OF LLL CHARACTERISTICS

The outstanding characteristics of the LLL circuit designed for No. 1 ESS are summarized in Table IV.

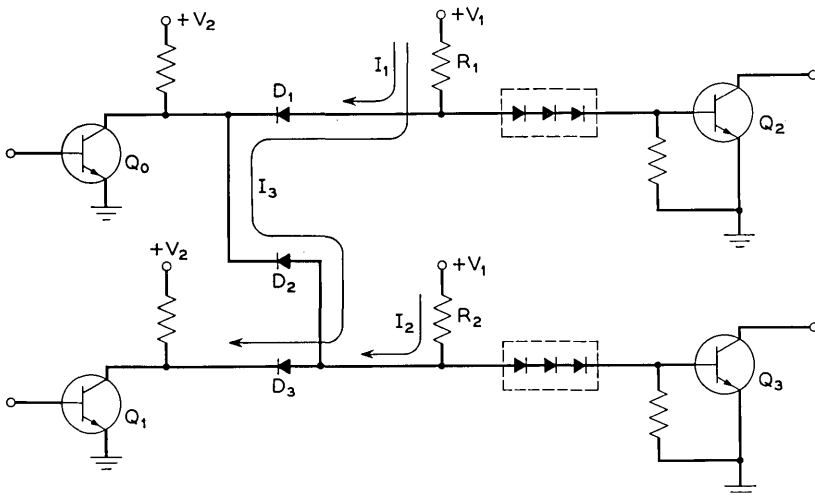


Fig. 6 — Diode recovery effect.

TABLE IV — LLL CHARACTERISTICS

Parameter	Range	Typical Value
Signal levels		
high (1)	4.0-5.1 v	4.6 v
low (0)	0.1-0.5 v	0.25 v
Fan-out	1-8	4
Fan-in	1-20	4
Turn-on delay	10-65 nsec	35 nsec
Turn-off delay	10-65 nsec	35 nsec
Noise margins		
high	1.6-3.6 v	2.5 v
low	1.0-1.5 v	1.2 v
Circuit margins		
resistor tolerance	$\pm 20\%$	
voltage tolerance	$\pm 10\%$	
Power consumption	37-140 mw	76 mw
Operating temp. range	0-55°C	

## VII. LOGIC DESIGN

## 7.1 Applications of LLL in Logic Designs

It was mentioned earlier that LLL gates can be used exclusively to synthesize any logical combination possible with AND, OR, and NEGATION. The gate performs the AND-NOT function for the convention that a "high" or  $+v$  signal is a logical 1 and a "low" or ground signal is a logical 0.

The standard logic symbol for the LLL circuit is shown in Fig. 7 along with a function table defining its properties.

There are many ways in which complex logic circuits can be synthesized using LLL. One of the simplest approaches is to perform the original logic design in terms of the more familiar AND, OR, and NOT functions and then to substitute LLL equivalents for each of the original functions.



(a)

A	B	C (OUTPUT)
0	0	1
0	1	1
1	0	1
1	1	0

(b)

Fig. 7 — Building block logic.

The basic equivalent circuits and several other important configurations are shown in Fig. 8.

In using the substitution procedure, the following two rules should be repeatedly applied until further simplification is impossible:

(1) cancel (remove) tandem pairs of single input LLL gates since two inverters in series perform no net logical function;

FUNCTION	AND, OR, NOT CIRCUIT	LLL CIRCUIT
INVERSION: $f_2 = f_1'$		
AND: $f = AB$		
OR: $f = A + B$		
AND-OR: $f = AB + CD$		
AND-NOT: $f = (AB)'$		
OR-NOT: $f = (A + B)'$		

Fig. 8 — Equivalent logic circuits.

(2) when two or more LLL gates drive a common gate and this gate is their only load, the outputs of all of the driving gates can be tied together and applied to a single input of the driven gate.

The example in Fig. 9 illustrates this process of simplification.

Several of the examples highlight the common connection which is

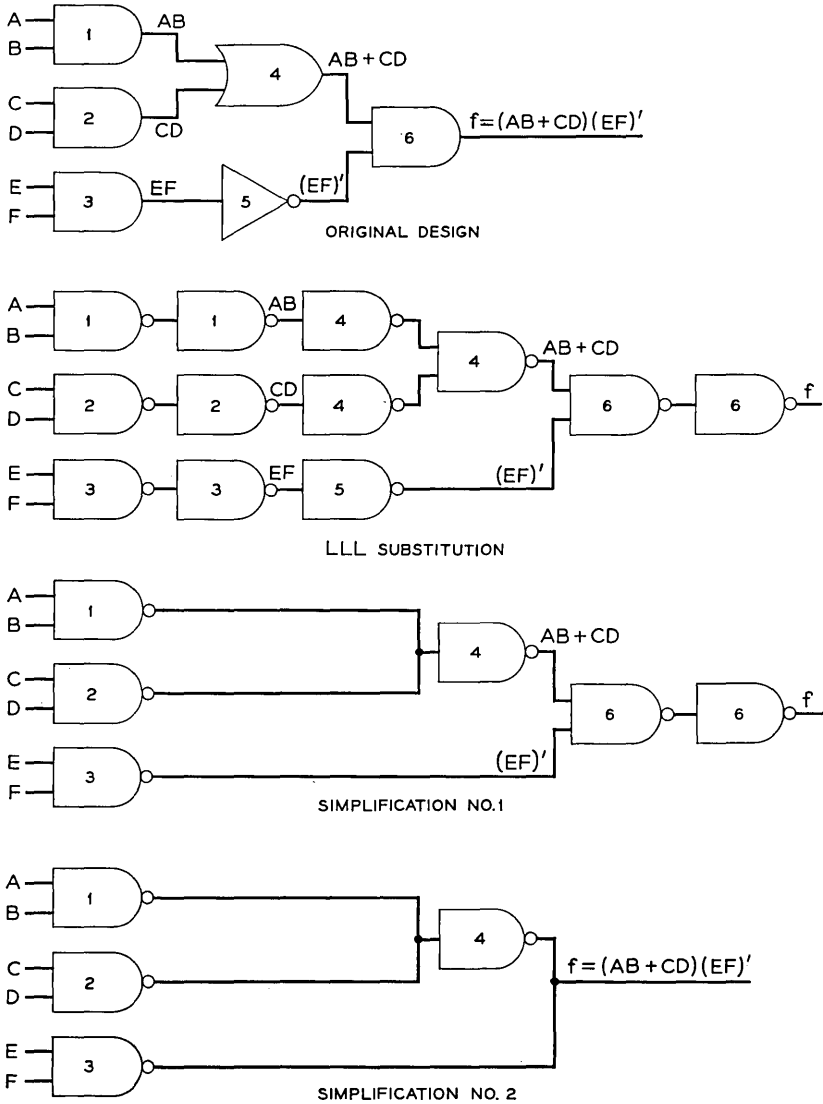


Fig. 9 — LLL simplification process.

made between the outputs of transistors which are driving a common input. An AND function is effectively formed here, since all transistors connected to the junction must be nonconducting in order that the output be high (logical 1). The use of this technique has resulted in the savings of several thousand logic diodes in the central control alone. The number of transistors which can be connected in this way is limited by the adverse effect on switching speed which is caused by the increases in inductance and capacitance.

Another approach in using LLL in logical designs is the "inhibit" technique. A simple example illustrates this approach in the design of a match circuit. In such a design, the output should be high when the two input variables ( $A$  and  $B$ ) have the same state. Fig. 10(a) illustrates this solution. Conversely, one can think of the match function as requiring a high output except when  $A$  and  $B$  mismatch, in which case the output should be inhibited. This leads to a solution utilizing one LLL for each input combination which is to inhibit the output [see Fig. 10(b)]

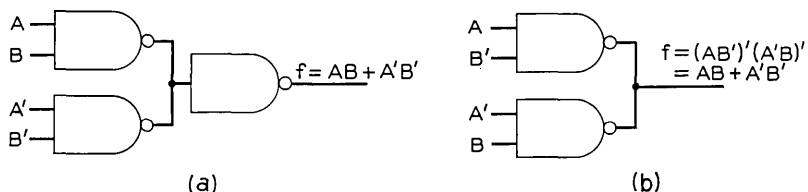


Fig. 10 — LLL match circuits.

### 7.2 The LLL Flip-Flop

Throughout the switching system, and particularly within the CC, there is the need for temporary memory facilities in the form of flip-flops. These are readily constructed by cross-connecting two LLL gates as shown in Fig. 11. The circuit operates as follows: gating functions insure that inputs  $A$  and  $B$  are both held in the high state when the

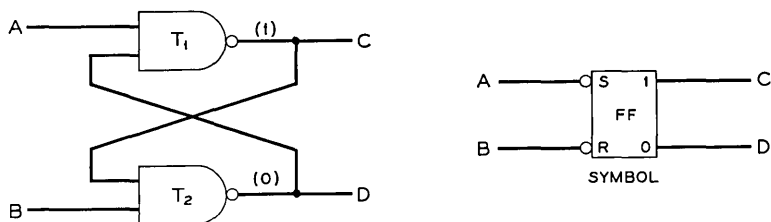


Fig. 11 — LLL flip-flop.

flip-flop is quiescent. Assume that the “set” output is high ( $C = 1$ ) and accordingly  $D = 0$ . Any change in the state of  $A$  is ineffective, since the cross-connected input from  $D$  keeps  $T_1$  nonconducting. However, if  $B$  goes to 0, the base drive for  $T_2$  is removed, and  $D$  will go high, providing drive for  $T_1$  and causing  $C$  to change from 0 to 1.

7.3 Binary Counters, Shift Registers and Cable Pulsers

In addition to the basic LLL gate and flip-flop, special circuits for binary counters and shift registers are also required. These consist of steering gates applied onto the LLL flip-flop to provide nonracing counting and shifting functions. “Nonracing” means that correct operation is independent of input pulse width as long as the width is greater than the minimum allowed, which here is approximately 0.25 microsecond.

The binary counter circuit is shown in Fig. 12. As can be seen, memory is provided by an LLL flip-flop, which has been modified to include direct connections to the transistor bases. Two steering networks and a

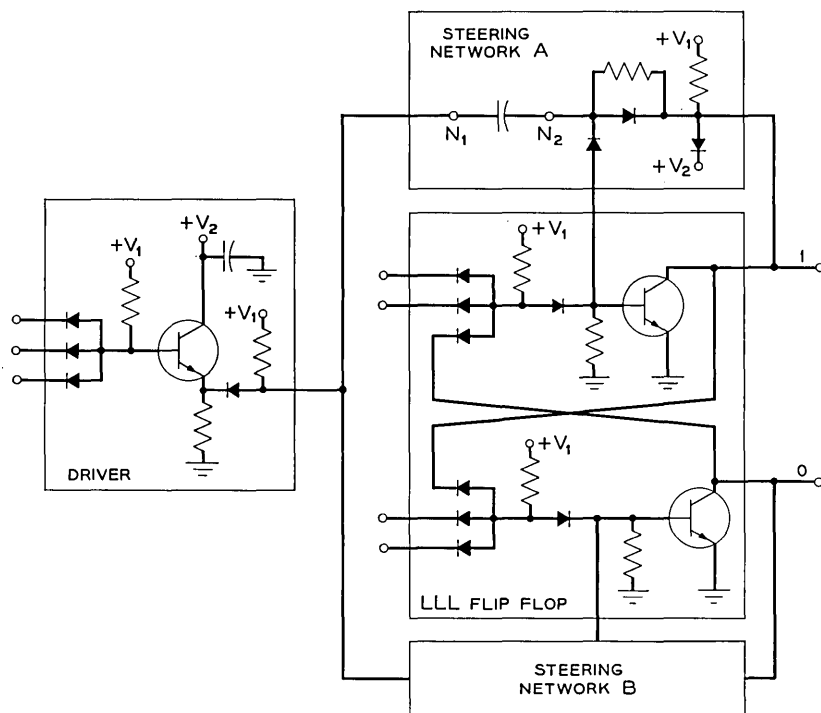


Fig. 12 — Binary counter.

special driving circuit have been added to obtain the nonracing binary counting action.

The design is such that if all inputs to the driver become high coincidentally for longer than 0.25 microsecond, the flip-flop will reverse its state when any input returns to the low condition and remains there for at least 0.2 microsecond. In the steady-state condition, the  $N_2$  points in the networks are at approximately the same potential as the output voltage of that side of the flip-flop to which they are connected (either +4.5 or approximately zero volt), while the other sides of the capacitors (points  $N_1$ ) are near ground potential. When all driver inputs go high, the emitter follower drives the  $N_1$  points to near +4.5v and the  $N_2$  points remain at their original voltage levels. When the output of the driver goes low due to an input going low, the  $N_2$  points attempt to drop by 4.5 volts. That  $N_2$  which was originally at +4.5v will drop to near ground potential, while the other  $N_2$  will go negative and forward bias the diode connected to the base of the transistor on its side of the flip-flop, turning it OFF. Since this transistor was originally ON, the flip-flop will change state. The  $N_2$  points in both networks will return to the new output voltage states as the capacitors in the networks charge to their new voltage values. The next pulse from the driver will repeat the process, except that the opposite transistor will now be the one which turns OFF.

The shift register cell is almost identical to the binary counter, except that the two steering networks are driven by individual drivers which have one common shift control input. The two drivers have individual data inputs which are connected to the complementary outputs of the preceding stage in a multistage shift register. In this way only one driver output will be activated when the shift control input goes high and the cell will be forced into the original state of the preceding stage when the shift signal goes low.

The LLL gate has one other important application in No. 1 ESS. It is used (at a higher current level) as the basic cable pulser for the interframe bus communication system (see Fig. 13). The bus receiver circuits which are designed to drive standard LLL gates are also shown. This application is described in detail in a companion paper.<sup>2</sup>

## VIII. LOGIC DESIGN IN THE CENTRAL CONTROL

### 8.1 *General*

By far the largest use of logic circuits in No. 1 ESS is made in the central control (see Fig. 14). Some feeling for the size and complexity

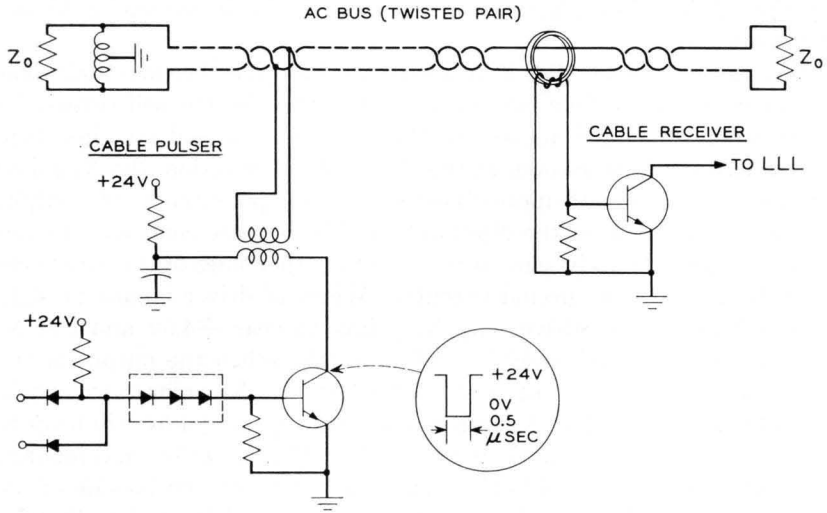


Fig. 13 — LLL gate used as a cable pulser.

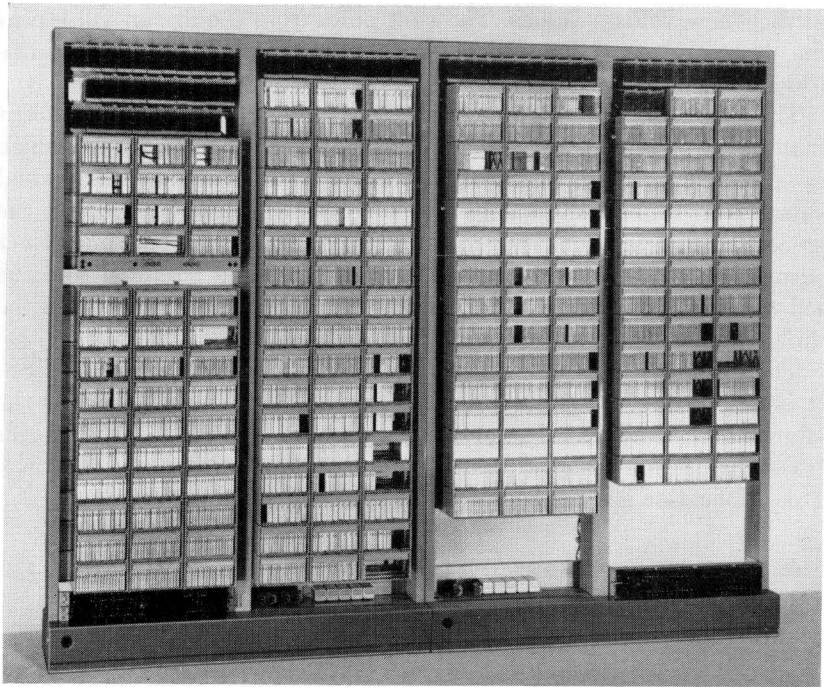


Fig. 14 — Front view of central control.



of the unit is found in the fact that the circuitry for a pair of duplicated CC's, while contained within about 5 per cent of the volume of a typical 10,000-line telephone office, contains about 50 per cent of the semiconductor devices. The statistics of Table V show the CC in perspective with the rest of the system.

Because of its size, it is impractical to discuss all of the details of the basic data processing circuits of the CC within the scope of this paper. The majority of circuits are used in translating program instructions, providing internal flip-flop registers between which data routinely shuttle, communicating with peripheral equipment, and performing routine maintenance checks. These circuits contain a high degree of design repetition and their implementation is fairly straightforward. Most of the real processing of data takes place in a small portion of the machine known as the "arithmetic system." The functions of this unit will be developed in detail, as an illustration of the over-all logic design and equipment layout techniques that have been employed.

### 8.2 Arithmetic System

In the process of completing telephone calls and diagnosing system troubles, the central control is called upon to manipulate both arithmetic quantities and logical quantities which are characterized in binary form. Arithmetic numbers contain 23 bits, labelled 0 to 22 in ascending order from the least significant at the right to the most significant at the left. Positive and negative numbers are permissible and are distinguished by the state of bit 22, the sign bit. Since the central control has a capacity of 23 bits, including the sign, the largest number therefore cannot exceed  $2^{22} - 1$  ( $= \pm 4,194,303$ ). For positive numbers, the sign bit is 0. Negative numbers are the "ones" complement of the equivalent positive value and contain a 1 in the sign bit. For example, +1 is 000 ... 001, and -1 is 111 ... 110. Both +0 (000 ... 000) and -0 (111 ... 111) can exist in this number system.

One of the chief constituents of the arithmetic system is a parallel

TABLE V — RELATIVE COMPLEXITY OF CENTRAL CONTROL

Item	No. of Items in Two CC's	% of Total	No. of Items in 10,000-Line No. 1 ESS
Circuit packs	4,716	37.0	12,600
Transistors	27,142	49.0	55,000
Diodes	89,934	54.0	160,000
Wire-wrapped connections	173,200	13.0	1,300,000
Equipment bays	8	5.0	147

binary adder circuit which generates the algebraic sum or difference of two or more 22-bit signed numbers in an interval of less than 1 micro-second. Addition is performed conventionally by summing corresponding bits of a number and combining them with carries that might have been generated in less significant bit positions. Subtraction is performed by the use of addition of complements with end-around carry. That is, the subtrahend is first complemented and then added to the minuend. A carry from the most significant bit sum is returned to be re-added to the lowest-order bits.

Another function of the arithmetic system is the ability to perform the purely logical functions of AND, OR, and EXCLUSIVE-OR upon two 23-bit quantities.

The arithmetic unit also has provision for shifting or rotating the positions of the bits of a binary word within a register. A shift register circuit is available which can move a word in either the right or left direction by any integral displacement of from 0 to 23 positions. A special mode rotates only 16 bits of the word, while leaving the others undisturbed. A complete shift cycle requires approximately  $3.5 \mu\text{sec}$ .

### 8.2.1 Block Diagram

A simplified block diagram of the arithmetic system is shown in Fig. 15. The elements divide into two categories, namely, those concerned with the additive-type functions and those relating to rotational functions.

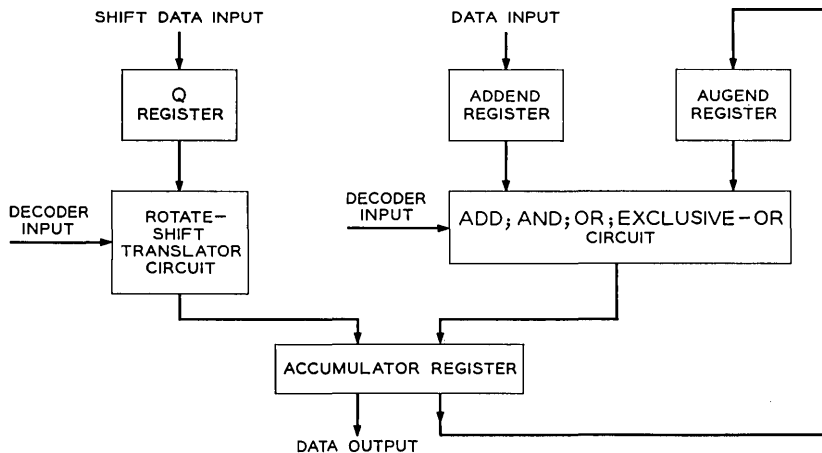


Fig. 15 — Block diagram of arithmetic system.

The circuit requirements for the additive functions are quite simple. Two flip-flop registers, the addend and augend, act as transient memories into which are inserted the incoming operands. The block labelled ADD, AND, OR, EXCLUSIVE-OR contains the logic to produce all of these functions. Finally, a register is required to hold the resultant. This is the accumulator.

The accumulator is constructed of LLL shift register circuit packs (Section 7.3) to provide greater flexibility than mere flip-flop memory. Steering networks associated with each side of the internal flip-flop allow tandem, race-free connections to other bits in the register. Upon command, the value of any bit in the register can be transferred to any other bit position to which it has been wired. The logic to interpret the shift-rotate commands and to translate the amount of displacement is contained in the rotate-shift translator. The Q register is a 6-bit flip-flop group which holds the shift-rotate translator data while the operation is taking place.

### 8.2.2 *Accumulator Input Registers*

The addend and augend registers consist of 23 LLL flip-flops with associated input gating. All new data are introduced into the arithmetic system by way of the addend register, which connects to the central control's main transmission artery — the masked bus. The augend register has two input options. It may be "reset" to zero by a signal from the order decoders, or it may be set up to the contents of the accumulator register. The latter path provides the ability to accumulate a sum by the process of repeatedly adding new numbers to the earlier totals. Since both input registers are connected to the adder circuit without gating, the adder continually monitors the sum of the register contents.

### 8.2.3 *Adder Circuit*

The inherent slowness of binary adders arises from the fact that "carries" which are generated in low-order bits create an appreciable delay by propagating serially toward the more significant bit positions. A complete sum cannot be available until all carries have been terminated. An adder circuit was developed which partially overcomes this problem and achieves the required speed without an unreasonable increase in circuit components. The circuit makes use of the group-carry concept, whereby all the carries within a prescribed group of bits are generated simultaneously and propagation is restricted to occur only between the adjacent groups.

When adding the corresponding bits of two binary numbers  $X$  and  $Y$ , a carry is obtained whenever both bits  $x_i$  and  $y_i$  are 1; or when either  $x_i$  or  $y_i$  is 1 in conjunction with a "carry in" from the adjacent lower-order bit. The relation can be expressed as:

$$C \text{ out} = X_i Y_i + (X_i + Y_i)C_{i-1}.$$

In order to take advantage of the inversion property of the LLL circuit, it is convenient to complement the expression to obtain

$$C' \text{ out} = X_i' Y_i' + (X_i' + Y_i')C_{i-1}'.$$

Now, the "carry out" from the  $i$ th position will be the "carry in" to  $i + 1$ , so that  $C_{i+1}'$  can be expressed as

$$C_{i+1}' = X_{i+1}' Y_{i+1}' + (X_{i+1}' + Y_{i+1}')C_o'.$$

Substituting the value of  $C_o$  and simplifying yield the product-of-sums expression:

$$C_{i+1} = (X_{i+1} + Y_{i+1})(X_i + Y_i + X_{i+1}Y_{i+1})(C_{i-1} + X_i Y_i + X_{i+1}Y_{i+1}).$$

Let

$$A_i = X_i + Y_i$$

$$B_i = X_i Y_i.$$

Then

$$C_{i+1} = A_{i+1} (A_i + B_{i+1})(C_{i-1} + B_i + B_{i+1}).$$

If the process is continued for successive carries, a general expression for the  $n$ th carry is obtained in terms of the input:

$$C_n = A_n (A_{n-1} + B_n)(A_{n-2} + B_{n-1} + B_n)(\dots) \dots \\ \dots (A_2 + B_3 + \dots + B_{n-1} + B_n)(C_{in} A_1 + B_1 + B_2 + \dots + B_n).$$

The equation consists of  $n$  product-of-sum terms whose complexity increases in a pyramidal pattern. For example, the term  $(A_1 + B_2)$  would appear as the second factor in generating the carry  $C_2$ . It reappears, in union with  $B_3$ , as the third factor of  $C_3$ . This new factor,  $(A_1 + B_2 + B_3)$ , combines with  $B_4$  as the fourth product of  $C_4$ , and so on. Each product term in every carry expression follows a similar growth pattern. These combinations are suited to LLL circuitry in that the highly repetitive B terms need only be generated once and can be combined successively (within fan-out limitations) to form a series of carry outputs.

Although any number of carries can be generated simultaneously in a two-stage logic circuit by implementing the above equations, the number of LLL gates required quickly becomes excessive when more than about 15 carries are generated. The curve of Fig. 16 summarizes the circuit requirements for multibit adders of any size up to 25 bits. It is useful in predicting the circuit complexity for tandem arrangements of carry groups to obtain a desired addition speed. The curve is divided into two areas. The portion with the larger delay occurs for group sizes greater than 11 carries and is occasioned by the fact that an extra logic stage is necessary to satisfy the fan-out limitations of the LLL circuit. The minimum delay of four logic stages includes the two stages for addition in the sum circuit, and applies to the first group of a tandem arrangement only, since the carries in adjacent groups are completed concurrently with the preceding sum. Therefore, each subsequent carry group after the first contributes only two delays to the total addition time. On this basis, an adder with two group carries requires six delays, one with three groups requires eight delays, etc.

Fig. 17 compares the circuit component requirements with the delay for several possible 23-bit end-around carry adders. Point A on the curve indicates that approximately 800 LLL gates are needed to design a circuit which generates all bits simultaneously and produces a sum in 5 stages of delay,  $\approx 300$  nsec. At the other extreme, point D describes the conditions of a quasiparallel adder where the carry propagates on a bit-by-bit basis. B shows the condition for which the carries are divided

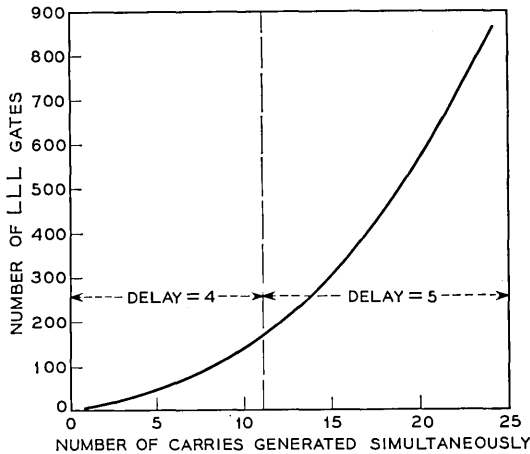


Fig. 16 — LLL circuit requirements for parallel carry adders.

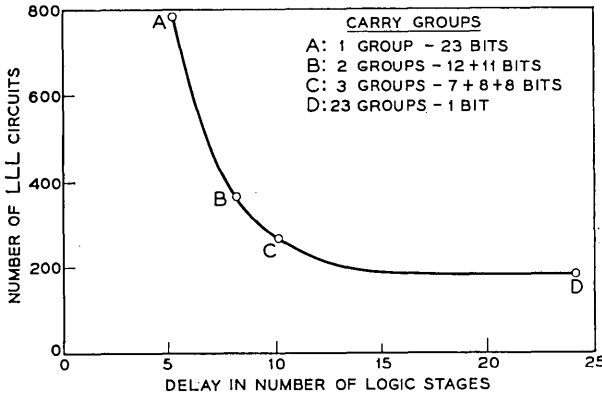


Fig. 17 — Circuit requirements and delay of 23-bit adders.

into groups of 11 and 12 bits, and indicates better than 2:1 savings in equipment over the circuit in A at the cost of  $\approx 200$  nsec. Point C defines the central control adder. It requires 270 LLLs and has a delay of 65 nsec, determined on the basis of 65-nsec worst-case transistor switching speed.

The block diagram of Fig. 18 shows the adder circuit used in the central control. For maximum economy in satisfying the speed requirements, the carry circuit is divided into three groups: one produces

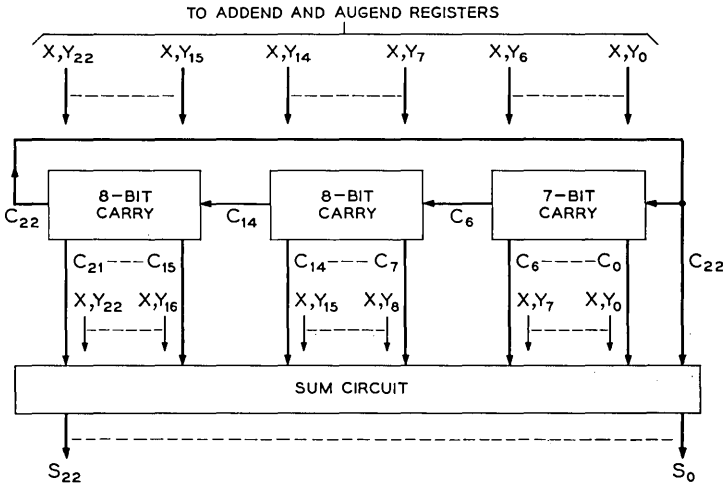


Fig. 18 — Block diagram of adder.

simultaneous carry outputs from seven bits, the other two groups from eight bits apiece. A single carry propagates from one group to the next in tandem, with the third group rejoined to the first in order to provide the end-around-carry connection. With each group after the first contributing the delay of two logic stages, and an added two stages of delay caused by the end-around-carry provision, the maximum propagation time is that of ten stages or, at worst, 650 nsec. Maximum propagation delay occurs when a carry terminates in the same group from which it had originated, after having propagated through the other groups. The addition of  $+1$ ,  $(00 \cdots 01)$ , to  $-0$ ,  $(11 \cdots 11)$ , is an example where the initial carry in the least significant bit causes a carry to ripple through all succeeding bits, until the end-around connection returns it to the origin where it terminates. The correct result is, of course,  $00 \cdots 01$ , or  $+1$ .

The end-around-carry connection between carry groups constitutes a feedback loop which can give rise to oscillation when two binary numbers, one of which is the exact complement of the other, are to be added. The problem exists because the carry circuit reflects the momentary conditions of its input registers at all times. The closed loop constitutes a delay line which can inject a remembered carry indication at the exact moment when a register is changing state. Normally, the circuit stabilizes when the register state has settled down due to the further generation of new carry terms, but when the numbers are complementary, no carries are initiated and the remembered carry could continually circulate or at least introduce an error. This problem is eliminated by inhibiting all carries (opening the feedback loop), during the transition period of either of the input registers.

The sum circuit combines the carries with the operand bits in the two-stage logic circuit shown in Fig. 19. The first stage of the circuit combines the inputs in four LLL gates as a sum-of-products which when inverted yields:

$$\text{SUM}' = X_i Y_i' C_{i-1} + X_i' Y_i C_{i-1} + X_i Y_i C_{i-1}' + X_i' Y_i' C_{i-1}'.$$

The output of this stage remains at the common-collector potential ( $+4.5$  v) when at least one input to every product term is at ground potential. Logically, the equation states that the sum is 0 whenever any two of the operand bits or the input carry are both equal to 1, or when all three are zero coincidentally.

An inverter amplifier stage is used to make the complementary sum output available to the connecting logic.

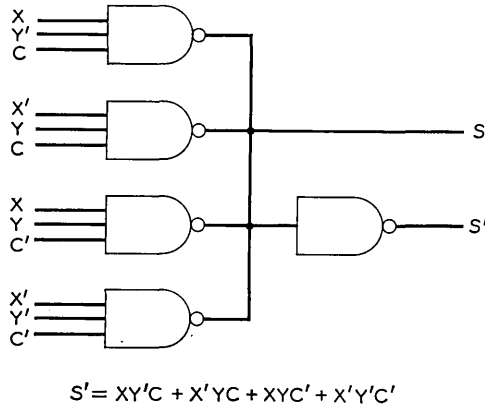


Fig. 19 — Logic diagram of sum circuit.

8.2.4 Special Logic Functions

The sum circuit logic contains each of the four possible combinations of the input variables  $X$  and  $Y$  in product with a carry term or its complement. By forcing the carry terms to become high ( $C = C' = 1$ ), the equation for the sum degenerates to

$$\text{OUTPUT}' = X_i Y_i' + X_i' Y_i + X_i Y_i + X_i' Y_i'$$

When appropriate control functions are introduced into each product term, the result can be expressed as

$$F' = f_1 X_i Y_i' + f_2 X_i' Y_i + f_3 X_i Y_i + f_4 X_i' Y_i'$$

This is the familiar expansion of two variables; it can generate any of the sixteen functions of  $X$  and  $Y$ , depending on the values assigned to the control functions.

Control inputs were added to the LLLs in the sum circuit to generate the logical combinations required in the CC. The condition that  $C = C' = 1$  was satisfied by grounding inputs to the appropriate logic packages in the carry circuit. Fig. 20 shows the sum circuit with the control inputs to generate logical AND, OR, and EXCLUSIVE-OR. None of the functions here required the condition of  $f_4 = 0$ ; therefore it was not provided. To obtain a high output when performing the AND operation, we let  $f_3 = 0$ . Then

$$F'_{XY} = f_1 XY' + f_2 X'Y + X'Y'$$

When  $f_1 = f_2 = f_3 = 0$ :

$$F'_{(x+y)} = X'Y'$$



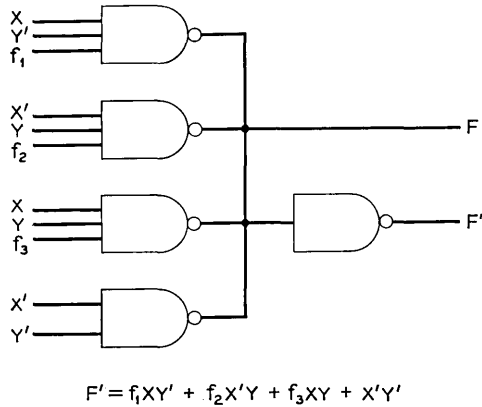


Fig. 20 — Modification of sum circuit to generate special logic functions.

which is high only when the bits satisfy OR logic. Finally, to obtain EXCLUSIVE-OR, let

$$f_1 = f_2 = 0:$$

$$F'_{(X \oplus Y)} = f_3XY + X'Y'$$

which is accordingly high for  $XY' + X'Y$  as required.

Since the carry circuit is inoperative when forming the logical combinations, the delay time is merely that of the two-stage sum circuit or 130 nsec maximum.

### 8.2.5 Shift-Rotate Functions

When manipulating data in the central control, it frequently happens that particular bits of information are not located in the bit positions that are most convenient to use. This commonly comes about, for example, when several small data words are packed together into adjacent bits of a larger memory word. Product masking is effective in isolating the desired data word from the rest of the group, but does not affect the relative bit positions. To provide the ability to relocate all bits of a word in some specified manner, shift register features have been designed into the accumulator.

The two basic types of operation in the accumulator register are rotation and shifting. Full rotation indicates a displacement of the 23 bits of a word by any instructed amount from 0 to 23 positions. Those bits which pass through one end of the shift register are reinserted into the other end so that no information is erased in the process. A modified,

form of rotation is confined to be effective only within a selected group of sixteen bits. In both cases, the rotation can be specified to be in either a left or right direction. Shifting is also a displacement of all 23 bits in either direction, but here the bits are forced out of the end of the register and the vacated spaces at the opposite end are replaced with zeros.

The accumulator is activated by synchronous commands from the rotate-shift translator, which receives the basic instruction from the order word decoder and a 6-bit descriptive word from the Q register. The five least significant bits of this word are the binary equivalent of the amount of displacement; the sixth, the sign bit, determines the direction of motion. When the sign is 0 (plus), motion is to the left; when it is 1 (minus), motion is to the right. In the latter case, the accompanying five bits are in complement notation and signify a negative number. To obtain the true displacement of a negative quantity, the rotate-shift translator must recomplement these five bits.

The program instruction itself can also specify that the shift or rotation take place in complemented form. That is, under the influence of a "shift complemented" instruction, not only will the magnitude of the displacement be complemented, as in the case for negative numbers, but the direction of the motion will also be reversed. Table VI summarizes the various forms of motion which are obtained.

### 8.2.6 *Derivation of Shift Mechanism*

Only 4.0 microseconds may be allotted to the shifting operation, so that the output of the accumulator can be available for some other operation in the cycle following the shift instruction. This interval includes the time absorbed in receiving and translating the instruction and the description of motion, as well as that allowed for the mechanics of shifting and register settling. Conventional series shift registers, in

TABLE VI — TRANSLATION OF SHIFT-ROTATE INSTRUCTIONS

Instruction in Order Word Decoder	Q Register Shift Code		Bit Motion in Accumulator
	Sign Bit	Magnitude	
Shift	0	00001	1 pos. to left
Rotate	1	11110	1 pos. to right
Rotate 16 bits only			
Shift complement	0	00001	1 pos. to right
Rotate complement	1	11110	1 pos. to left
Rotate 16 complement			

which a single position shift of all bits is triggered by successive clock pulses, were thereby ruled out. Instead, a semiparallel operation was used which allows all bits to be relocated by discrete groups. This reduces the number of shift operations.

Shifts involving any number of positions, within the 23-bit capacity of the accumulator, can be formed by successively adding together any three of the integers 0, 1, 4, 7, and 8, if we include the possibility of using an integer repeatedly. For example:  $19 = 4 + 7 + 8$ ;  $9 = 7 + 1 + 1$ , or  $9 = 8 + 1 + 0$ ;  $3 = 1 + 1 + 1$ ; and so on. Based on shift groups of these values, all accumulator movements are realized as a combination of, at most, three jumps. The 0 group is trivial, of course, since it requires no shifting action.

After determining the direction of motion and performing any resultant complementation, the shift circuit translates the binary magnitude of the remaining five bits and sorts the results into the four categories. Each bit in the accumulator shift register is connected to its neighbors on both sides which are 1, 4, 7, and 8 positions away. Fig. 21 is a sketch of the bit positions of the accumulator showing the shift register connections for bits 10 and 17. Each bit provides an input to eight other positions, four on each side, to allow bidirectional motion. Bit 17 illustrates the circular nature of the rotation process, whereby a left rotation of 7 positions causes bit 1 to assume the value of bit 17. All bits of the register would be connected similarly for the shift orders and 23-bit rotation orders.

Further interconnections are required for the 16-bit rotation instructions that confine the rotation process to the particular 16 bits between positions 6 and 21. As shown in Fig. 22, a left rotation of 7 positions from bit 17 in this mode requires a connection to bit 8. Similarly, bit 10 is connected to bit 19 as well as to its other destinations. Double connections are shown between bits 10 and 18, and bits 17 and 9, to indicate that these destinations are equidistant from the source bits in

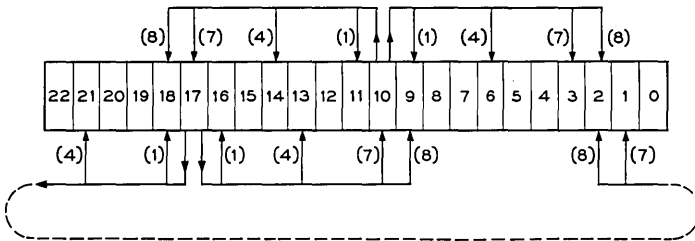


Fig. 21 — Interconnection of accumulator bits for rotate-shift functions.

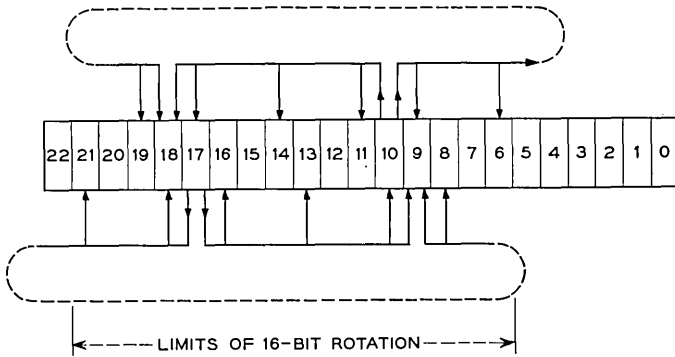


Fig. 22 — Bit interconnections for 16-bit rotation mode.

both directions. Bit 18, for example, assumes the state of bit 10 either in response to the instruction “shift (or rotate) all bits 8 positions to the left,” or in response to “rotate 16 bits a distance of 8 positions to the right.” These and similar patterns for other bits are redundant entry conditions and have been logically simplified in the final design.

Both the translated shift description and the bit interconnection logic are assembled in the final stages of the shift-rotate circuit. The results are combined with the three required clock pulses to trigger the synchronous operation of the shift register. Fig. 23 is a timing diagram illustrating the sequence of events which occur as a result of a shift or rotate instruction. During the initial 0.75- $\mu$ sec interval, the instruction is recorded and the Q register is loaded with the 6-bit word that defines the direction and amount of shift that is to be applied to the binary word in the accumulator. Suppose that a shift instruction had been given, specifying that all accumulator bits be displaced 13 positions to the left. In binary form, the 6-bit description is 001101,

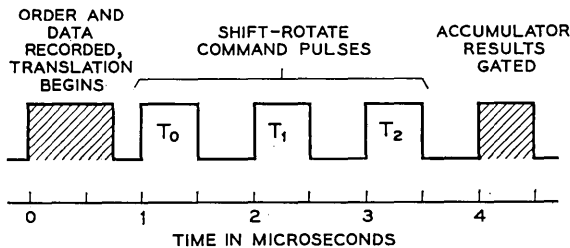


Fig. 23 — Timing diagram for shift register.

where the most significant bit signifies that motion is to be to the left. The shift translator decomposes the binary code into commands of jumps of 1, 4, and 8 positions. The first clock pulse of the chain,  $T_0$ , combines with the "shift 1" command, and the circuits in the accumulator prepare to respond. Activity commences on the downward swing of the  $T_0$  pulse, causing all bits to move simultaneously one position to the left. At the same time, a 0 is inserted into bit 0. The ensuing 0.5  $\mu$ sec interval allows the shift register stages adequate time to settle down in their new states.  $T_1$  is combined with the "shift 8" signal. At its conclusion, all bits move 8 positions to the left, with 0's being inserted into the vacated spaces at the right. Finally,  $T_2$  combines with "shift 4" to complete the action. Within 0.5  $\mu$ sec after  $T_2$ , the new accumulator contents are gated out and the register is available for other use.

Table VII shows the translation of the shift magnitude from binary to the 1, 4, 7, 8 code and indicates the interval when each jump occurs. There are several limiting conditions which have not been previously

TABLE VII — TRANSLATION CODE FOR SHIFT-ROTATE

Shift-Rotate Code	Displacement	Jumps/Unit Time		
		$T_0$	$T_1$	$T_2$
0 0 0 0 0	0	0	0	0
0 0 0 0 1	1	1	0	0
0 0 0 1 0	2	1	1	0
0 0 0 1 1	3	1	1	1
0 0 1 0 0	4	0	0	4
0 0 1 0 1	5	1	0	4
0 0 1 1 0	6	1	1	4
0 0 1 1 1	7	0	0	7
0 1 0 0 0	8	0	8	0
0 1 0 0 1	9	1	8	0
0 1 0 1 0	10	1	1	8
0 1 0 1 1	11	0	7	4
0 1 1 0 0	12	0	8	4
0 1 1 0 1	13	1	8	4
0 1 1 1 0	14	0	7	7
0 1 1 1 1	15	0	8	7
1 0 0 0 0	16	8	8	0
1 0 0 0 1	17	8	8	1
1 0 0 1 0	18	7	7	4
1 0 0 1 1	19	7	8	4
1 0 1 0 0	20	8	8	4
1 0 1 0 1	21	7	7	7
1 0 1 1 0	22	7	7	8
1 0 1 1 1	23	8*	8*	8*
1 1 0 0 0	None			

\* Results in resetting accumulator on shift instruction. No action on rotate instruction.

discussed. When either 0 displacement or a magnitude greater than 23 is specified, the shift-rotate mechanism is inhibited. An exception is the 16-bit rotation instruction, which rotates the bits "modulo 16," i.e.,  $n - 16$  positions.

## IX. EQUIPMENT DESIGN

### 9.1 *General*

Apparatus counts based on early block diagrams showed that the central control would be so large as to require precise control of logic circuit locations and their interconnections. With these concerns in mind a "breadboard" version of the central control's arithmetic system was built and tested. This unit, dubbed SPADE (Stored Program Arithmetic Digital Exerciser), contained wired LLL circuit packs tailored to meet many different combinational patterns and utilized point-to-point wiring between connector terminals.

SPADE provided the testing ground for prototype semiconductors, LLL combinational patterns, and many system concepts. The four-bay central control reflects the circuit pack designs, wiring patterns, and logic pack arrangements resulting from these early studies.

### 9.2 *LLL Circuit Packs*

Plug-in circuit packs<sup>8</sup> to accommodate the various combinations of gates, flip-flops, etc., were designed to allow rapid isolation and replacement of suspect components, thereby facilitating system maintenance. The design approach was to provide accommodations for various quantities of separate, independently accessed LLL gates on specific packs. Logic combinations are formed by interconnecting suitable gates at the plug-in connector terminals. This approach is in contrast to that of designing a large variety of low-usage packs in which the gates are internally wired to generate specialized functions.

To achieve high density, it is important to place as many logic gates as possible on a circuit pack and to use them with the greatest efficiency. The controlling variables in the design are the circuit pack area and the number of connector terminals available (28). With these limitations, the most densely packed logic element contains eight two-input gates. This pack, shown with its apparatus mounting in Fig. 24, has the largest usage in a No. 1 ESS office. Table VIII contains a listing of other LLL-type circuit packs used in the system, predominantly in the central control.

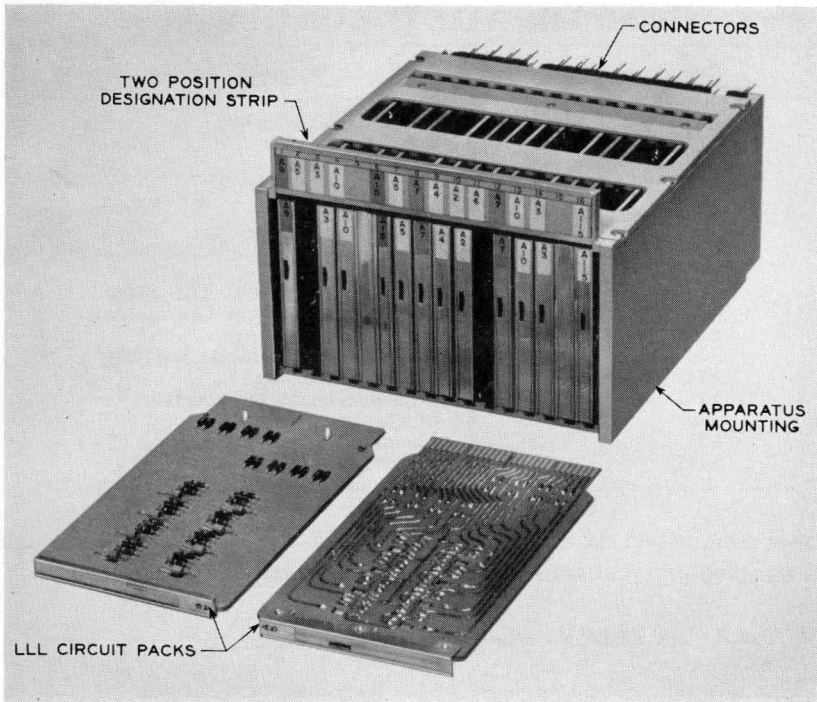


Fig. 24 — LLL circuit pack and mounting.

A measure of the success of logic circuit standardization is best illustrated by the fact that sixteen logic circuit packs, making up about 10 per cent of the entire No. 1 ESS catalog, account for more than 50 per cent of the packages and 70 per cent of the semiconductors used in a typical office. In the No. 1 ESS office at Succasunna, New Jersey, 6,675 of these sixteen types are required, the duplicated central controls alone accounting for 4,036.

Previous sections have pointed out that LLL gate outputs can be connected together (with certain restrictions) to form a common driving AND node. When this is done, a single dummy load resistor (Section 4.4) is sufficient to maintain the proper operating conditions for the combined circuits. To facilitate this use, the resistors are omitted from the logic packs and are assigned instead to resistor packs. These contain twenty-four separate collector resistors commonly connected to the +4.5-v collector source. A single LLL gate then receives its base drive through its own internal base resistor supplied from a common +24-v

TABLE VIII — LLL-TYPE CIRCUIT PACKS

Code	Description
A1	2 9-input LLL gates
A2	3 7-input LLL gates
A3	4 5-input LLL gates
A4	6 3-input LLL gates
A5	8 1-input LLL gates
A6	8 2-input LLL gates
A15	8 1-input LLL gates (individual power connections)
A7	8 1-input high fan-out LLL gates
A8	8 2-input high fan-out LLL gates
A11	1 binary counter stage
A13	2 flip-flops with input gating
A14	1 shift register stage
A18	8 high-sensitivity bus receivers
A19	8 bus receiver amplifiers
A21	8 bus driver transformers
A177	8 bus driver LLL gates

power terminal on the connector and its +4.5-v collector source through an assigned external resistor at the collector output terminal.

### 9.3 Circuit Pack Interconnection Practices

The interconnection of logic gates to generate a circuit function is dependent upon the over-all circuit organization and the diagnostic and maintenance techniques which are applied in its operation.

Because of its word-organized design, faults and errors in a CC are most easily diagnosed to the troublesome bit position in a word. It seems only natural, then, that as many logic gates as possible associated with the same bit of a word be assigned to the same circuit pack. In effect, this approach to logic gate assignment keeps to a minimum the number of circuit packs which must be tested to isolate a particular fault. Unfortunately, assigning logic gates to circuit packs on a word-bit basis is often incompatible with the approach which assigns logic gates on the basis of achieving high circuit pack density. As a result, it is often necessary to compromise and to assign two or more bits to the same circuit pack.

Output leads from flip-flops, binary counters, and shift registers are critical because of their internal cross-connections to inputs; therefore the total wire length is restricted to a maximum of three feet. To some degree, this requirement also affects the assignment of logic gates to circuit packs. The following are examples of circuit pack assignment practices as applied to the central control.

Fig. 25(a) shows a common-collector configuration where all logic



gates can be assigned to the same circuit pack if no input lead from a register exceeds three feet.

Fig. 25(b) shows a common-collector configuration where all logic gates are not assigned to the same circuit pack. These logic gates are assigned to three different circuit packs as follows: one 7-input logic gate to an A2 circuit pack, three 5-input logic gates to an A3 circuit pack, three 3-input and two 2-input logic gates to one A4 circuit pack. In this case, the 3-input circuit pack is used where the circuit logic requires 2-input gates. Had the 2-input gates been assigned to an A6 pack, four types of circuit packs would have been required for this word bit, rather than three.

9.4 Frame Wiring Methods

During the brassboard development stage it became evident that the shop and field wiring practices of relay switching systems could not be

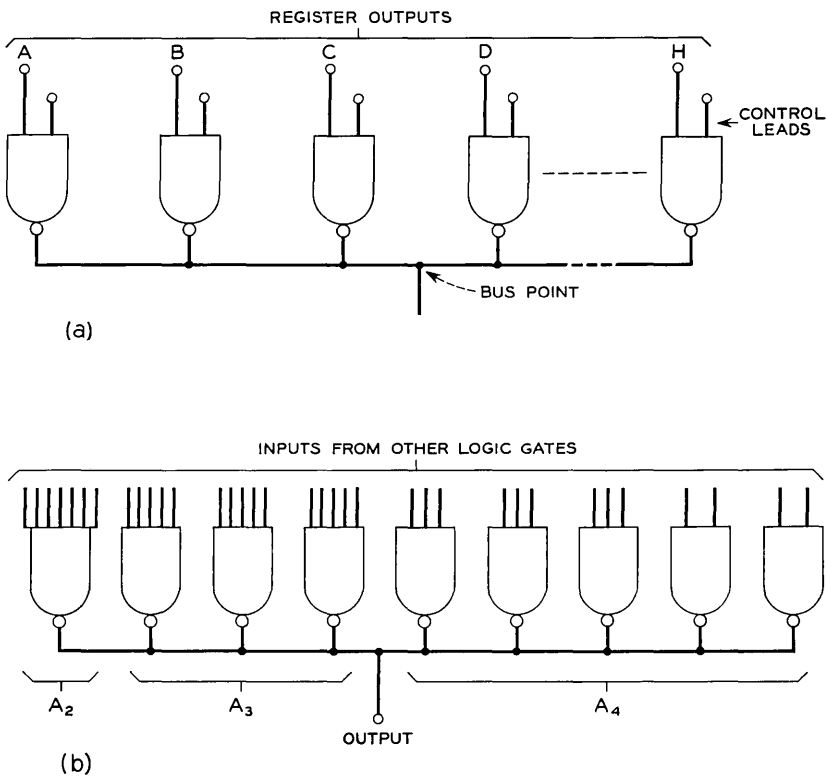


Fig. 25 — Common-collector configurations.

used for the high-speed pulse circuit wiring of central control. The lengths and parallel spacings of signal wires had to be controlled to minimize electrical interference (crosstalk) between circuits. This resulted in circuit pack placement restrictions and the development of new wiring methods suitable for this type of equipment.

Where high densities of circuit packs and wired interconnections exist, special wiring procedures are employed which dictate specific routings for surface wiring, loose wiring, and local cables. This eliminates the shop wiring variability that could appear in surface wiring by different operators.

The individual mounting plates are first surface-wired in the shop and verified with a dc test. The associated mounting plates of a unit are next surface-wired together. The loose wiring support details are then mounted on the units, and the units in turn installed on the bay or frame. A preformed loose wiring harness is mounted on the support details and the leads are terminated.

The loose wiring support details are epoxy-coated metal combs and fingers which are grounded to the mounting plate via their mounting screws. These provide an extension of the ground plane to minimize inductive coupling between the loose wire signal paths. The high wiring density shown in Fig. 26 requires the use of no. 26 gauge wire.

An analysis of the CC wiring disclosed the following distribution on the four-bay frame:

(a) surface wiring per mounting plate (average): 450 wires between connectors on the same plate, and 100 wires between connectors on different mounting plates of the same unit

(b) loose wiring per mounting plate: 50 wires between connectors on different units.

(c) local cable wiring per mounting plate: four wires to fuse panel

(d) interframe loose wires between bays 1 and 2: 750 wires.

Approximately 43,300 wires or 86,600 wrapped connections are required within the four bays.

### 9.5 *Grounding Techniques*

Because of the rather small voltage swings necessary to control the transistor circuits, and the speed with which they react, the grounding technique is very important. A very low impedance (inductance) ground plane is required to prevent excessive noise voltages (a few tenths of a volt) from being generated by the high-speed switching circuits in the CC. For this reason, the equipment frame is utilized as

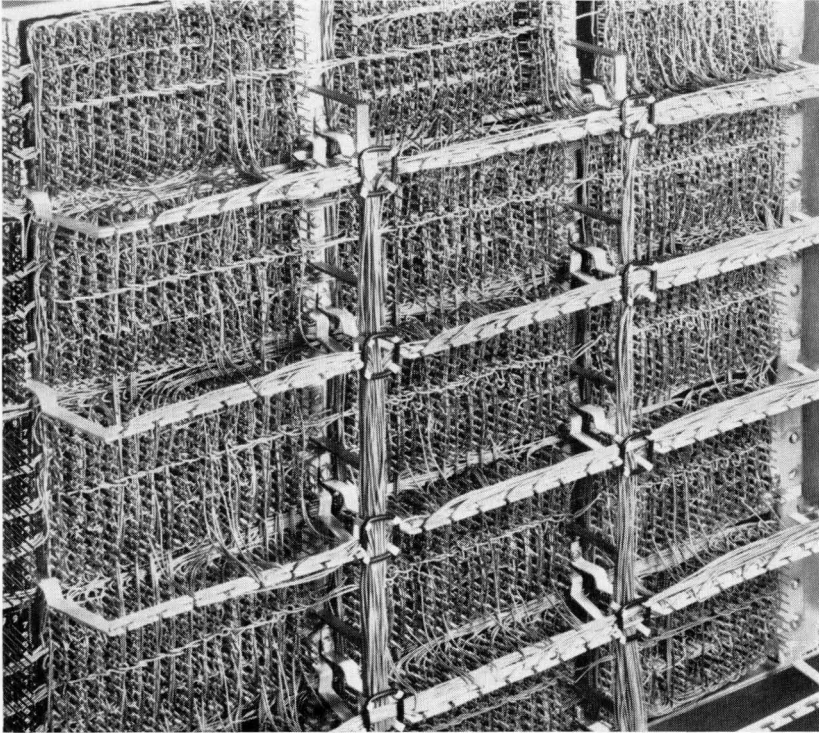


Fig. 26 — Photograph of central control wiring.

a circuit ground. A ground terminal intimately connected to the frame is located immediately below each circuit pack. Each circuit pack has its circuit ground tied into frame ground with a wire approximately three inches in length. These precautions have reduced the ground noise internally generated in the CC to a level where it is not a problem. The power and grounding arrangements in the system are relied upon to keep extraneous circulating currents from creating disturbing voltage drops in the framework.

#### X. SUMMARY

The design of the central control has been discussed in terms of its influence upon the design of building block logic circuits and their utilization in performing some of the specialized functions which are carried out in the process of controlling a telephone switching system. The complexity of the unit made a detailed description of each internal

circuit beyond the scope of this paper; therefore many features have had to be omitted.

The emphasis for reliability has been perhaps the most important factor dictating all CC design decisions. A major part of the circuit and equipment design effort was directed toward enhancing dependability by prescribing inherently reliable circuits, components, and equipment. The logic circuit design provides standardized but versatile building blocks with fast, low cost performance. Over  $10^9$  LLL gate hours have been logged during the No. 1 ESS preoperational phase. The failure rates during this phase are, as might be expected, higher than will be experienced when installation troubleshooting is completed. Based on current reliability predictions, about one component failure per operating month is anticipated in each CC. Because of the internal fault recognition and self-diagnostic maintenance features of the system, along with the provision for rapid repair by circuit pack replacement, this failure level is sufficiently low to insure the requisite system dependability.

#### XI. ACKNOWLEDGMENTS

The authors wish to thank their many colleagues who contributed their efforts and talents to the development of this project. Particular credit is due R. C. Buschert, S. F. Sampson, and S. M. Neville for their work in evaluating the devices and developing the basic circuits; also to R. W. Donalies, R. A. Estvander, E. Graeve, F. R. Huff, A. V. Jensen, R. Korte, A. Perretto, and J. Pindroh for logic designs; to A. C. Luebke, I. W. Morris, R. Nelsen, W. E. Smith, A. M. Wagner, and H. J. Wirth for equipment designs; and to J. B. Connell, A. H. Doblmaier, R. W. Downing, M. P. Fabisch, J. S. Nowak, F. F. Taylor, and W. Ulrich for their cooperation in relating the CC functional requirements to practical designs. The assistance of the semiconductor device development areas in the practical characterization of the semiconductors is also gratefully acknowledged.

#### REFERENCES

1. Downing, R. W., Nowak, J. S., and Tuomenoksa, L. S., No. 1 ESS Maintenance Plan, B.S.T.J., this issue, p. 1961.
2. Connell, J. B., Hussey, L. W., and Ketchledge, R. W., No. 1 ESS Bus System, B.S.T.J., this issue, p. 2021.
3. Harr, J. A., Taylor, F. F., and Ulrich, W., Organization of No. 1 ESS Central Processor, B.S.T.J., this issue, p. 1845.
4. Yokelson, B. J., Cagle, W. B., and Underwood, M. D., Semiconductor Circuit Design Philosophy for the Central Control of an Electronic Switching System, B.S.T.J., **37**, Sept., 1958, pp. 1125-1160.

5. Cagle, W. B., and Chen, W. H., A New Method of Designing Low-Level, High-Speed Semiconductor Logic Circuits; I.R.E. Wescon Convention Record, Vol. 1, Part 2, 1957, pp. 3-9.
6. Jarvis, D. B., The Effects of Interconnections on High-Speed Logic Circuits, IEEE Trans. Electronic Computers, Oct., 1963.
7. Dodson, G. A., and Howard, B. T., High Stress Aging to Failure of Semiconductor Devices, Proc. 7th Natl. Symp. on Reliability and Quality Control, Jan., 1961, pp. 262-267.
8. Chevalier, J. G., and Eisenhart, R. K., No. 1 ESS Circuit Packs and Connectors, B.S.T.J., this issue, Part 2.



# No. 1 ESS Program Store

By C. F. AULT, L. E. GALLAHER, T. S. GREENWOOD  
and D. C. KOEHLER

(Manuscript received January 15, 1964)

*Line and trunk translation data and operating programs for No. 1 ESS are stored in a large semipermanent memory. This memory is provided by modular units known as program stores. Each program store provides 5.8 million bits of randomly accessible permanent magnet twistor memory organized into 131,072 parallel words. The information is stored in the state of small magnets affixed to aluminum cards. Each card contains 64 forty-four-bit words.*

*Each store is designed to operate over a duplicated common bus system for both normal and diagnostic operations. The stores have a cycle time of 5.5  $\mu$ sec. Such stores are an attractive and economical solution to the problem of providing large storage capacity for information which must be protected against accidental change.*

*To provide an efficient and routine method for updating the information content of such stores, offices are provided with card writing equipment. This includes both card handling equipment and card magnetizing equipment under system control.*

## I. GENERAL

### 1.1 Storage Requirements

No. 1 ESS is under control of a very large program. The complete program may require a storage capacity in excess of 4 million bits. In addition, the office must store a considerable amount of data about each line and trunk. Such data as directory number, class of service, special feature lists, etc. must be available internally to the system. The storage needs for these items are variable with office size and range from 1 to 14 million bits. This program and translation information together constitute (a) the knowledge necessary to perform the telephone switching function and (b) the memory of the service commitment to each customer. To ensure service continuity this information must be

protected from accidental destruction by either equipment malfunction or operator error.

The system must have direct and immediate access to all of the information. However, it will be noted that the system does not require the direct capability of altering the information. This is because the information and its changes are generated externally to the system, as by a program designer or the telephone business office. What is required is a suitable way of introducing the information changes into the system.

To meet this need of an economical, high-capacity, random-access memory, No. 1 ESS uses permanent magnet twistor modules as basic storage elements. These provide a memory that is fundamentally "read-only" while at the same time providing a simple and straightforward way of replacing old information with new. As desired, no electrical malfunction can alter the information content.

### 1.2 Store Objectives

Twistor modules, with their circuitry, form an ESS unit designated a "program store." The general objectives set for the development of program stores are given in Table I. The word size of 44 bits allows 37 bits of information and 7 bits of redundant encoding.<sup>1</sup> The high-reliability requirement, coupled with past experience in the Morris, Illinois, ESS trial, indicated strongly the desirability of high redundancy in program information. The coding used includes a Hamming single error correcting code plus an over-all parity, both taken over the data and its storage address. While with equipment operating normally this much redundancy may seem extravagant, the ability to operate under serious degradation of circuit or memory, as well as the enhanced ability to detect and isolate malfunction, is felt to more than repay the cost.

The capacity of 131,000 words represents the basic storage needs of a small office. It was felt that this size also represented about the largest size that could be achieved with common-access circuits. The cycle time of 5.5  $\mu$ sec was essentially determined by the over-all system,

TABLE I — ESS PROGRAM STORE OBJECTIVES

Word size	44 bits
Capacity	131,000 words
Speed	5.5 $\mu$ sec
Cost	low
Reliability	high
Maintenance	automatic
Power	battery
Environment	non-air-conditioned



including the twistor modules, but did not represent an attempt to build the fastest store possible. Low cost is of course an inherent objective; an over-all economic balance in the system dictated the need to achieve a cost per bit of the order of one-fifth that of the system's variable memory.

Programmed fault diagnosis is an essential feature of an ESS. In the case of a program store, the interaction between the fault and the fault-detection program can be especially severe. This requires that advance planning and coordination between the circuit, system, and programming engineers be especially effective.

The requirements of battery power and non-air-conditioned space are office-wide. They provide somewhat more of a challenge in the case of the program and call stores, where temperature-sensitive devices and nondigital circuits occur.

The over-all objectives, at the time they were established, represented a goal that would require significant device and circuit development. This article reports the successful attainment of that goal.

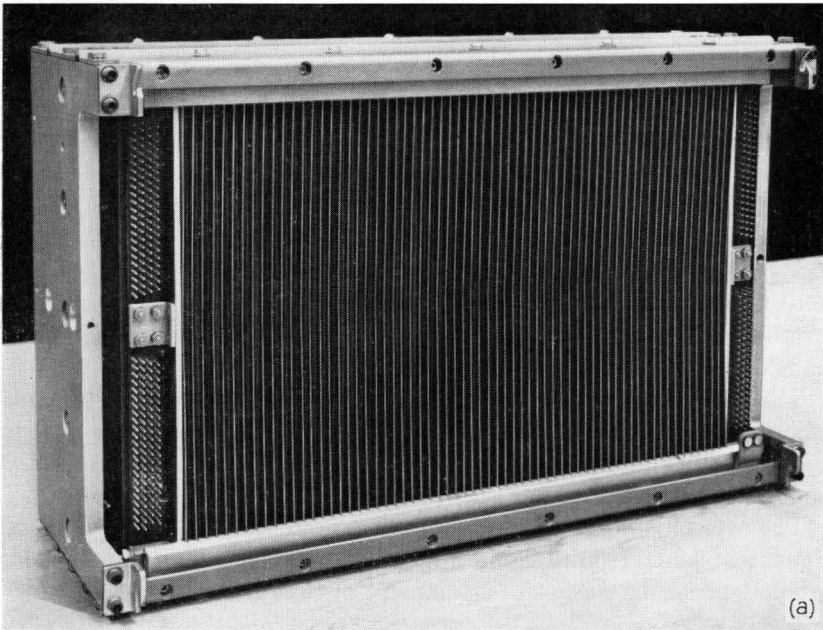
### 1.3 *Devices*

#### 1.3.1 *Semiconductors*

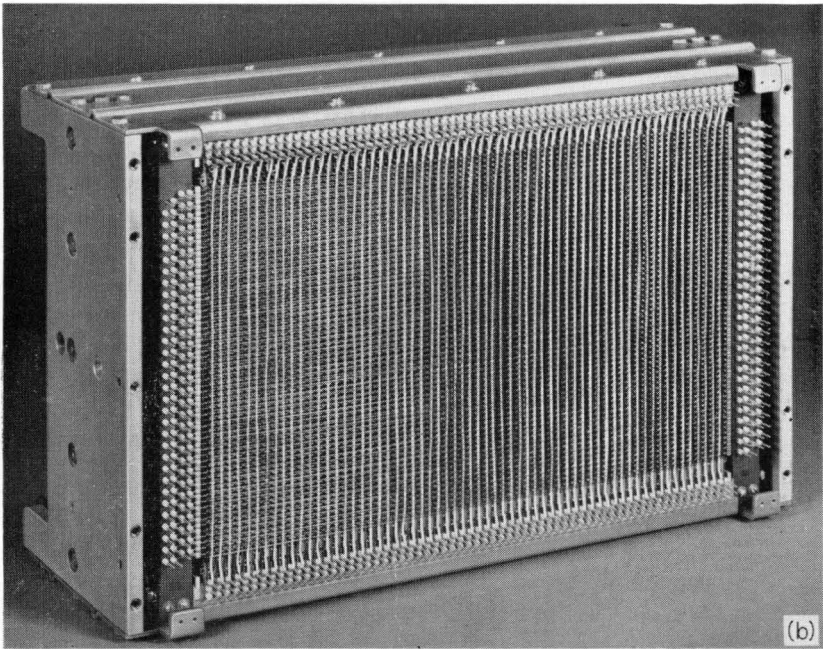
As in the remainder of the system, in digital or low-level applications the store used the single code of transistor developed for No. 1 ESS. However, it was clear at the outset that a fast, higher-powered transistor and companion diode would be required for program store access. For this purpose the 20D transistor and 426AC diode were developed. The 20D transistor, which can handle 1.35 amps with  $\frac{1}{3}$ - $\mu$ sec switching times and a breakdown voltage of 50 volts, also proved a suitable transistor for other fast, high-power needs of the system.

#### 1.3.2 *Twistor Modules*

The permanent magnet twistor memory has been previously reported.<sup>2,3,4,5</sup> The twistor modules used in ESS were a further development of the module described in Ref. 4. The front and rear views of a module are shown in Fig. 1. On the rear is a  $64 \times 64$  biased-core switch matrix. A particular core is switched by the combined action of a horizontal and vertical half-select current overcoming a common bias current. Each core is coupled to a strip solenoid referred to as a "word solenoid" (see Fig. 2). The word solenoids are attached to the surface of an insulating board over which have been placed thin sheets of permal-



(a)



(b)

Fig. 1 — (a) Front (card insertion side) of a 1A twistor memory; (b) rear (core access side) of memory.

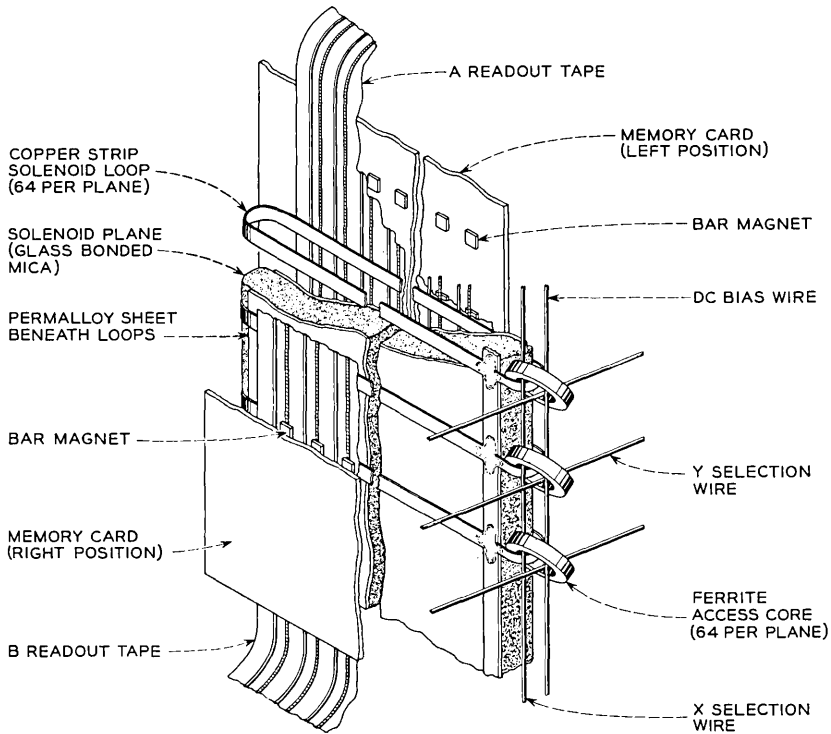


Fig. 2 — Cutaway schematic showing the basic elements of permanent magnet twistor memory.

loy. Each module contains 64 boards, each with 64 solenoids. Through the module, accordion fashion, run two flat plastic belts, each containing 44 twistor wires and 44 adjacent return wires. A section of each tape is cemented to one side of each board. Each of the modules provides space for 128 magnet cards, one of which is shown in Fig. 3. Each aluminum card carries 64 columns of 45 thin permanent magnets. Each column represents a word and each of the first 44 magnets, a bit of the word. (The 45th row is not used in this application.) When magnetized, each magnet represents a stored "zero." A demagnetized magnet represents a stored "one." When all cards are in place in the module, a magnet appears over each intersection of a twistor wire and word solenoid. If the magnet is magnetized, it fully saturates a region of the twistor wire beneath it. The magnets are always magnetized in the same direction as the initial field of the word solenoid.

When an individual word solenoid is selected by applying a half-

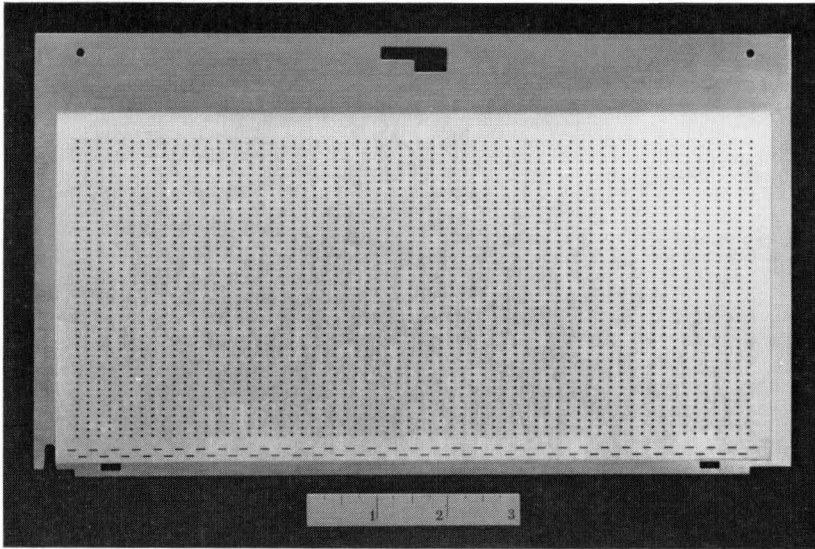


Fig. 3 — Twistor memory card.

select current to individual horizontal and vertical access wires, a current pulse is induced in the word solenoid. The resulting magnetic field acts on the twistor wire. Because of the orthogonal geometry no significant voltage is induced in the wire unless the magnetic material of the twistor wire is switched. This can occur only at the sites containing nonmagnetized magnets. When the half-select currents are removed, the common bias current switches the selected access core back to its initial state. The resulting word solenoid pulse restores initial twistor wire conditions.

It will be seen that the selection of a single word solenoid causes a readout of 88 bits. Within the readout circuits, it is thus necessary to select the 44 bits associated with the desired magnet card.

For uniform outputs from the memory, the magnetic field applied to the twistor wire should completely switch its magnetic material. In these modules the magnetic field produced by the solenoid is concentrated onto the twistor wire by two mechanisms. The first is the underlying permalloy sheet, which provides a low-reluctance return path for the field; the second is the conductivity of the magnet card, which produces, by eddy currents, a magnetic barrier above the twistor wire. These two mechanisms help to reduce the drive currents required and the interaction between bits.

In these modules, a 2.62 ampere-turn bias and half select are used. These produce a "one" output at the end of a twistor pair of 2.5 mv

across 300 ohms (far end short-circuited). For a bit with a magnetized magnet, essentially no output is generated in the twistor wire. However, because any selected access core is always accompanied by 126 half-selected cores, some "shuttle" or "delta" noise may be generated due to the summation of small voltages at each of these solenoids. This delta noise is information dependent. It may subtract from a "one" signal as well as create a "zero." The modules used are designed to insure a 2.5-to-1 ratio of "one" to "delta" for worst drive, temperature, and information pattern over the complete population of manufactured modules. Under nominal conditions of drive and temperature almost all bits will exceed 5-to-1 with worst pattern.

#### 1.4 *Store Organization*

A complete program store is shown in Fig. 4. The store circuitry contains three major divisions: access, readout, and control. Each of these will be treated in detail in this article. The function of each is as follows:

##### 1.4.1 *Access*

To achieve the required capacity, 16 twistor modules are used in the store. These are arranged in a  $4 \times 4$  array which results in a  $256 \times 256$  access core matrix. The access circuits must provide this array two pulses of accurately controlled shape, amplitude, and coincidence to cause the readout of the desired word.

##### 1.4.2 *Readout*

The readout section must provide the amplification of the low-level twistor signals and the selection and sampling necessary to provide the binary output word. The twistor wire signals all appear on terminals on the front surface of the modules. Because of the large area of the core matrix and the low level of the twistor signals, the readout leads must be well shielded to avoid interference. Ideally, the 88 twistor pairs from each module could be paralleled before amplification, since only one solenoid is accessed at a time. For two reasons, this is not done: first, it would result in a considerable reduction in available output power; second, and more importantly, the delta noise would be several times that from a single module. On the other hand, individual handling would require 1408 preamplifiers. In the interest of economy a compromise solution is used. The modules are paralleled in groups of four. Each module in the group is chosen from a different row and column so that no delta noise penalty is incurred. The 6-db loss in power is not great enough to significantly degrade signal-to-noise performance.

The selection of the desired group of 44 pairs from the four groups of 88 pairs is made after a small amount of amplification. A single group of

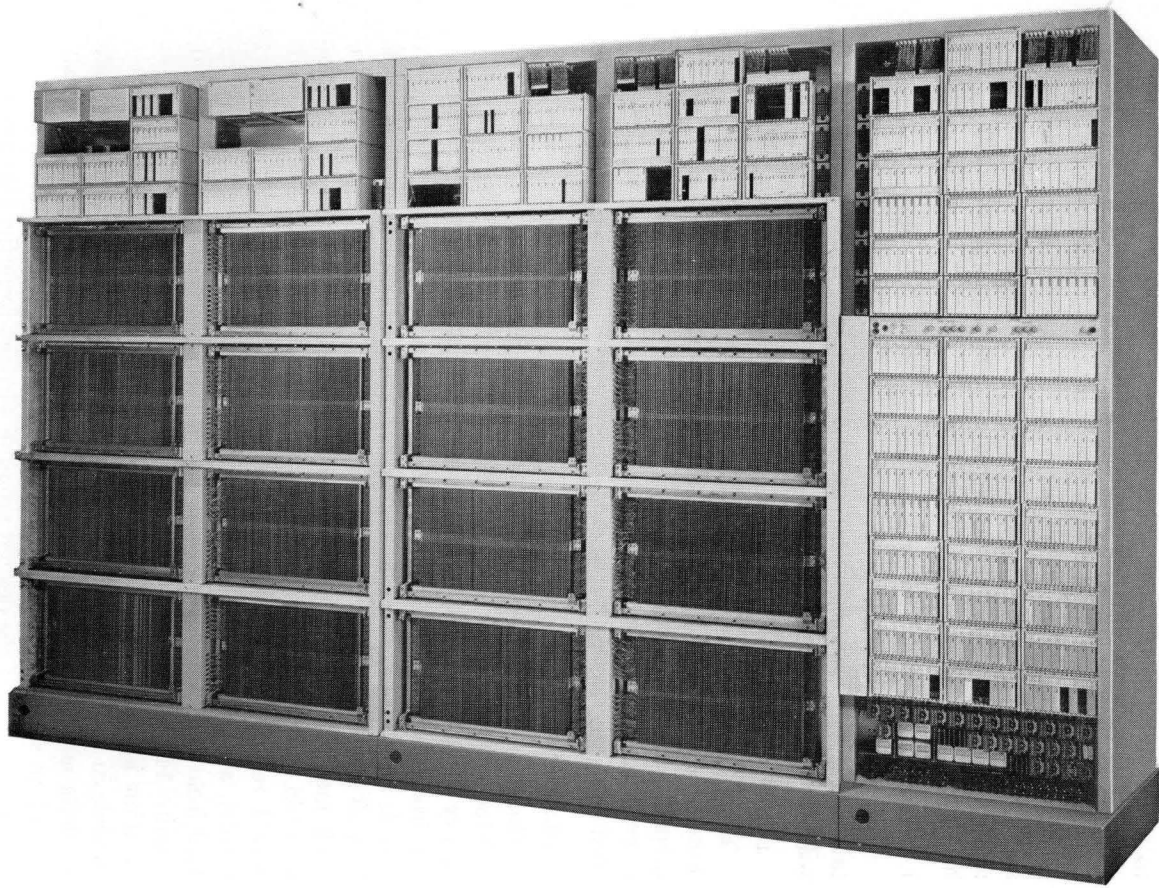


Fig. 4 — No. 1 ESS program store.

44 readout amplifiers and samplers is used to generate the final quantized one and zero signals. (See Fig. 5.)

### 1.4.3 Control

These circuits must provide all of the communications with the remainder of the system, both for normal action and maintenance. Additionally, they must make continual checks to detect abnormal conditions and initiate diagnostic action. Duplicate communication circuits to the central controls are provided. This duplication requires control circuits which must have further communication paths. To provide flexible growth a common bus system is used for all stores. Thus each store must have circuits for recognizing coded bus signals before responding. To insure that effective automatic diagnosis is possible, logical separation of circuit faults must be made possible by judicious provisions of circuit and communication redundancy. An indication of the complexity of this function is that nearly one-half of the store circuitry is in this section.

### 1.5 Card Writing

The insertion of information into the program store requires the magnetizing or demagnetizing of individual magnets. To meet the

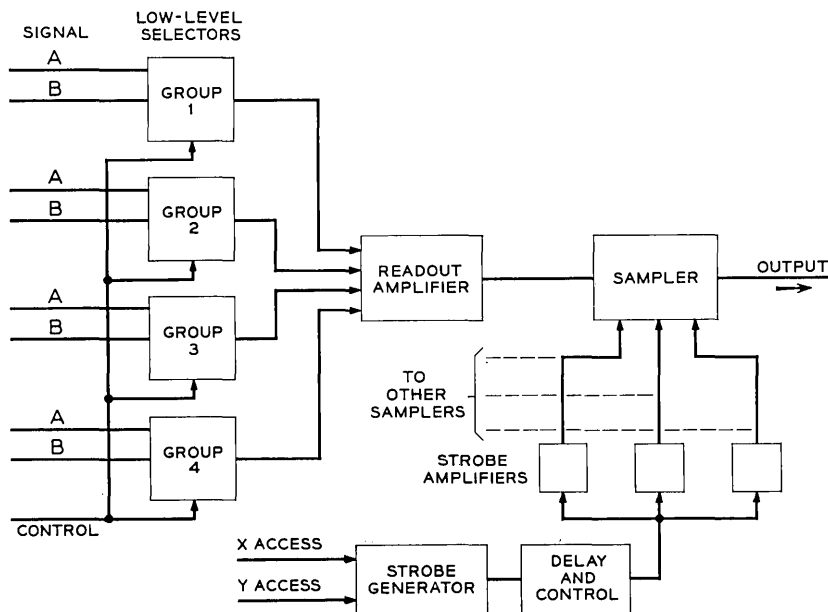


Fig. 5 — Readout block diagram.

system objective of a simple and straightforward method for accomplishing this, two auxiliary units are used: (1) the magnet card writer, which provides a mechanism for handling cards and a movable magnetic head for magnetizing or erasing magnets, and (2) a card loader which provides an automated method of removing or inserting a full module of cards from or into a store and supplying them to the card writer.

The card writer is used by the system as a peripheral unit. It is supplied information based on old program store information modified by the changes desired. Using this information the card writer magnetizes and erases a spare set of cards. Once prepared, this set may be substituted for the old set in the program store. This process of updating translation information in the office is straightforward and insures that the new information is correct before the old information is destroyed.

## II. STORE ACCESS

### 2.1 *General Philosophy*

#### 2.1.1 *Coincident-Current Word Selection*

Word selection is obtained in the No. 1 ESS program store by means of a coincident-current selection method utilizing a ferrite biased core switch. Normally, all of the access core switches are biased with 2.62 ampere-turns, which in the absence of other drives causes the ferrite core to be in a state of saturation. The core has a squareness ratio,  $B_r/B_{max}$ ,

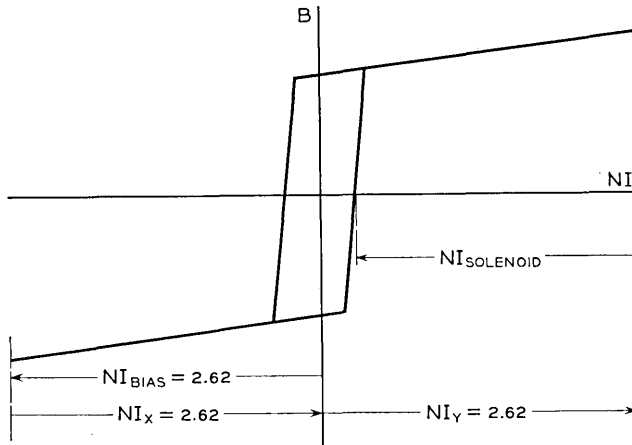


Fig. 6 — Operation of biased core switch.



of about 0.95, together with a coercive force ( $H_c$ ) of about 0.2 oersted. Thus the state of the core is much as shown in Fig. 6. Also shown are the driving forces,  $NI_x$  and  $NI_y$ , applied to the biased core switch. It is obvious from the figure that either drive by itself is insufficient in magnitude to cause any switching action in the core. On the other hand, if both drives are present at the same time, the core will begin to switch. During the switching interval, a voltage will be induced in the solenoid winding which will cause a current in this winding that will flow in a direction such as to oppose the drive current. Thus the magnitude of the ampere-turns drive in the solenoid is equal to the sum of the two drives minus the bias, the coercive force, and appreciable losses due to high-speed switching of the core.

If the drives were maintained on the biased core switch, the core would eventually saturate. For the duration of the pulse required ( $2 \mu\text{sec}$ ), a 400-mv- $\mu\text{sec}$  core provides ample margin for worst-case considerations of temperature, drive currents and solenoid impedances. Normally, the core is only partially switched at the time the drive currents are removed. Upon removal of the drive currents, the bias current switches the core back to its initial, or set, state. This results in a reverse current flow in the solenoid winding that resets all the twistor wire bits. Such action points out one great advantage of the biased core switch for memory access: unipolar drive currents in the core lead to bipolar pulses in the solenoid. This provides for automatic resetting of the memory word bits after interrogation.

#### 2.1.2 *Size of Access Matrix*

The size of the access matrix is necessarily a balance between economy and peak power requirements. Generally speaking, the cost of memory on a per bit basis is less as the size of the memory increases. This is particularly true of coincident-current access memories. The larger memories, however, require an increased drive power which results in several problems. One is the problem of generating the high-power drive pulses while meeting the system cycle time and another is the problem of interference between the access and readout signals.

The No. 1 ESS program store has a single access system. The access matrix is  $256 \times 256$ . That is, there are 256 X-drive windings and 256 Y-drive windings, together capable of selecting one of the 65,536 cores used as biased core switches. The load presented by the winding consists of three parts. The primary load consists of the air inductance of the access winding itself. The secondary load is the shuttling of the 255 cores which are not driven by the other set of drive windings. Of little con-

sequence is the load presented by the selected core. This is not to say that the core properties are not of great importance. The squareness ratio determines the load presented to drivers when the 255 cores are shuttled. The smallest core that has the requisite output is desirable, since larger cores increase the shuttling load.

The choice of the number of turns to be used on the biased core switches for the drive windings is dependent on several factors. The most important factor is the semiconductor to be used and its ratings. The 20D transistor is specified primarily for its needs in the access circuitry. The transistor has to have fast turn on and off times, with 1.35-amp current capability, and be able to interrupt a 50-v level without breaking down. These are approximately the values of current and voltage encountered with two turns per core. Operation with three turns requires excessive voltage ratings on the transistors and lowers the resonant frequency of the drive winding into the frequency band used by the pulse drivers. Single-turn operation would be desirable if an adequate transistor were available. Its advantages do not appear sufficient to warrant paralleling three 20D transistors, since this would entail added cost and circuit complications. As a result, a two-turn winding was chosen.

The circuit access matrix for one axis is shown in Fig. 7. It is basically a simple diode matrix with 16 access switches on each side of the matrix. A closure of one access switch each in the upper set and the lower set will establish a path through the matrix, thus selecting 1 out of 256 drive windings. As mentioned previously, each of the drive windings drives 256 cores. The matrix diodes prevent sneak paths from occurring which would subtract from the current pulse desired in the selected winding. They are normally back-biased to a voltage slightly higher than the peak drive voltage. This protects the access switch from a transient turn-on problem that could degrade the shape of the leading edge of the drive pulse. The diodes above the upper switches and below the lower switches also serve to protect the transistors of the associated switches from voltage pulses which could turn on a switch on a transient basis. These diodes are also normally back biased.

### 2.1.3 Access Waveform

Fig. 8 shows the basic waveforms of current and voltage as related to the twistor modules. Traces (a) and (b) represent the drive currents,  $I_x$  and  $I_y$ , that drive a particular core in the memory. Trace (c) shows the current in the word solenoid. The solenoid current shows the bipolar pulse characteristics plus the exponential recovery portion. The ex-

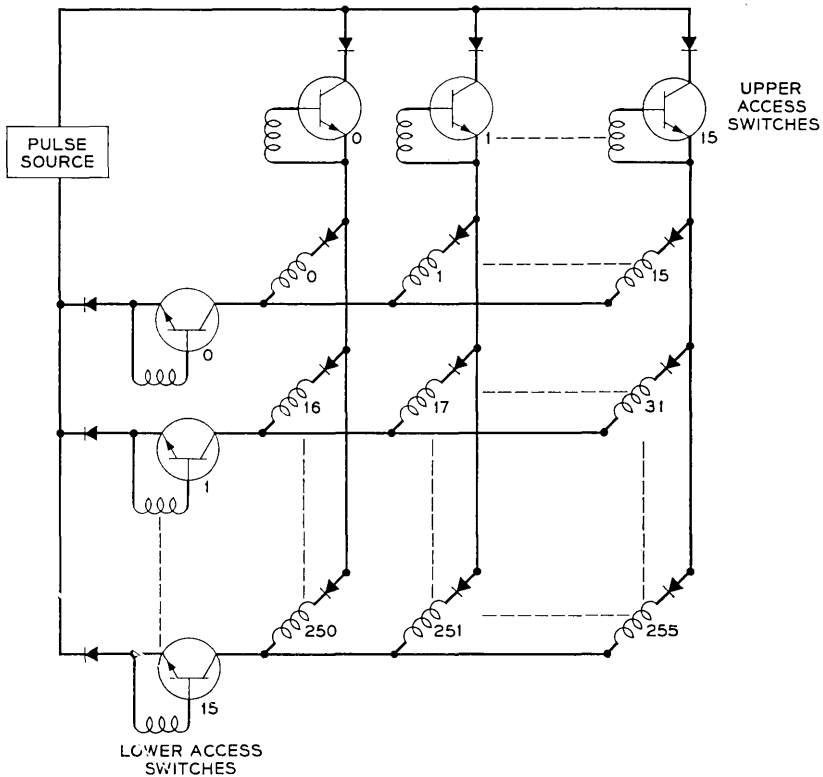


Fig. 7 — Access matrix for one axis of selection.

ponential decay occurs when the biased core switch returns to saturation. At this time the core is essentially a short circuit to the word solenoid. Thus the inductance and resistance of the solenoid determine the rate of decay of the solenoid current. This  $L/R$  time is important, since another interrogation cannot be started until the solenoid is nearly recovered. The program stores are operated at a  $5.5\text{-}\mu\text{sec}$  cycle rate in the No. 1 ESS system but could be operated as fast as  $4.5\ \mu\text{sec}$  without degrading the solenoid current waveforms.

The drive currents are "on" for  $2\ \mu\text{sec}$  in the program store. This is primarily so that the readout can be detected before the transient caused by turn-off occurs in the readout circuitry. The approximate waveform of readout from the memory is shown as trace (d). It is included for timing reference only and is explained in some detail in the readout section of this article. The drive currents must be left on long enough to

switch enough flux in the biased core switch to insure a sufficient reverse current in the word solenoid to reset the twistor bits.

The rise and fall times of the drive currents are approximately 0.5  $\mu\text{sec}$ . The shape of the current waveform is nearly trapezoidal in order to obtain the desired current level in a minimum amount of time without exceeding the drive transistor voltage limits.

## 2.2 Current Pulse Generating System

### 2.2.1 Basic System

The basic current pulse generating system for one axis consists of a constant-current source connected to a parallel circuit consisting of the load (access matrix) and a normally closed switch, as shown in Fig. 9. Normally the current source feeds current through the closed switch, A. When switch A is opened the current is forced to flow through the load and the normally closed B switch to ground. This presupposes that the proper access switches in the load section have been closed prior to opening switch A. Since the load is primarily an inductance, it is necessary to limit the voltage across the parallel circuit to prevent damage to switch A. For this purpose, limiter A is placed directly across switch A. Thus, when switch A opens, the limiter takes all of the current but places a fixed voltage across the load and switch A, causing a linear

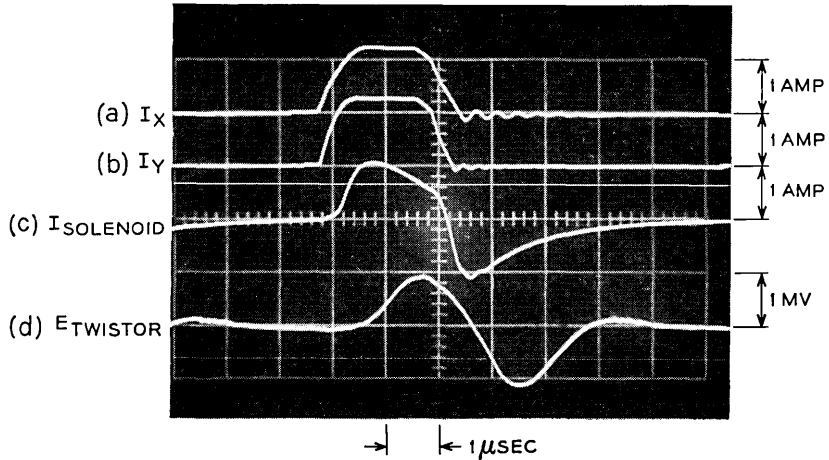


Fig. 8 — Access waveforms; (a) horizontal select current, 1 amp/division; (b) vertical select current, 1 amp/division; (c) drive solenoid current, 1 amp/division; (d) twistor output, 1 mv/division; horizontal scale, 1  $\mu\text{sec}$ /division.

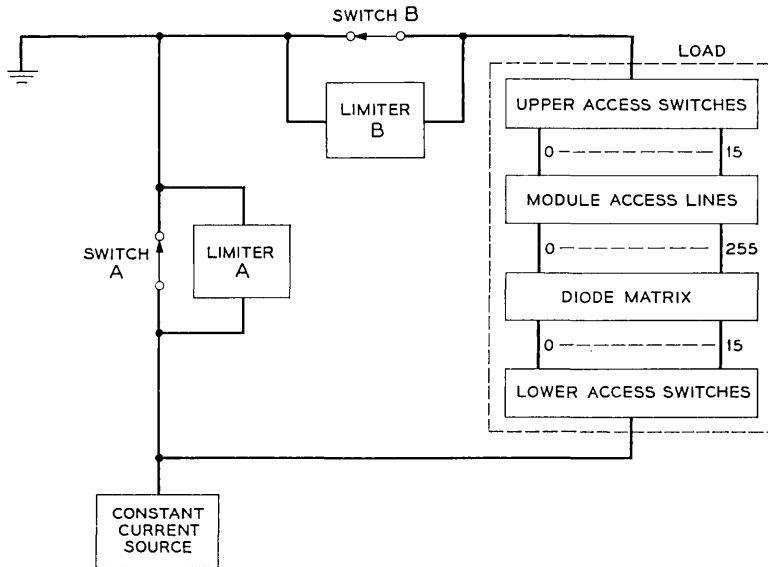


Fig. 9 — Basic access pulse generation.

buildup of current in the load. This continues until the load current equals the source current, at which time the load current remains fixed and the voltage on the parallel circuits drops to a low value determined primarily by voltage drops of the various semiconductor devices.

This mode of operation is uniquely suited to the twistor. In the twistor the inductances of the drive lines are all well matched and independent of the memory contents. Thus the application of a constant voltage results in accurate control of the rise time.

A similar method is used to effect turn-off. Again referring to Fig. 9, switch B is opened and switch A is reclosed. The inductance of the load must now drive limiter B. Thus a constant reverse voltage occurs across the load until the load current drops to zero. Switch B can then be reclosed and the access switches opened. With this method only switches A and B are used to make and break the current paths. The access switches are merely used to route the current pulse to the proper drive line.

### 2.2.2 Switches A and B

Switches A and B are identical circuits. They are both normally closed switches utilizing a pair of 20D transistors in the output stage and

operated in an antisaturation circuit to provide fast turn-off by elimination of the storage time inherent in saturated transistors. The basic circuit is shown in Fig. 10. The input stage is compatible with LLL and is driven from timing chain flip-flop circuits used as gate generators. The second stage is connected across the base-emitter terminals of the Darlington-connected output stages. This connection allows very fast turn-off even when driving inductive loads. The voltage transient that occurs as the switch is opened drives the second stage harder into saturation, preventing the power stages from turning on again.

The paralleling of the output stage is necessary in this circuit to keep the junction temperature below  $100^{\circ}\text{C}$  in the worst case, since these switches work in a nonsaturating mode and may run at 100 per cent duty factor.

### 2.2.3 Limiter

The basic limiter circuit is shown in Fig. 11. The limiter is a floating limit circuit. That is, both ends are free while the circuit is in a limiting mode and the capacitor,  $C_1$ , controls the limiting voltage. In its quiescent state, the  $-40\text{-v}$  regulator controls the voltage across  $C_1$ . The transistor used is a 20D with a resistor in its collector to lower the transistor power dissipation when the peak current of 1.35 amps is being limited. The limiter is expected to limit the voltage across its corresponding switch and carry a  $0.5\text{-}\mu\text{sec}$  triangular current pulse every  $5.5\ \mu\text{sec}$ . A pulse transformer is provided in the collector circuit of the limiter. It produces

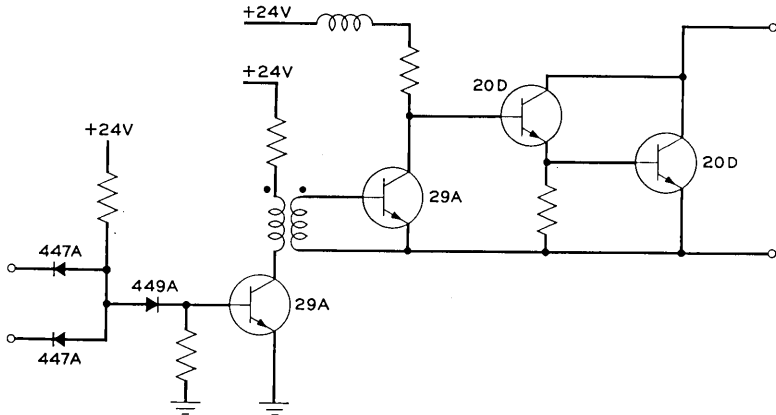


Fig. 10 — Circuit schematic — switches A and B.

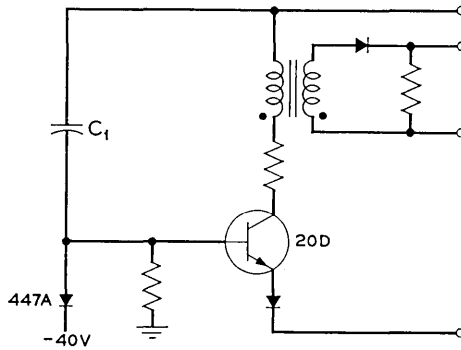


Fig. 11 — Circuit schematic — limiter.

a pulse used for maintenance checks on the access circuitry. The limiters for one axis are mounted on a common plug-in board.

#### 2.2.4 Access Switches, Voltage Pulser, and the Diode Matrix

The access switch is a normally open switch used to select a current path through the diode matrix. The basic circuit for the access switch is shown in Fig. 12. The input stage is basically an LLL stage with 5 inputs. Four of the inputs are from the address register and provide the 1-of-16 selections required. The fifth is a gating input that controls the closing and opening times of the selected access switch. The collector of the input stage contains a 3-to-1 stepdown pulse transformer to drive the base of the output stage. This transformer coupling allows the output stage to float relative to the input stage. The transformer is designed to cause a 20 per cent droop in the current pulse applied to the output stage. This causes a negative drive to the base of the output stage to improve its turn-off characteristics when the input stage is turned off. The connections of the upper access switches for one axis are shown in Fig. 13. The capacitors ( $C_s$ ) shown on the emitters of access switches represent the stray wiring capacitance of approximately  $3000 \mu\mu\text{f}$ . The capacitance is charged to negative battery voltage to back-bias the diode matrix. Thus, when an upper axis switch is selected, it must discharge the  $3000 \mu\mu\text{f}$  capacitance. To prevent this discharge current from exceeding the switch ratings, a filter is provided to limit the current surge to 1.35 amps at 50 volts. This allows the capacitance to be discharged in about  $0.25 \mu\text{sec}$ . For this reason, the upper access switches are turned on  $0.25 \mu\text{sec}$  before the lower access switches and switch A, which initiates the main drive pulse. Conversely, switch B is opened

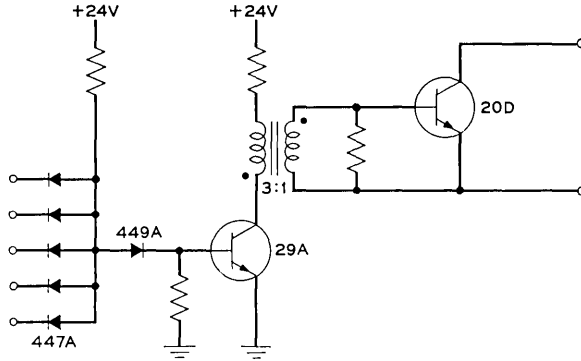


Fig. 12 — Circuit schematic — access switch.

0.25  $\mu$ sec before switch A is closed, which forces the selected path to recharge its capacitance back to the level set by limiter B. The capacitance is charged the rest of the way to battery voltage by the action of the voltage pulser. The voltage pulser is identical to switches A and B; it is opened just prior to the initiation of a drive pulse and closed after the access switches have all opened. The voltage pulser connects the emitter side of each upper axis switch (and its associated capacitance) through a diode to battery.

The voltage pulser contains, as do switches A and B, a pulse trans-

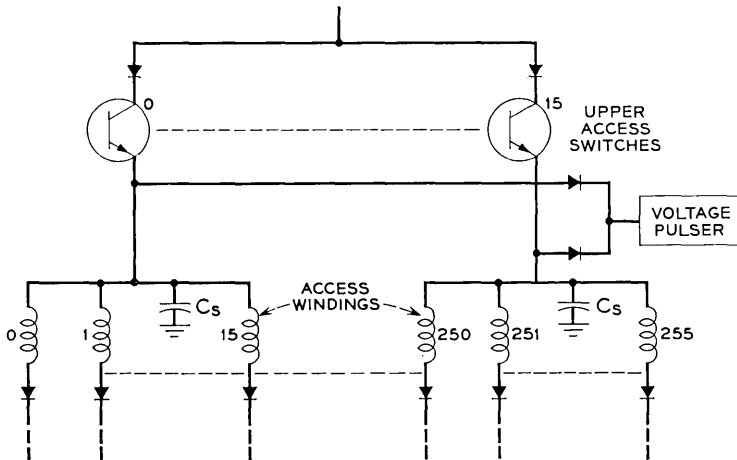


Fig. 13 — Partial schematic — access matrix showing stray capacitance load handling.



former to provide a voltage pulse representative of the switched current. This is used for diagnostic and maintenance purposes.

### 2.2.5 Bias and Drive Current Regulator

The bias and drive currents are regulated by identical plug-in current regulators. The currents at which the regulator operates, 1.31 amps for drive and 2.62 amps for bias, are determined by the socket interconnections. Because of the large amounts of power that must be handled, the regulator output stage is operated as a high-speed switch rather than in a linear mode. The average output current is controlled by adjusting the percentage dwell time of the switch closure. In this mode of operation the 20D transistors are either cut off or in saturation except for the short time required to transfer from one state to the other. Thus the power dissipation of the transistors is a small fraction of the power that is controlled.

The operation of the regulator can be seen from Fig. 14. The transistor switch and the resistor,  $R_s$ , are connected in parallel. This combination is then connected in series with the reference resistor  $R_R$  and the choke  $L$ . Feedback control is used to control the "on" and "off" times so as to generate the desired dc average current. (The resistance of  $R_s$  must be chosen large enough so that the highest steady-state voltage across it causes less than the desired regulator output current to flow into  $L$ .) Thus, the regulator's output current has an ac component; however, this component can be made quite small. The amplitude of the ripple component of the regulator's output current is controlled by the reference resistor value, the difference amplifier gain, and the Schmitt trigger's "window" or difference between "on" and "off" trigger voltage. The ripple frequency depends primarily on the above items plus the value of  $L_F$  but is also a function of the average battery and load voltage. The

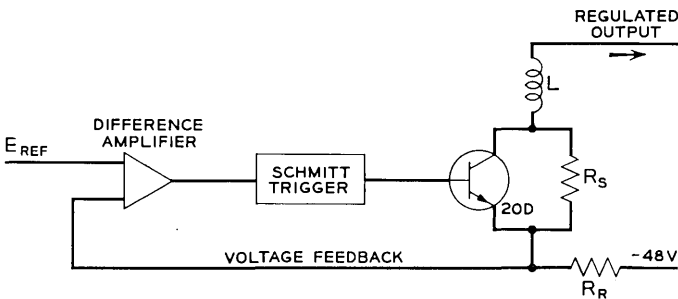


Fig. 14 — Access current regulator.

regulators operate at a nominal frequency of 50 kc with less than 1 per cent peak ripple. The choke serves to maintain constant current during transients too rapid for the regulator to handle directly.

Since the actual drive seen by a biased core switch is the sum of the  $X$  and  $Y$  drive currents minus the bias current, it is important that the bias and drive regulators do not drift in opposing directions and thus cause additive errors in the net drive. To prevent this occurrence, tracking circuits are provided that cause the output of each bias regulator to equal the sum of the  $X$  and  $Y$  drive regulator outputs within 1 per cent, ripple not included. The tracking circuit is composed of magnetic amplifiers driven with a 1-kc carrier current. A fail-safe tracking check detector is provided for diagnostic checking of the tracking circuitry.

### 2.3 Access Timing

The more important timing functions for the access portion of the program store are shown in Fig. 15. The various timing gates are generated by setting and resetting various flip-flops under the control of the access timing chain. Also included in Fig. 15 are some of the currents that are generated by these timing functions.

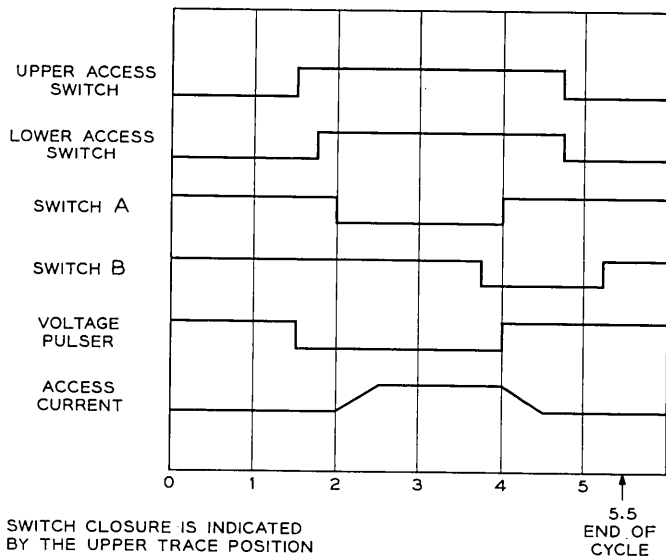


Fig. 15 — Access timing for one axis.

## III. STORE READOUT

3.1 *Signal Selection*

As previously mentioned, in each group of four modules corresponding readout wires are paralleled. This results in eight groups of forty-four wires, of which only one contains the desired information. The other groups contain unwanted information or noise which must be suppressed. The selection of the desired group takes place in the low level selectors (LLS).

The LLS (see Fig. 16) accepts two twistor pairs, both associated with the same bit of the words common to an access solenoid. The two inputs are amplified and one is selected. The selection is made as early in the store cycle as possible so that selection transients will not interfere with the readout signals. The design of the LLS is such as to avoid in-service gain adjustment.

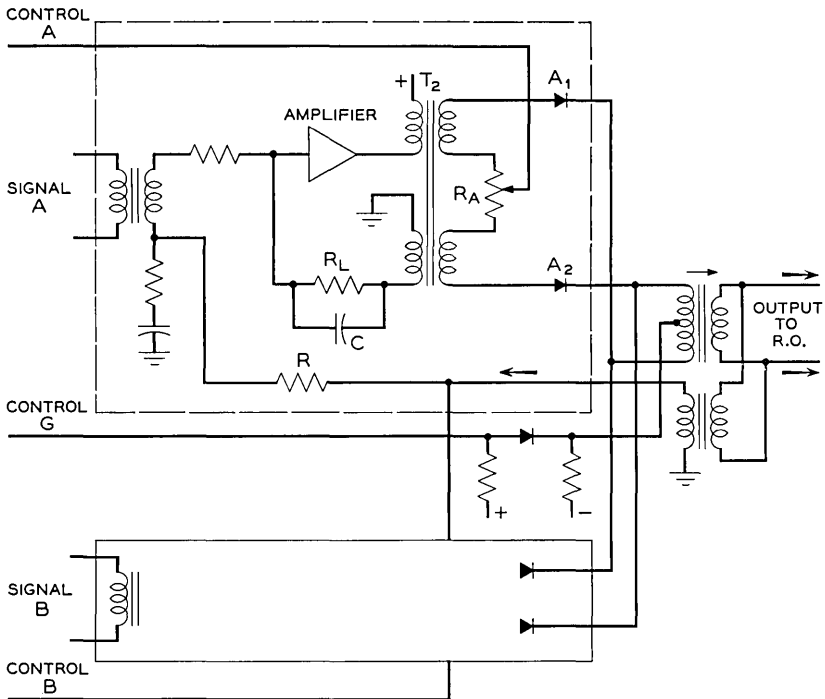


Fig. 16 — Circuit schematic — low-level selector.

The input signal is transformer-coupled to give good common mode rejection and to improve the impedance match. There are two stages of amplification with two feedback paths. There is a local feedback path,  $R_L C$ , and an over-all feedback path,  $R$ . The local path controls the high-end frequency response and gives the amplifier a low output impedance working into  $T_2$ . The over-all feedback path controls over-all gain and improves the low-frequency response.

Selection of signal A is made by making control A positive, control B ground, and G ground. With control A positive, equal currents flow through diodes  $A_1$  and  $A_2$ . To avoid the requirement for matched diode pairs,  $R_A$  is provided to equalize currents. Since the currents are equal, only a minimum transient occurs when a path is selected. The signal at diodes  $B_1$  and  $B_2$  is sufficiently small that the diode threshold blocks the signal B.

Over a store cycle, the signal from the twistor memory goes both positive and negative and averages to zero. To avoid generating a dc component in the LLS, the path selection cannot be changed until the complete signal has passed. To accomplish this, a separate buffer register holds the selection information from cycle to cycle.

The LLS has a very large amount of feedback. In addition, the feedback resistors are very precise and have excellent long-term stability and tracking. The combination assures gain stability of a few per cent over the life of the circuit.

### 3.2 *Signal Amplification and Sampling*

Each of the 44 readout amplifiers shown in Fig. 17 accepts the outputs from four low-level selectors and provides the necessary gain to operate the sampler. The four LLS outputs are mixed by four resistors,  $R_1$  to  $R_4$ , operating into the emitter of  $Q_1$  and the feedback resistor,  $R_F$ . The mixing point is a very low impedance compared to the 100 ohms of  $R_1$  to  $R_4$ . The readout amplifier has a large amount of feedback which provides frequency shaping and gain stability. Its long-term gain is constant to approximately one per cent. The signal at the output of the readout amplifier is bipolar and has a nominal amplitude of one-half volt for a one. A typical readout for several bits is shown in Fig. 18. Because of the wiring pattern, one-half of the ones have an initial positive loop, while the other half have an initial negative loop. Typical delta noise zeros shown are representative of a severe pattern interaction. The polarity of the zero is information-dependent and may be of either polarity at a given address.

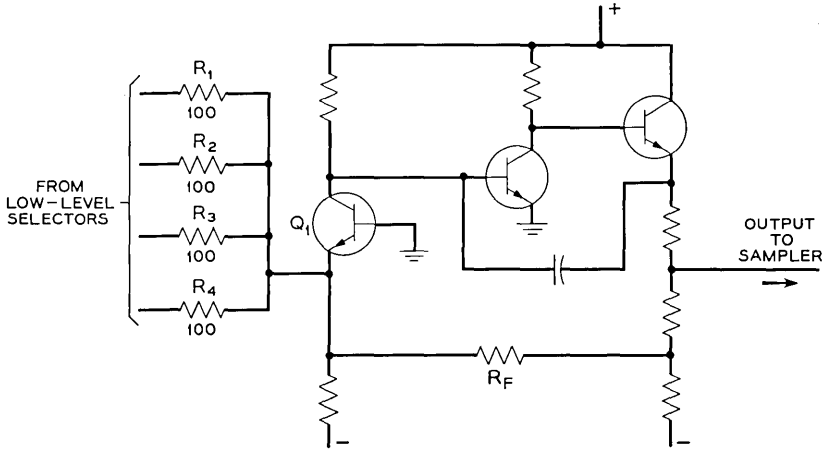


Fig. 17 — Circuit schematic — readout amplifier.

A sampler (see Fig. 19) must make the decision as to whether the output is a one or a zero and quantize the output.

In designing the sampler certain characteristics of the signal were exploited. One of these characteristics is that the delta noise passes through zero near the time the one signal reaches a peak. Advantage was taken of this characteristic by placing a very narrow strobe at the delta noise zero crossing, as shown in Fig. 18. Another characteristic of the signal is that at a given address the one is of a known polarity. The sampler is designed to observe only this polarity, so that delta noise of the opposite polarity, no matter how large, cannot generate an output.

The two-stage input amplifier provides gain and a very low output impedance. The low output impedance is necessary because the load is variable, depending on the size of the signal. There is no load on the amplifier until the strobe occurs and no load then unless the signal exceeds the threshold. The low output impedance of the amplifier makes it possible to deliver enough power to trigger the output if the signal only slightly exceeds the threshold. This leads to high stability. The low impedance also prevents modulation of the signal by the sample pulse.

The strobe pulse, correct for the polarity of "one" being sampled, is supplied to each sampler at the correct time. The amplitude of this pulse is not critical. If the signal exceeds the threshold reference, diode  $D_1$  is back-biased and the voltage change is coupled by  $C_1$  to the output transistor,  $Q_1$ . Since  $Q_1$  is biased in a Class A region, any minute voltage change is amplified and fed back to the base. This feedback path is

activated  $0.25 \mu\text{sec}$  before the narrow strobe pulse. The purpose of this feedback path is to guarantee full quantization of "one" outputs and to lengthen the pulse to the standard half microsecond used for bus communication. Once regeneration starts it can be terminated only by opening the feedback path. The feedback path cannot cause regeneration in the absence of an input pulse.

### 3.3 Strobe Timing and Generation

In the laboratory, optimum strobe time can easily be set for a particular set of circuits. It is more difficult to design circuitry which will produce the strobe at the correct time for any combination of access or readout circuitry. To accomplish this, the strobe timing must be referenced as closely as possible to the start of the access current flow. Fig. 20

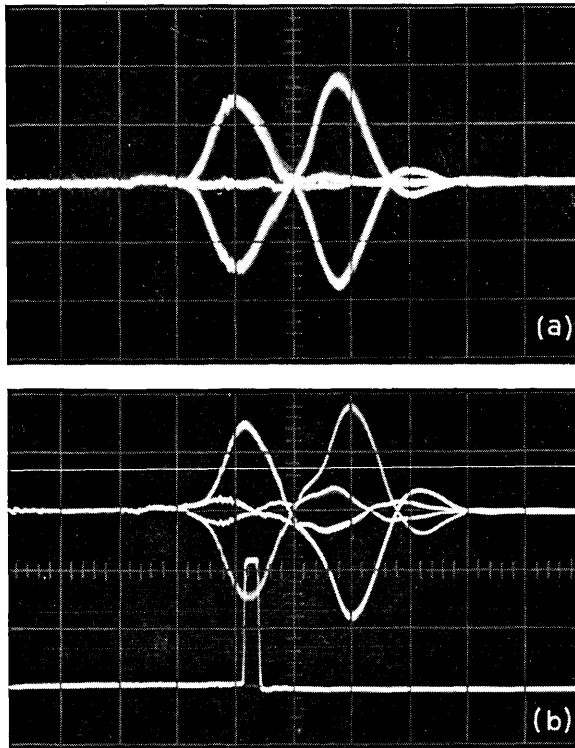


Fig. 18 — Readout waveforms: (a) typical readout; (b) readout with high "delta" noise.

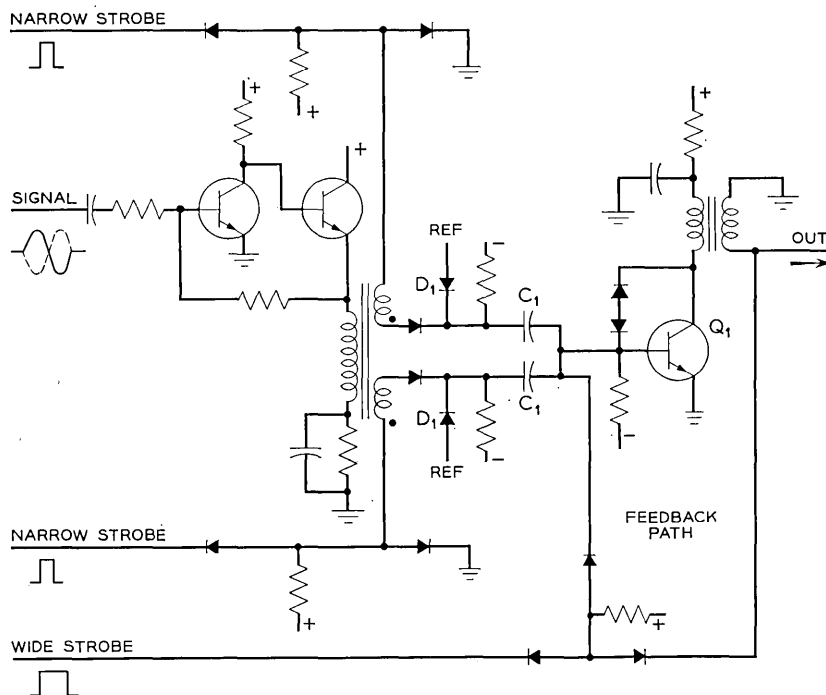


Fig. 19 — Circuit schematic — sampler.

shows the strobe generating circuitry. Three pulses must be generated: a narrow strobe of  $0.25 \mu\text{sec}$  duration, a wide strobe that starts  $0.25 \mu\text{sec}$  before the narrow strobe and ends  $0.25 \mu\text{sec}$  after the end of the narrow strobe, and a  $0.5\text{-}\mu\text{sec}$  sync pulse starting with the narrow strobe.

The readout signal occurs at a fixed time after the start of the access current. The beginning of access current for both axes is monitored by the strobe generator; only when both X and Y currents have started is the strobe circuitry initiated. This avoids the delay time variability of the access circuitry. The initiating pulse is delayed the proper amounts to set flip-flops 2 and 3, which initiate the narrow and wide strobes respectively.

The strobe amplifiers are relatively powerful, since they must drive 44 samplers. The strobe amplifiers each have an auxiliary output; these are added logically to start the timing of the wide strobe termination. This approach assures that the width of the pulse output of the sampler is  $0.5 \mu\text{sec}$ .

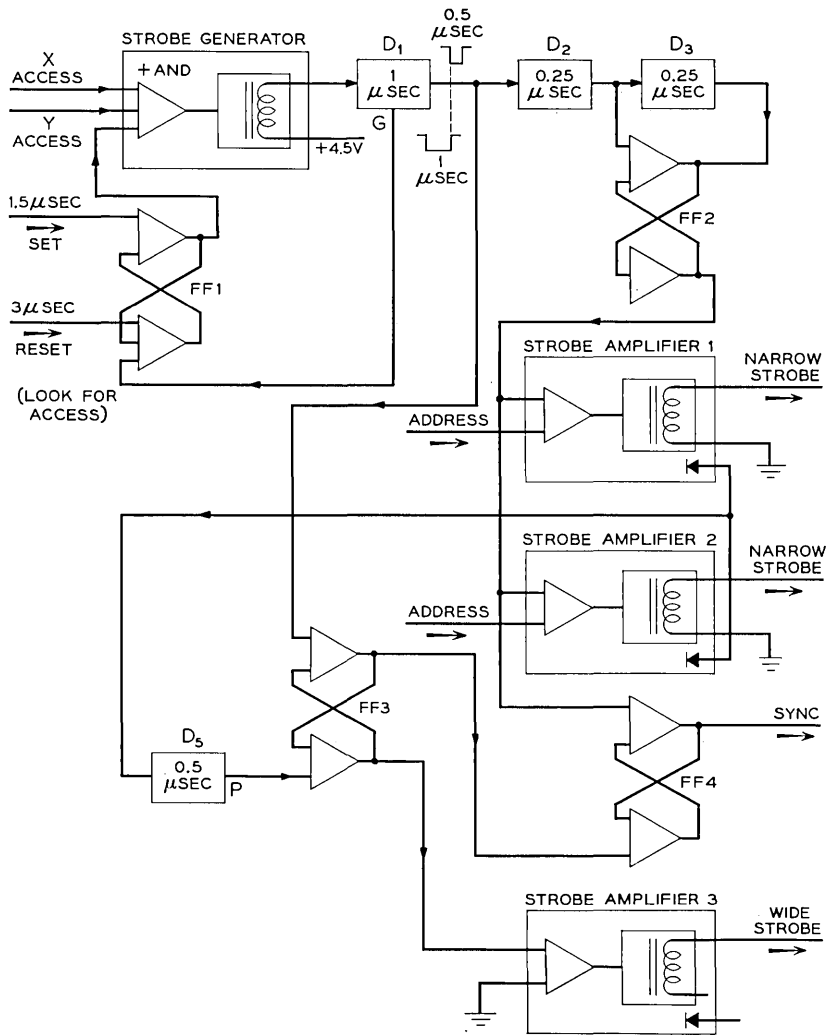


Fig. 20 — Strobe generation.



#### IV. CONTROL

##### 4.1 *Communications*

The principal communication path of a program store is a duplicated two-way bus system connecting all program stores and both central controls. These buses carry the necessary addressing and control information required to read a particular word from a particular store or pair of stores, or perform a control function on a particular store. The buses also carry the reply from the program store containing either the desired stored word or, in special control modes, the states of various flip-flops. The signals are sent as 0.5- $\mu$ sec pulses on a balanced line for binary ones and no pulse for zeros. The busing arrangement is described in more detail elsewhere in this issue.<sup>8</sup>

The program store also receives inputs from the central pulse distributor, the signal distributor, and the master control center and sends outputs to the master scanner. The inputs from the central pulse distribution are direct connections to each store, as opposed to a bus, and consist of either positive or negative 0.5- $\mu$ sec pulses. These pulses must be accompanied by a pulse on a sync bus to be effective. They are used to set or reset various control flip-flops in the program store.

The inputs from the signal distributors are relay contact closures used to control special relay circuits or lights in a particular store. The inputs from the master control center are also dc contact closures. These inputs provide manual override control of the program store and can be used to force a particular store or stores off-line or to force a store to listen on a particular bus from central control.

The outputs from the program stores to the master scanners are divided into two groups. One group consists of direct wires from each store to a scanner ferrod and is used primarily to detect the status of various control relays and flip-flops in the store when they cannot be interrogated over the bus system. The second group is another common bus system to which any program store may be connected. This 48-pair monitor bus terminates in a bank of ferrods in the master scanner. When so ordered by the signal distributors, a program store will connect a group of test points to the monitor bus. This operation is used for maintenance checks of the voltage and current regulators and other dc levels throughout the store.

##### 4.2 *Codes and Modes*

Each program store has its memory field separated into H and G halves. Each program store half will have a numerical code wired into

the store at the time of installation. This allows an office to operate with an odd number of stores and yet provides full duplication, as shown in Fig. 21. As can be seen, duplicate information (identified by the same numerical code) is always provided in the logically adjacent store. That is, information field 2 in the H side of store 2 is duplicated in the G side of store 1, and so forth. Normally, all H-half outputs will communicate on one bus to central control and the G-half outputs on the duplicate bus. Thus, when a particular information word is requested, the two stores containing the word will interrogate the required memory location and answer back on separate buses. If one of the program stores should fail, then the store carrying the duplicate information would be arranged to answer on both buses to provide duplication. The decision as to which bus or buses to answer on is made by system programs and sent to program stores as control orders.

Along with the address and code bits sent by the central controls on the program store buses are several mode bits which establish the type of operation desired of a store. Four modes of operation are decoded from these bits by the program stores. They are normal, maintenance H, maintenance G and control. If a normal mode is recognized, the stores (or store) that also recognize the code bits will send back the desired memory word. If a maintenance mode H is requested, only the store that recognizes the code bits as being the code of its H half will answer. Similarly, for a G mode, only the store that recognizes the code bits as being the code of its G half will answer. The control mode order affects only the store that recognizes the code bits as being the code of its H half. The control mode is not a memory read operation. The control mode is subdivided into two submodes, the control write and the control read. This selection is made by an additional wire pair in the bus system. In the control write mode, the address leads are used to write, dual rail, into various flip-flops within the store. Some of the address leads select which bank of flip-flops will be written; others contain the information to be written. In control read, the states of various flip-flops throughout the

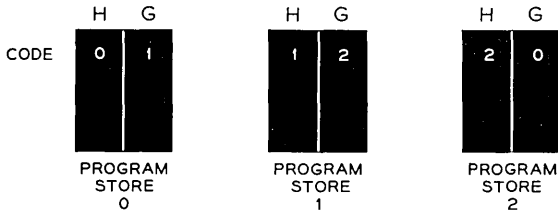


Fig. 21 — Program store duplication.

store are sent back to the central controls. The address bits are used again to select which bank of flip-flops is to be interrogated. The control mode constitutes a powerful maintenance and diagnostic tool by providing a means of writing into and reading out of most of the control and maintenance flip-flops of the program stores.

#### 4.3 *State Flip-Flops*

In order to control the store communications, several basic state flip-flops are provided. One group of such flip-flops controls the store routing. For example, one of the flip-flops selects the input bus on which the store should receive address and control information. This flip-flop is normally controlled by the central pulse distributor, but may be controlled by the master control center. Four flip-flops are used to control the normal store output to the buses. The H and G halves are each controlled by two of these flip-flops, which are normally set and reset by a control write operation.

In addition to the foregoing, two trouble flip-flops are provided that can prohibit the store from either sending or receiving on either or both of the buses. These can be set by the store itself, the central pulse distributor, or the master control center.

In addition, the office contains an emergency-action circuit which can set the state flip-flops. This circuit is activated whenever the system stops, and sets one of the stores containing program information to communicate on one of the buses and the store containing duplicate program information to communicate on the other bus. In offices with more than two stores, the remaining stores are disconnected from the buses. Following similar action with other units, the circuit attempts to restart the system.

#### 4.4 *Timing*

The store timing is initiated by a sync pulse on the input bus from central control. The pulse is then delayed in each of two independent timing chains. One chain is used primarily for access timing, while the other is primarily used for control circuit timing. The two timing chains provide reliability in that critical communication control circuits are wired to both chains so that a failure of one timing chain does not cause a breakdown of communications and thus allows diagnostic programs to use the control mode feature to check the timing chains. Having two chains also makes it possible to check the accuracy of one timing chain against the other. The timing chains consist of a series string of delay gate

generators (DGG). The DGG's are available with delay times ranging from 0.25  $\mu\text{sec}$  to 1800  $\mu\text{sec}$ , all of which use the same printed wiring board but have a different timing capacitor. The DGG consists of two monostable multivibrator circuits connected as shown in Fig. 22. The first three transistors comprise one monostable circuit with a negative input signal used to trigger the circuit. This circuit generates an accurate gate pulse whose width is equal to the desired delay. The output of this circuit is applied to the final transistor, which turns on at the end of the gate pulse for approximately 0.5  $\mu\text{sec}$ . Thus the output of a DGG is a negative-going pulse with a width of 0.5  $\mu\text{sec}$  delayed by a fixed amount from the input negative pulse. The DGG has several diodes for temperature compensation and can be set to better the 0.1 per cent. The long term drift of the DGG is expected to be less than  $\pm 5$  per cent of the delay value. A DGG can drive up to five input loads which may either be LLL circuits or other DGG's.

The DGG's on the access timing chain are used primarily to set and reset gating flip-flops which are in turn used to control access timing gates such as required for the access switches or switch A.

The timing chains are checked against each other at several places along the chains, including the ends, to ensure agreement in the two chains. In addition, circuits are provided to check the output level of each DGG to ensure that it is normally in its high state and not loaded

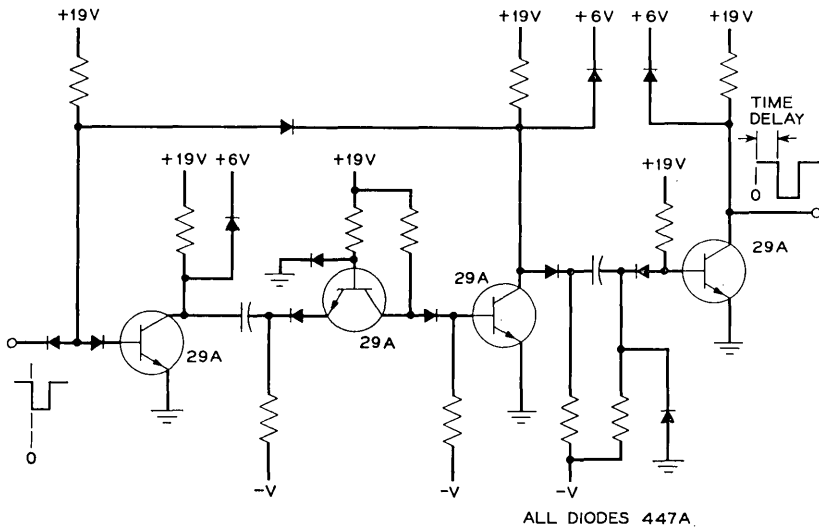


Fig. 22 — Circuit schematic — delay gate generator.

down by some other package with an input problem. Also, the operation of either timing chain can be inhibited to check the check circuits.

#### 4.5 Power

The program store obtains its power from the +24-v and -48-v office batteries. This power is filtered and switched at the store. Separate power is provided at the store to power the card loader when the latter is connected to a store.

The program store draws essentially a constant load of almost 1000 watts from the battery plant. The actual battery drains are 18 amps on the +24-v bus and 12 amps on the -48-v bus. The power is filtered by a double L-section filter at the store input terminals both on the +24-v bus and the -48-v bus. The filters serve both to prevent noise spikes from entering the store and prevent power turn-on and -off transients from interfering with other operating units. The filter is so damped that the turn-on current increases smoothly to its operating value with no overshoot.

Several voltage regulators are used in the program store. A 4.5-volt regulator is used to supply the LLL stages as well as the low level selectors. Also +19- and -40-volt regulators are provided to supply several special circuit packs.

The +4.5-volt regulator is a high-efficiency "off-on" regulator which can deliver 1.5 amps at 4.5 volts while drawing about 0.4 amp from the +24-v bus. A basic diagram of the regulator is shown in Fig. 23. The output stage consists of a choke input filter which is either being charged by the 20D transistor connected to +24 volts or is being discharged by being forced to draw current from ground through the diodes. The "off-on" operation of the 20D is controlled by the output of the Schmitt

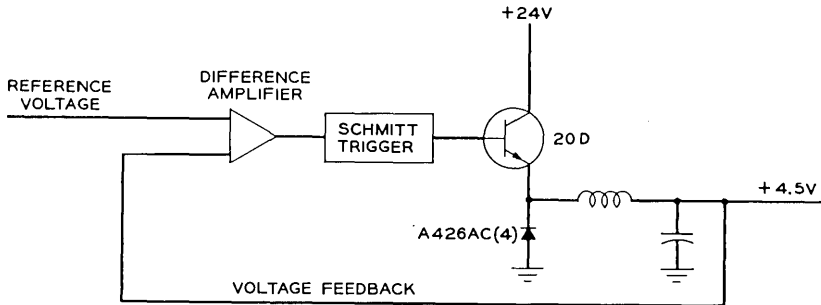


Fig. 23 — Circuit schematic — 4.5-volt regulator.

trigger, whose input is an amplification of the difference between a Zener reference stage and the regulators' output voltage. The feedback is so controlled that regulator "off-on" frequency is about 50 kc, with the 20D transistor being on about 20 per cent of the time. This type of regulator has the advantage of providing good regulation with high efficiency and low power dissipation in the semiconductors. The operation of this circuit is similar to the operation of the bias and drive current regulators described in Section 2.2.5.

The +19- and -40-volt regulators are conventional series linear control regulators. Since many of these regulators are used in each store, special reference regulators operating at +17 and -38 volts were made for the +19- and -40-volt regulators respectively. This circumvents the need of providing a Zener reference for every regulator package. Actually, two reference units for +17 and -38 volts are provided to facilitate maintenance and diagnostic checks.

The voltage regulators are checked for the correct voltages by means of the scanner monitor bus. The general method of testing the +19-volt regulators is indicated in Fig. 24. Since the regulators are cross checked against another regulator and reference unit, the failure pattern is unique for each regulator. To ensure that the check circuits are operating properly, the 19-volt check regulator outputs can be shifted by operating

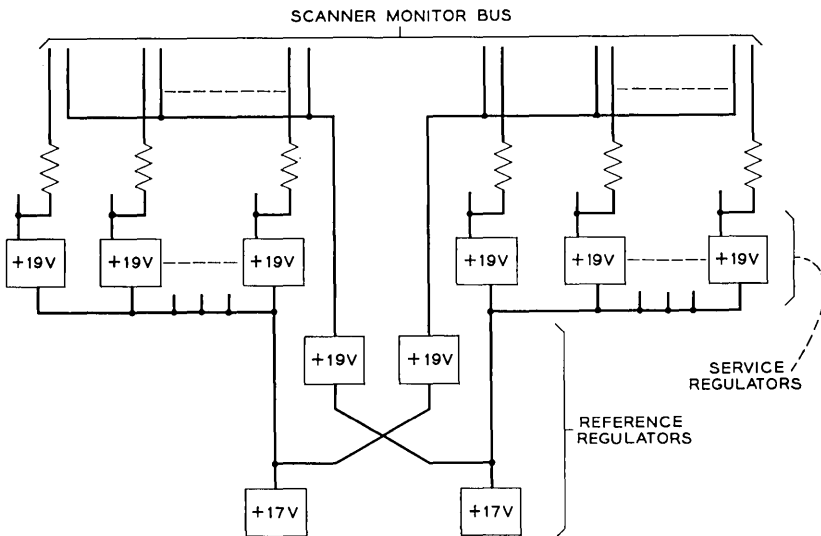


Fig. 24 — Regulator diagnostic checking.

a signal distributor input to give trouble indications of all associated regulators. The  $-40$ -volt regulators are tested in the same manner as the  $+19$ -volt units. The  $+4.5$ -volt regulators are checked two against one on the scanner bus, since only three are required per store.

## V. MAINTENANCE

### 5.1 *Maintenance Philosophy*

The program store maintenance and diagnostic check circuits were designed along with the operational circuits. Each circuit, including the maintenance circuits, became the subject of a detailed study to ensure that a fault could be readily localized to, in most cases, a single circuit package, and in the worst case no more than three circuit packages. The effect of the failure of any semiconductor device used in the store was scrutinized to ascertain that the diagnostic checks provided could localize the fault. In general, the maintenance and diagnostic system can be divided into three sections. The first section is the all-seems-well (ASW) section. An ASW pulse is returned along with every readout from the program store to central control and indicates the success of all internal store checks. The second section makes use of the control read and write modes to diagnose and test the stores' condition. The third section is composed of the direct scan point that checks various states within the store and the monitor bus, which can be used to check many points within the store, as previously discussed.

### 5.2 *All-Seems-Well Circuit*

The ASW circuit provides an extremely powerful check on a store's condition. The ASW pulse indicates that: (1) the mode decoder has received a valid code, (2) the timing chains are both in step, (3) the address register and other control flip-flops were reset at the beginning of the cycle, (4) one and only one  $X$  and  $Y$  access drive winding was pulsed (for modes other than control), (5) the access pulses were of the proper amplitude, and (6) only one readout group was selected. Thus the ASW signal is derived from the status of several individual circuits. The failure of the central control to receive an ASW signal from a program store will generate a system attempt to reread the desired word. If the failure continues, the store will be checked by a diagnostic routine to isolate the faulty unit.

### 5.3 *Read Control and Write Control*

The control modes together with a circuit known as "freeze reset" comprise a powerful diagnostic tool for checking the program store.

When a control read (CR) or a control write (CW) is recognized by a program store, it knows the system wishes either to interrogate the state of a set of store flip-flops or to write into a set of flip-flops rather than to perform a memory operation. For example, the system can, in CW, write into a flip-flop that normally is set when an access failure occurs. This should result in an ASW signal failure if the ASW circuitry is functioning properly. The pulse detectors that check the amplitude of an access pulse can be forced to indicate either high or low as desired to check that these circuits are functional.

A CW command can also be used to place the freeze reset circuits into operation. In this type of operation, many of the flip-flops which are normally reset at the end of each system cycle (5.5  $\mu$ sec) are blocked from being reset. Thus the address register, for example, can be filled and its contents checked by a CR, an operation which could not otherwise be checked. Also, the freeze reset operation can be used to localize the cause of an ASW failure. The store is placed in a freeze reset condition by a CW; then a maintenance operation is called for. This latter operation will cause the memory to be interrogated. If the ASW indicates a failure on this operation, a flip-flop for the defective circuit area will have been set and will not reset at the end of the cycle. Thus this flip-flop can be read with a CR command to localize the fault. The freeze reset condition is restored to normal by a CW command.

To localize a fault to a specific package it is often necessary to run a special sequence of instructions and determine the pattern of ASW failure indications. For example, an open diode in the group select flip-flops could cause two readout groups to be selected at the same time. The ASW circuit will indicate the program store failure, the control mode and maintenance mode operation will determine what section of the store is causing the failure (in this case the group select flip-flops), and then a simple four-step pattern will determine which of the four group selects are in trouble, since the ASW signal will indicate a failure except when the faulty package is selected.

### 5.4 *Direct Scan Points and the Monitor Bus*

The slowest maintenance access, but the most direct, is the use of scan points.

Each store has 17 direct scan points, most of which are used to deter-



mine the state of the more important control flip-flops such as the code, trouble and bus receive flip-flops.

The direct scan points are also used to check the store manual control pushbuttons and power alarms as well as checking the states of relays controlled by the signal distributor, as discussed later. It is necessary to have direct scan points on functions such as the code detectors, since the store will not answer on the main communication path if it cannot recognize its codes, making the previously discussed diagnostic checks useless.

The scanner monitor bus is a 48-wire pair bus common to all the program stores within an office. The bus can be connected to points within a store by operating banks of relays under the control of the signal distributor. Each store has two banks of relays used to test the bus and connections to it, and three banks of relays to test various points within the store.

All of the store voltage and current regulators are checked by means of the monitor bus. The monitor bus is also used to check many flip-flops to aid in trouble diagnosis. In particular, the monitor bus is used to check the many packages whose failure could cause the store to become inoperative. The failure of a store to generate an access pulse in one axis could be caused by several different packages, most of which cannot be checked by a control read operation. The scanner monitor bus provides an inexpensive method of checking these packages.

## VI. STORE EQUIPMENT<sup>6</sup>

The functional arrangement of the program store equipment is suggested by the flow chart, Fig. 25. Here the principal routes of information flow are superimposed on the equipment layout of the store. The address input signals are received by the pulse transformers in centrally located terminal strips at the top. These signals are then routed via the address register and timer unit to the  $X$  and  $Y$  drive circuits at the left and to the readout selectors at the right. The 256-point  $Y$ -select matrix reaches the select windings of the 16 memory modules with a vertical cable at the left end, while signals from the  $X$ -select matrix are distributed to the vertical select windings by means of a horizontal cable above the memory modules. These cables are shown in the rear view of the store in Fig. 26, together with the short horizontal jumpers which multiple the  $Y$ -select windings of the individual memory units and vertical jumpers which multiple the  $X$ -select windings. Thus, the desired  $256 \times 256$  coordinate selection is achieved through short, direct wiring in a rectangular array.

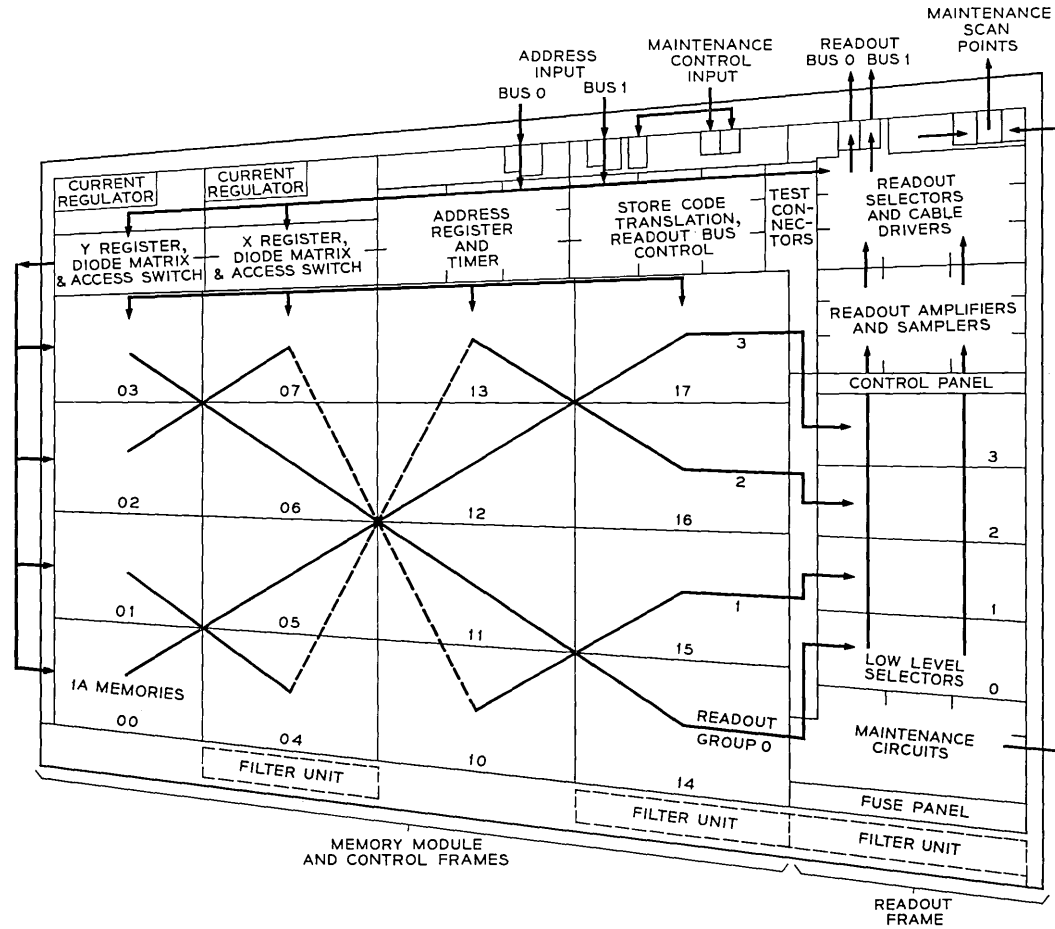


Fig. 25 — Program store flow chart.

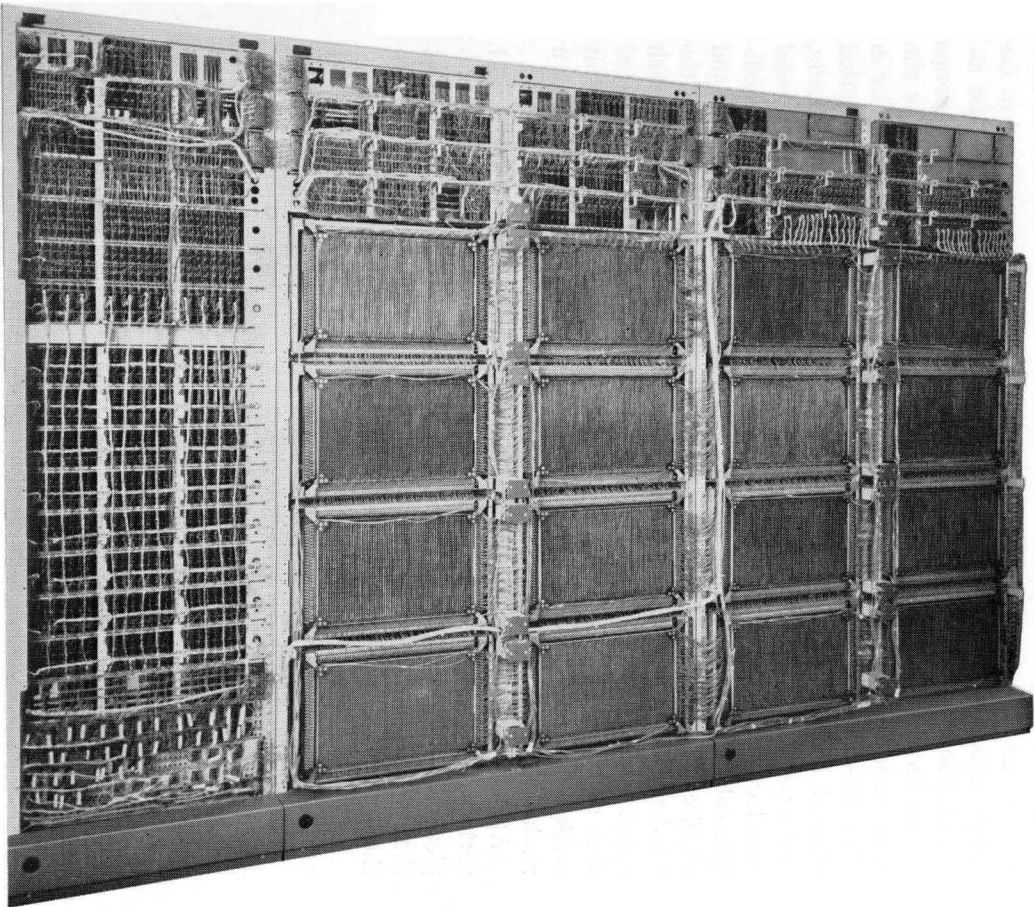


Fig. 26 — Program store — rear view with shield removed from readout bay.

For readout, the memory modules are arranged in four groups of four modules each. The twistor tapes for the four modules within each group are multiplied in parallel. To minimize noise associated with half-selected addresses, no two memory units of a group are located in the same horizontal row or vertical column. The memory modules are associated as shown in Fig. 25, with readout connections to the four groups of low-level selectors at the right. These connections, as well as the interconnections between modules, are made with 26-gauge BY wire in close-twisted pairs ( $\frac{1}{2}$ -inch lay). These wires are precabled and routed through horizontal and vertical metal ducts between the memory units, and a vertical duct between the units and the low-level selectors. Special care has been exercised in the design of these cables and ducts to facilitate cable replacement, if necessary, in the event a memory unit should require replacement. The ends of the cables which terminate on the low-level selectors are distributed in the eight horizontal branches which appear in the left-hand bay of Fig. 26. The two branches associated with each readout group reach 44 circuit packs, which receive the 44-bit readout signals from each of the two twistor tapes associated with a readout group.

After selection, the readout signals are carried upward on the 11 vertical cables (four bits per cable) to the readout amplifiers and samplers, to the cable drivers and thence to the readout terminal strips at the top. The quality of the readout is due in considerable measure to the orderly and functional organization of this equipment.

## VII. CARD WRITING

### 7.1 *Objectives*

The card writer is provided to rewrite, as necessary, magnet cards containing translation information for the office. It must be economical and reliable. It must not burden the real-time capabilities of the office while writing cards or require a high degree of operator skill. The procedure followed in writing cards should not destroy an existing module of cards until the new set is written and verified. Because there is no requirement of 100 per cent up time, duplication of communications, power, and equipment is not required.

The card writer must keep up with the writing requirements of a large office. Such an office, containing 6 stores, may require that  $4\frac{1}{2}$  stores be rewritten each week. The card writer is designed to accomplish this in less than 18 hours of machine time.

The objective at the outset was to write the cards so well that no store margins were deteriorated due to marginal cards.

Every effort was made to design the writing process in such a way that no preprocessing such as bulk magnetization or demagnetization is needed and that the card will be correctly written regardless of the prior state. Accomplishment of this objective permits single-word writing, which is useful during program debugging.

### 7.2 *System Requirements*

The card writer magnetizes or demagnetizes the bits on a card by passing a head across the card. The system must provide the information to the head at the proper times. This requires suitable communication between the card writer and the system. The time between successive words is 13 milliseconds. This is slow enough that a scan point can be set by the card writer as soon as a word is written and the system can pick up the scan point on a routine scan and deliver a 44-bit word before the head reaches the next word.

Between each card writing operation approximately two seconds elapse while the system assembles the information for the next card. There is no check on the information written until the cards are inserted in the store. Every effort is made to assure a well written set and to complete the writing even when the system has difficulty delivering words on schedule. Some troubles, such as failure to receive a word when requested, cause the card writer to repeat the card. If three tries fail to write a card successfully, the process is aborted and an alarm sounded. Other troubles cause the process to be immediately aborted. In general, any indication of an incorrectly written module leads to termination during the card writing process. This follows the general philosophy of not putting suspect cards into operating stores even for checking purposes.

### 7.3 *Equipment*

The memory card writer, shown in Fig. 27 with a card loader attached, contains a centrally located card writing mechanism with circuitry above and below. The logic unit above is functionally arranged with terminal strips and sequence control circuits at the top and with registers and write control circuits between these and the control panel. A group of relays to control polarity of the writing heads and the writing head connectors are mounted just below the control panel on the sub-frame of the card writing unit. The lower part of this subframe supports a group

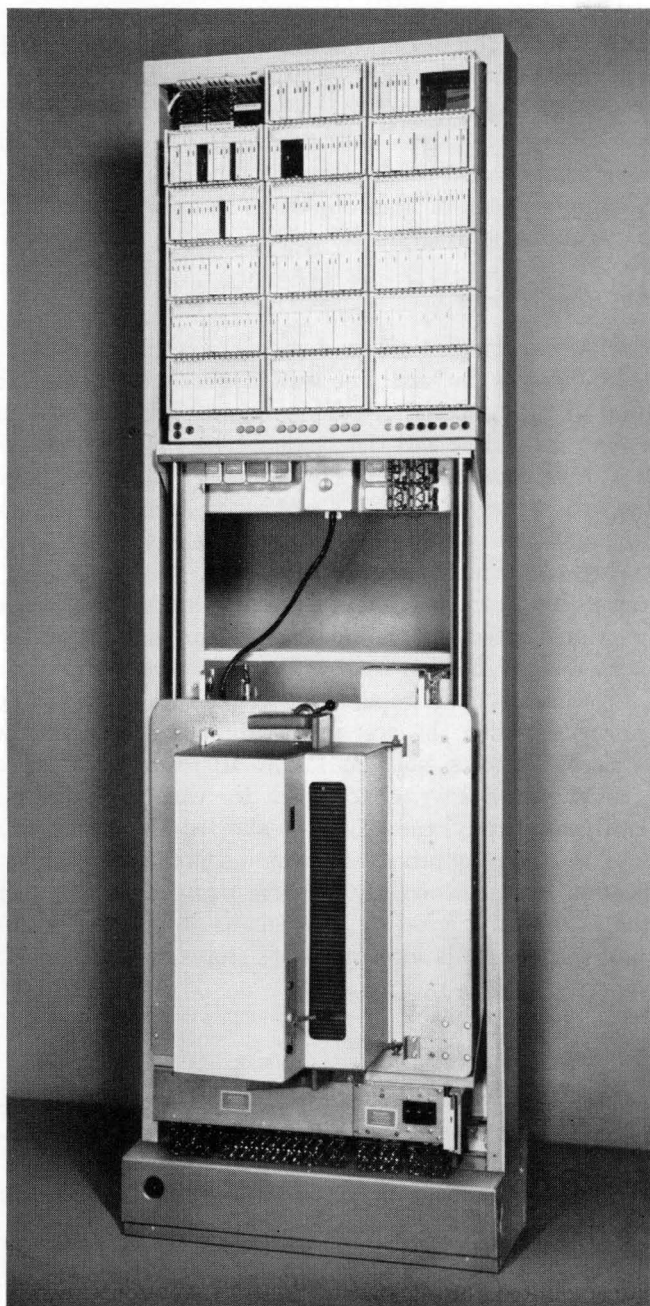


Fig. 27 — Memory card writer — front view with card loader attached.

of relays which control the mechanical sequences of the writing unit and power control circuits.

When a card loader contains a set of cards which are to be written, it is mounted on the memory card writer by clamping it in a vertical position to a carriage which forms a part of the card writing unit. This carriage is mounted on vertical rails at the sides which permit it and the attached card loader, to be indexed upward as successive cards are written. In Fig. 27 the carriage and loader are in a partially raised position.

At each index position, a pair of fingers on the card writing unit reaches forward, seizes the card in that location and draws it backward over a horizontal writing table to a fixed stop position. The card writing head then passes in a smooth, continuous motion across the card (Fig. 28). In this figure, a guard rail which prevents accidental interference between the fingers and the writing head has been removed for better visibility.

During its travel, which requires approximately one second, the head magnetizes the initializing magnets, senses the position of these along the length of the card, and magnetizes or demagnetizes each of the 2880 bit magnets. After passing the length of the card in a right-to-left direction (viewed as in Fig. 28), the head is returned to its normal position at the right, the fingers push the card back into the loader, the fingers disengage the card and return to an intermediate position, the loader indexes upward, and the cycle is repeated.

In the loader, the magnet sides of the cards face alternately downward and upward, corresponding to the alternately right and left directions which these cards face in the program store. Since only upward facing cards can be written, alternate cards are first written with the loader oriented as shown in Fig. 27. This sequence is called "pass A." When completed, the carriage and loader are automatically returned to the downward position and a buzzer sounds to alert the operator that pass A is complete. The loader is then inverted, end for end, reclamped to the carriage, and pass B is started, during which the remaining cards are written. The carriage is again automatically returned to the normal start position and the buzzer signals the operator that writing is complete.

The card writing unit and card writing head require moderately high precision in mechanical construction for satisfactory card writing. Sensing the initializing magnets to synchronize the writing function is done primarily to ease the mechanical tolerances for positioning the card in the longitudinal direction. The design incorporates fail-safe and interlock features to avoid or limit damage in the event of power failure or com-

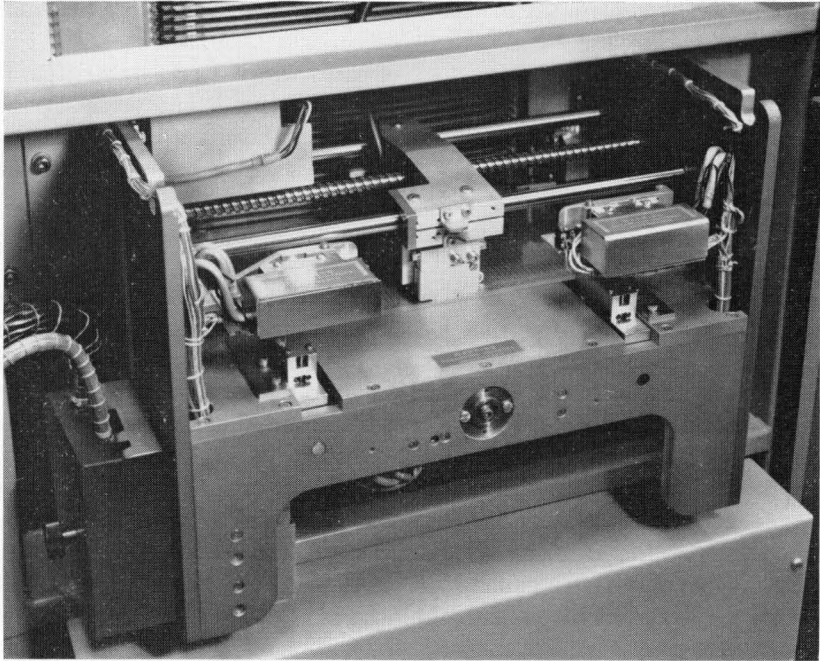


Fig. 28 — Memory card writer — rear view of writing mechanism.

ponent failure. The writing head and finger assemblies are designed for easy replacement in the field since they are more vulnerable to damage than some other parts because of repeated contact with the twistor cards. Also, the central mechanism portion of the writing unit, together with the writing head, have all electrical connections on a plug-in basis and can be removed from the frame for servicing or replacement by disconnecting two connector assemblies and removing four screws.

The card loader (see Fig. 29) is used to insert the twistor cards into and to withdraw them from the memory modules in the program store, to support the cards on the memory card writer during the writing operation, and to transport cards between the program store and the memory card writer. At the program store, the loader always inserts or withdraws all 128 cards of a set simultaneously, using a motor drive mechanism. The loader is controlled by means of three pushbuttons and a lever at the right, which is used to engage or release cards from the drive mechanism. In addition, an adjustable indicator assists the operator in keeping track of the particular store and module number with which he is working. Each memory module carries preadjusted brackets and



registration details which support the loader in accurate relationship with respect to the twistor planes in the memory. Moderately high precision is needed in control dimensions for the loader, memory module, and twistor cards for the loading and unloading operations to be carried out satisfactorily.

#### 7.4 *Circuitry*

Successful card writing depends on accurate positioning information, which must be obtained from the initializing magnets. A head precedes the sensing head to magnetize these magnets. The initializing magnets are very long (130 mils) compared to the desired positioning accuracy ( $\pm 5$  mils). The position sensing head has a special shape which develops a large pulse just as the head is centered over a magnet. Fig. 30 shows the sensing head and the relationship of the two gaps to the ends of the magnet. Only when the head is centered does the flux change through the two coils at the same time in an aiding sense.

Since there is a double row of initializing magnets, two heads are used alternately. The voltage from each head is amplified and sensed in an



Fig. 29 — Card loader in use on program store.

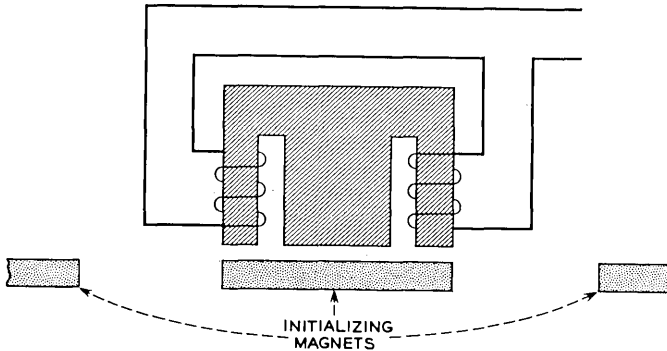


Fig. 30 — Position sensing head.

amplitude discriminator. To avoid false triggering, particularly during writing, the discriminators are enabled shortly before the proper head is at the expected position and are disabled as soon as writing is initiated.

Writing begins just as the write heads are approaching the center of a bit magnet. A relatively high frequency is used to magnetize or demagnetize the magnet. The basic waveform generated is a 20-kc sine wave which initially drives the heads into saturation and is gradually reduced in amplitude. The heads saturate at about 2 amperes of current at 26 volts peak amplitude. If it is desired to magnetize a bit, the waveform is altered during a saturated cycle just as the current crosses zero. The next current maximum is enhanced, and the current is forced to remain in the direction of that maximum. If the bit is to be demagnetized, the waveform is left on until the head is well off the magnet. The 20-kc waveform, Fig. 31, is developed by the circuit of Fig. 32. The head and two capacitors,  $C$ , form a resonant circuit which is in series with a Zener diode-capacitor network. The transistors are triggered "on" alternately for about  $6 \mu\text{sec}$ , one each half cycle. The waveform reaches a peak in two to three cycles, at which time the head saturates. At this time the waveform is altered if the bit is to be magnetized, or made a zero.

To completely erase a bit magnet is difficult. Any biasing, either of unbalanced current through the head, or flux from an adjacent head, leaves a residue of flux in the magnet. Even a few milliamperes of unbalanced current can upset the erasure. It is difficult to obtain this degree of balance when the peak currents through the head are amperes. The Zener diode-capacitor network prevents slight differences in average voltages on the two sides of the head from causing unbalanced currents to flow but allows dc to flow through the head while magnetizing. During

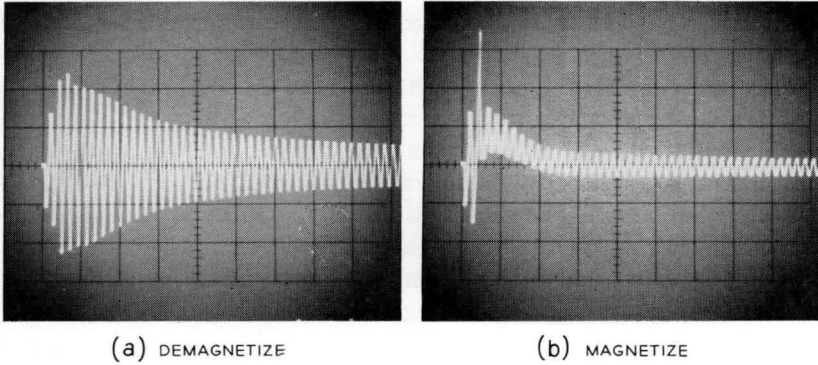


Fig. 31 — Basic demagnetize and magnetize waveforms.

the time the head is in saturation, exact balance is not maintained. Balance is recovered once the head is out of saturation, which occurs while the head is still over the bit magnet.

Biasing due to magnetizing adjacent magnets is avoided by essentially terminating the magnetizing currents during the early part of the demagnetizing period.

Additional circuitry is used for controlling the writing and mechanical sequencing of the unit. There are LLL circuits to check the count and

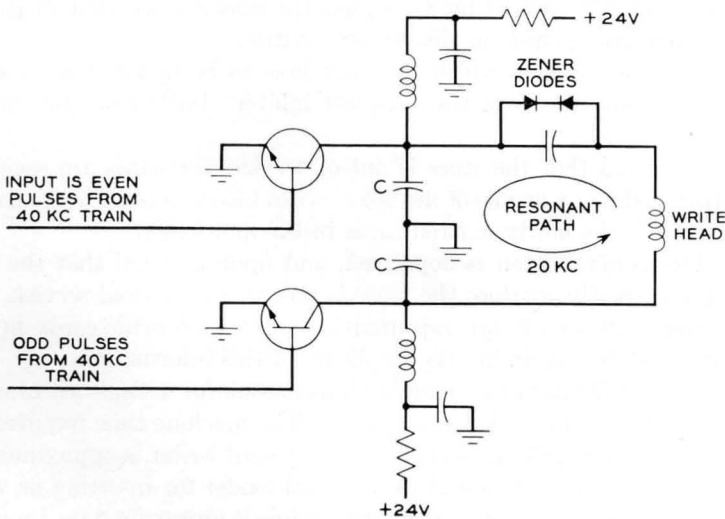


Fig. 32 — Circuit schematic — write head driver.

register the bits to be written. The control of the motors and solenoids is mainly by relay logic.

There are two drive motors, each 208-volts, 3-phase, 60-cycle. Both motors must be abruptly stopped, the head motor when at the home position and the finger motor when full forward and when full back. This braking is supplied by passing a direct current through two of the phases of the motors. The current is initially supplied by a large surge from a capacitor and is capable of stopping a 3600-rpm, fractional-horsepower motor in less than a revolution. The direct current is then allowed to decay to a value within the rating of the motor.

### 7.5 Translation Changes

Changing the translation information which is stored on the twistor memory cards is accomplished in the telephone office with the aid of two card loaders, the memory card writer frame, and a spare set of 128 twistor memory cards which is usually stored in one of the card loaders. The procedure normally followed in updating translation information is as follows:

(1) The first module to be rewritten is selected and an instruction is typed to the system identifying this module.

(2) A card loader containing the spare set of cards is mounted on the memory card writer in the pass A position and appropriate buttons are pressed to initiate the writing. On signal, the loader is inverted for pass B and the remaining cards in this set are written.

(3) The first store in which the module is to be updated is removed from service by means of the "request inhibit" button on the control panel.

(4) On signal that the store is out of service, the cards are removed from this module by means of the second card loader and are immediately replaced with the newly written cards in the first loader.

(5) The verify button is depressed, and upon a signal that the new cards are correctly written, the store is returned to normal service.

(6) Steps (2) to (5) are repeated in order to rewrite cards in the memory module containing the duplicate of this information.

A simplified diagram for changing information for a single set of cards in one program store is shown in Fig. 33. The machine time required for processing a set of 128 cards in the memory card writer is approximately 10 minutes, and time required by the card loader for inserting or withdrawing a set of cards from a memory module is approximately  $\frac{1}{2}$  minute each. Including manual operations, a total time of about 12 minutes is required to update information in a single memory module. However,

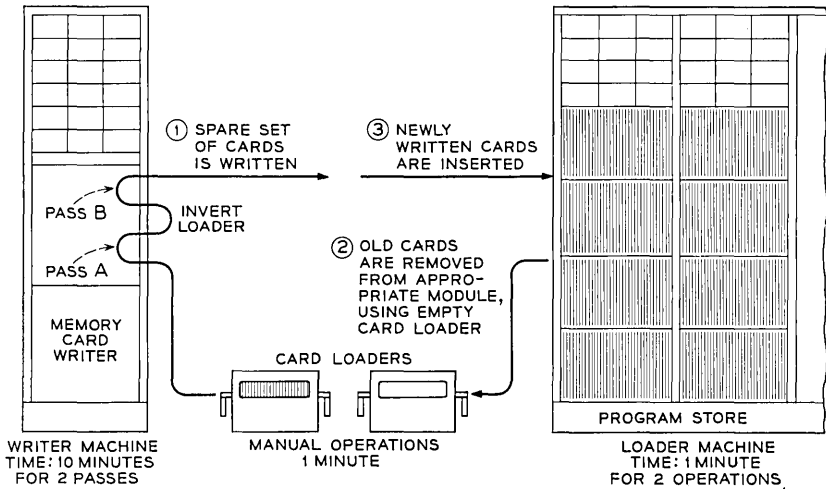


Fig. 33 — Card writing sequence.

it is seen that the down time during which the store being changed is unavailable to the system is only a little more than one minute per module.

In the event that a set of cards is incorrectly written, this is indicated after insertion into the store by the failure of the “verify” lamp on the control panel to light. In this case the new cards are immediately removed and replaced with the old set, and the store is returned to service.

### 7.6 Program Changes

In an operating office there is seldom an occasion for writing program information. However, during system debugging work this is the primary use of the card writer. To facilitate this operation a special laboratory machine called a tape reader assembler and processor (TRAP) was developed.

TRAP contains a conventional digital tape transport, a small core buffer memory, and a moderate amount of logic. Ordinarily, TRAP is supplied with a tape prepared on a commercial data processor containing the program information in the order the card writer will write it. TRAP reads the tape into the buffer and supplies a word at a time to the card writer.

An alternate mode of operation of TRAP permits the writing of a single word on a card. This is of considerable usefulness in making short program corrections.

The modifications to the card writer for TRAP operation involve the addition of several special circuit packs. These are not normally provided in an operating office.

#### VIII. PERFORMANCE

The program store is designed to operate over wide battery voltage limits (42 to 52 v) and ambient temperatures (32°F to 115°F) with any stored pattern of information. The evaluation of a unit of this size to assure it has met its design objectives is far from an incidental part of the development program. Ultimately the full evaluation requires operation with a complete ESS. To achieve a simulation of such operation a test set was developed to operate the store and analyze its outputs. The large number of variables involved in such an operation is clear from the control panel of the test set (Fig. 34).

Because the store has a read only memory, evaluation with different patterns of information is laborious. In lieu of actual worst-case patterns, a limited number of worst-case repetitive patterns is used. To ensure that under actual worst-case patterns the store will retain margin, the one-to-zero ratio must remain above 2.5 to 1 for the repetitive patterns. A very large number of tests of this type were carried out on a prototype store at varying conditions of temperature and worst tolerance circuits. These led to changes in both circuits and modules during development. Four stores of essentially final design have been operated for more than five store years in an actual system environment. As would be expected from the worst-case design approach, this operation has uncovered no case of errors not associated with failures of circuit or writing.

Circuit failures have been gratifyingly low despite the usual initial troubles encountered in first manufacture of a complex system. The average failure rate has been one package a week, but has shown suspicious bunching that pointed toward inexperienced debugging techniques. The mean time to failure has steadily risen during the interval.

Card writing performance has been generally excellent. The objective of erasing or writing to a point where no decrease in store margins could be assigned to card writing has been consistently realized. This required a ratio of 26-to-1 in magnet field strength between a zero and a one. Forty-to-one (2 gauss to 80 gauss) is generally obtained.

Error-free writing has not been as easy to obtain. A bit error rate exceeding 1 in  $10^7$  has been routinely achieved but at this rate to produce error-free modules requires rewriting one in ten. Most errors have been due to communication or mechanical problems. It is expected that an improvement of a factor of 10 will be achieved before the initial service

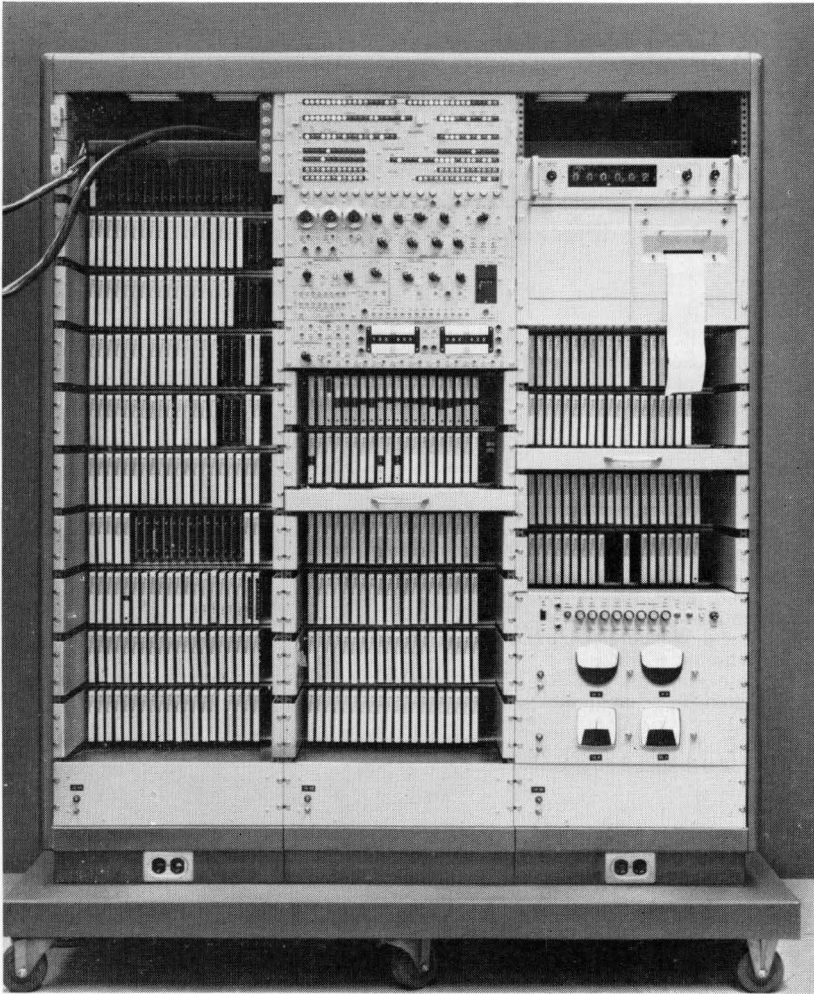


Fig. 34 — Program store test set.

date. This improvement is desired mainly to eliminate the nuisance effect of errors, since the system can tolerate a small number of errors in translation information through the recent change mechanism which allows any program store translation word to be superseded by a call store entry.

The program store has been evaluated against all of its initial objectives, and in all cases has met or exceeded them. A separate evaluation of the memory modules has been carried out and reported.<sup>7</sup>

## IX. ACKNOWLEDGMENTS

The design of the program store has involved a large number of people, both directly and indirectly. It is not possible to acknowledge individually all of the many contributors but we are well aware that without them the development could not have been a success. The authors wish especially to acknowledge the excellence of the twistor modules developed under the direction of L. W. Stammerjohn and J. J. Suozzi.

## REFERENCES

1. Downing, R. W., Nowak, J. S., and Tuomenoksa, L. S., No. 1 ESS Maintenance Plan, B.S.T.J., this issue, p. 1961.
2. Bobeck, A. H., A New Storage Element for Large Sized Memory Arrays — The Twistor, B.S.T.J., **36**, Nov., 1957, pp. 1319-1340.
3. Looney, D. H., A Twistor Matrix Memory for Semi-Permanent Information, Proc. Western Joint Computer Conf., March, 1959, pp. 36-41.
4. Gianola, U. F., Looney, D. H., Ruff, J. A., and Munn, A. J., Large-Capacity Card-Changeable Permanent Magnet Twistor Memory, *Symposium on Large-Capacity Memory Techniques for Computing Systems*, Macmillan Company, May, 1961, pp. 177-194.
5. De Buske, J. J., Janik, J., Jr., and Simons, B. H., A Card-Changeable Nondestructive Readout Twistor Store, Proc. Western Joint Computer Conf., March, 1959, pp. 41-46.
6. Ferguson, J. G., Grutzner, W. E., Koehler, D. C., Skinner, R. S., Skubiak, M. T., and Wetherell, D. H., No. 1 ESS Apparatus and Equipment, B.S.T.J., this issue, Part 2.
7. Stammerjohn, L. W., An Evaluation of the Performance of the Permanent Magnet Twistor Memory, Proc. Internatl. Conf. Nonlinear Magnetics, April, 1964.
8. Connell, J. B., Hussey, L. W., and Ketchledge, R. W., No. 1 ESS Bus System, B.S.T.J., this issue, p. 2021.



# No. 1 ESS Call Store — A 0.2-Megabit Ferrite Sheet Memory

By R. M. GENKE, P. A. HARDING and R. E. STAHLER

(Manuscript received January 10, 1964)

*A call store is an 8,192-word, 24-bit per word memory system using multi-aperture ferrite sheets in a destructive-readout, coincident-current configuration and operating in a 5.5-microsecond read-write cycle. It is designed for 40-year life and operation in ambient temperatures from 32° F to 115° F.*

*The paper describes in detail the memory medium and memory circuits. General description is given of call store usage on the common bus in the No. 1 ESS and the maintenance and diagnostic features provided. Finally, a discussion of testing and test pattern philosophy is included, along with results measured on production stores.*

## I. INTRODUCTION

### 1.1 System Requirements

As the name of the call store implies, its prime function is to provide the read-write or erasable storage for all the call-related temporary information processed by the No. 1 electronic switching system (ESS).

Aside from the conventional call processing functions,<sup>1</sup> other common system functions involving administration, maintenance, AMA and TTY buffering and network simplification (by map and queue techniques) are also assigned to the call store. This concentration of system functions has imposed rather large capacity requirements on the store. Of course the total number of bits required is a function of both the number of lines in the office and the number of calls per busy hour. A range of 100,000 to 4,000,000 bits is required to cover the gamut of office sizes and calling rates envisioned for No. 1 ESS. To achieve a compatible solution, a specific store size is required to provide a modular building block with few stores in small offices and many in large offices.

Maintainability and dependability considerations imposed rather extensive requirements<sup>2</sup> on the call store. A duplication scheme to permit

economic growth by single stores requires splitting the memory in each store into two halves or two information blocks, with common access circuitry used for both. Each information block is then duplicated in another store.

Furthermore, high-speed and reliable communication between the call stores and the central controls, as well as the high-speed switching of duplicated information blocks, calls for the use of a bus system.<sup>3</sup> All the call stores in the office share the same set of signaling and addressing leads, with each call store on the bus capable of recognizing its own name code (sent as part of the address) and responding with the addressed word. Each store must communicate over either or both duplicated buses as directed by the central control. Furthermore, the central control must have the option of changing or rerouting these central control-call store bus assignments at system cycle speeds, thus bypassing faulty individual stores or buses to achieve an operational system.

Special maintenance control modes are required to permit use of the normal communication bus for the interrogation of various internal test points within the store and also the alteration of the operational features of various internal circuits. Separate monitor and direct scanning buses with access to additional internal points are also necessary to allow alternate channels of interrogation. The system requires verification of proper operation and transmission. This is accomplished by a parity check on each address received, as well as tests of a number of internal store circuits.

Finally, the call store must be designed against a 40-year life requirement with no program or environmental restrictions. These requirements dictate redundancy and safeguards against catastrophic failure, so that failures can be rectified by maintenance. Standardized plug-in circuit packages must be used to make failed components readily accessible and rapidly repairable.

## II. DESIGN OBJECTIVES

### 2.1 *Memory Medium*

The basic memory medium selected for the call store is the multibit ferrite sheet with a  $16 \times 16$  array of holes.<sup>4,5</sup> A copper conductor is evaporated and subsequently electroplated onto the sheet in a pattern forming the equivalent of a copper wire threading through all 256 holes in series.

The plated lead on the homogeneous sheet is a prime factor in achieving low memory module fabrication costs, since this eliminates the tedious

hand wiring operation that would be necessary to interconnect 256 discrete memory elements in an equivalent manner and lends itself to automated high-level manufacturing techniques. Also, the precise geometry of the ferrite sheet permits easy registration of holes in large stacks, and this simplifies and reduces the cost of the few hand wiring operations that remain.

The use of ferrite sheets also reduces the number of soldered connections over the use of discrete memory elements by a factor of 5 to 1. This is extremely significant in large-scale memories and enhances the over-all reliability and dependability.

To achieve these benefits in terms of cost and reliability, a considerable initial device development effort was required, which, based on present-day yields and performance, was well worth the effort.

### *2.2 Store Characteristics*

The organization of a single call store is chosen to provide a capacity of 8192 words of 24 bits, or a total storage capacity of 196,608 bits. This appears to be a size that satisfies the minimum requirements of small offices and provides the growth flexibility for the larger offices. The 24-bit word length and the 5.5- $\mu$ sec cycle are dictated by general system considerations and do not represent an attempt to realize the maximum capabilities of the memory. A median-size office of 10,000 lines having a calling rate of 13,000 calls per busy hour would require only two basic call stores. Of course, two additional units would be added for duplication purposes. Many smaller offices would require only one call store and one duplicate, while large offices would require up to 40 total stores.

The 768 ferrite sheets necessary to provide the 196,608 bits are wired for coincident-current access. A 13-digit binary address is necessary to control the access selection.

### *2.3 Environment*

The call store must be operational over an ambient temperature range of 32°F to 115°F without recourse to environmental control.

### *2.4 Simplified Block Diagram*

The simplified functional block diagram of the call store is shown in Fig. 1. There are four major blocks: the memory module, the memory circuits, the logic and sequence circuits, and the maintenance circuits.

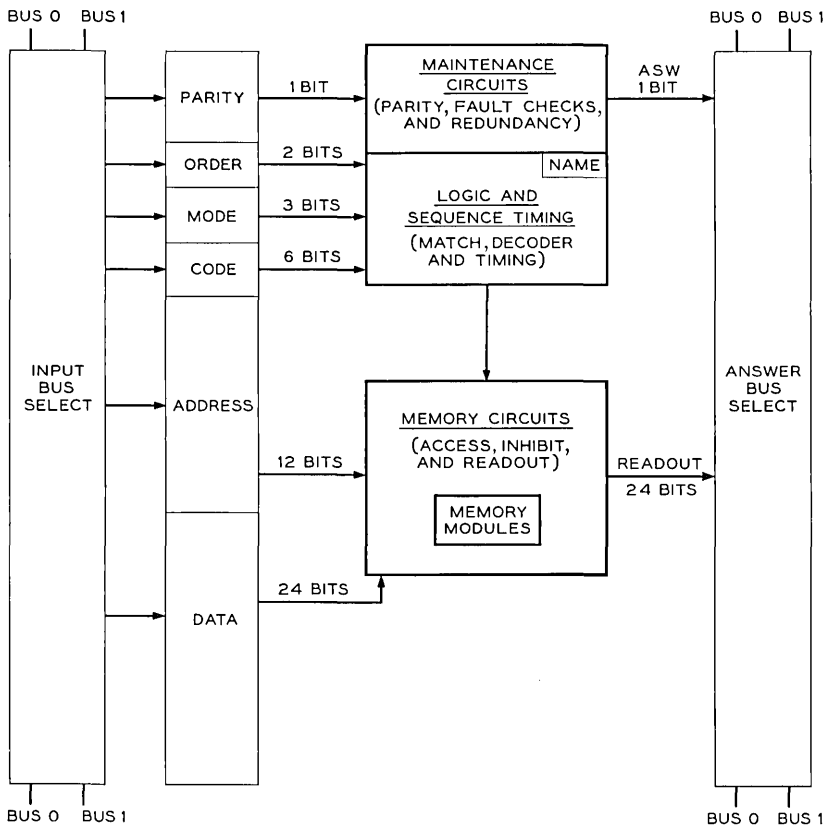


Fig. 1 — Simplified functional block diagram of call store.

2.4.1 *Memory Modules and Memory Circuits*

The ferrite sheet memory modules are surrounded by the memory circuits. A 12-bit memory location address directs the access circuits to switch the 24 "holes" desired in either of the two information blocks in the store. A single bit is derived from the name code to select one of the two information blocks. The readout signals from the memory are amplified and returned on the answer bus.

In the "write" mode the data register is set by the 24-bit data pulses received via the input bus.

In the "read" mode the data register is used for buffer storage of the readout, which is then rewritten into the memory.

#### 2.4.2 *Logic and Sequence Circuits*

The logic and sequence control circuits decode the incoming 6-bit name code, the 2-bit read or write order, and the 3 mode bits to determine whether or not the call store should respond and in which mode of operation. Internal timing circuits are also included in this block.

#### 2.4.3 *Maintenance Circuits*

The maintenance circuits check the correctness of the single parity bit generated over both the code name and address. These circuits also verify a number of internal store test points, and if these and the parity are all satisfactory an "all-seems-well" pulse is generated and sent on the answer bus.

Aside from the basic read and regenerate and erase and write modes, other modes are added to facilitate the maintenance of the store.

First, a "maintenance" mode is added in which the store responds as a memory but most of the flexibility in store selection and in store response is eliminated. Only one store can respond, and the store address is uniquely pinpointed by the incoming address.

A "control read" mode permits the interrogation of test points within the store. It accomplishes this by bringing the test data from the selected test points through the data register out on the answer bus.

A "control write" mode allows central control to alter the internal control states in the store. Special data and address codes are used to internally manipulate all the circuitry without actually switching the memory. Thus central control is able to electronically change the particular memory assignment or code name of a store.

The three major circuit blocks, along with the memory modules, will be discussed in detail in the following sections.

### III. MEMORY

The basic storage device is a multiaperture ferrite sheet approximately 1 inch square and 30 mils thick. Each sheet contains a 16-by-16 square array of holes 25 mils in diameter, placed on 50-mil centers. The material used in this device is a mixed magnesium-manganese ferrite similar to those used in square-loop toroids but with a higher Curie point, which allows an extended temperature range. Square hysteresis loop characteristics of 1.9 oersteds coercive force, 1800 gauss saturation flux density and 0.97 squareness ratio are the governing parameters of the material. However, behavior of the material surrounding a hole is also influenced by

neighboring material and magnetic field distribution. Each hole, when excited by approximately 250-ma coincident-current pulses, acts as a square-loop memory core with the field constrained to the annular region immediately surrounding each hole. Interaction effects are reduced by generating opposing fields in neighboring holes. In this application the typical disturbed "one" output of a switched hole is 75 millivolts with 0.9  $\mu$ sec switching time.

A printed copper conductor linking all the holes on a sheet is one of the four necessary conductors (see Fig. 2). The three remaining wire conductors are threaded after the ferrite sheets are mounted into stacks.

### 3.2 Constraints of Large Coincident-Current Ferrite Sheet Memories

Three of the four conductors combine to form a standard coincident-current access.<sup>6</sup> In this application there are 128  $X$  coordinates and 64  $Y$  coordinates whose product (8,192) equals the number of words in the memory. Simultaneous half-read currents,  $I_d$ , along one of the  $X$  and one of the  $Y$  coordinates, are sufficient to fully switch the selected holes of an address to the read or "zero" state but will not disturb any of the many holes half-selected by  $X$  current and  $Y$  current. Subsequent re-

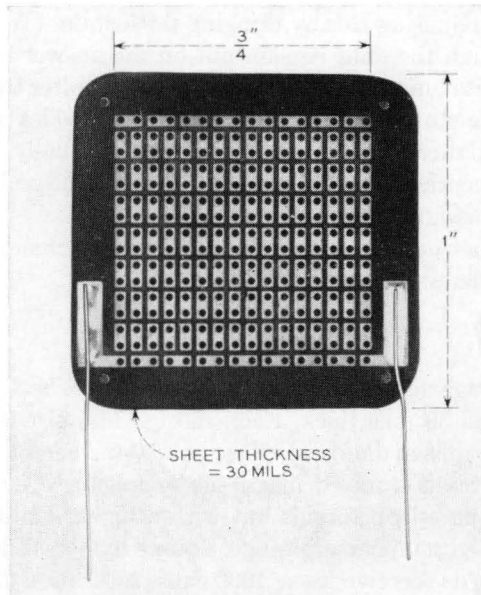


Fig. 2 — Ferrite sheet.

versal of current on the same conductors switches the previously selected address to a write or "one" state unless a separate read direction half-drive, called an "inhibit current," is applied on another conductor. For a 24-bit word there are 24 distinct bit planes, each containing an inhibit conductor and a sensing or read conductor; both are coupled through all the 8,192 address-location holes.

The major constraint in coincident-current memories is on the sense wire noise voltages due to the accumulation of signals across holes excited by the half-current drives. Attempts are made to cancel these signals by directing half the sense and drive lead hole intersections positively and the remaining half negatively. However, the positive and negative noise voltages, which are functions of information storage, past history and addressing sequences in other sections of the memory, do not cancel exactly, leaving a residue called "delta noise." A byproduct of delta noise is the large and variable load imposed upon the  $X$ ,  $Y$  and inhibit drivers. The impedance to a driver, due to the summation of the inductances of each hole which is pulsed by the driver, is a function of the information stored and the remanent state of flux. These factors are, in turn, a function of the memory contents and the prior addressing sequences. The variable-load inductance, as well as the large interwinding capacitances, forms complex transmission lines that impose severe design problems on the drivers.

Methods of generating delta noise and driver loading variations are discussed in Section 9.2.

A unique problem to the ferrite sheet is the interbit reaction due to magnetic coupling of flux between the holes. To combat interhole coupling, special flux constraining wiring patterns are used which limit the fields to annular regions about each hole. (See Fig. 3.) The constraint is simply generated by guaranteeing that any hole selected with a full drive during read or write intervals is countered by an opposite half drive in the nearest neighbor holes. Therefore special wiring rules, covered in the next section, are applied.

The fixed 16-by-16 array of holes imposes another constraint, because this size must be used as the basic building block. But this larger-size sheet reduces the number of connections and improves reliability. The 24-bit per word application necessary in No. 1 ESS is achieved by coupling the sense wire through two adjacent rows for each bit; the 16 rows in a sheet can be subdivided into 8 row pairs which allow 8 bit planes. The full complement of 24 bits is formed by using 3 stacks of sheets, each stack supplying 8 rows of paired bit planes. This folding of the read or inhibit plane complicates  $X$ -plane wiring because two sep-

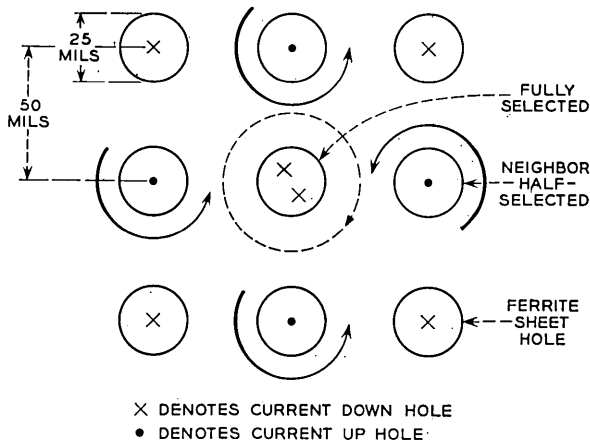


Fig. 3 — Flux locking.

arate  $X$ -plane conductors are necessary to guarantee that only 1 bit is switched in a bit plane and that all ferrite sheet holes are fully utilized.

### 3.3 Basic Module — Size and Wiring Patterns

The 8,192 words of a call store are obtained from 2,048-word modules which in turn are constructed from 512-word submodular assemblies. The modules are interconnected with all electrical connections on a plug-in basis, while the submodule assemblies are permanently mounted. Conductor length and possible increased manufacturing (yield) rejection are the major considerations which limit the maximum module size. The number of interconnecting terminals increases for smaller modules, with an accompanying reduction in reliability and an increased cost. A compromise between the yield or wiring length problems of a large module and the reliability or cost constraints of a small module has led to the 2048-word size.

The formation of the ferrite sheet module is dependent upon a number of noise-reducing techniques. An attempt is made to maintain all current carriers on a tight twisted-pair basis outside of a module and within the module, with geometric orthogonality between  $X$ ,  $Y$ , and inhibit conductors. The positive and negative terminals of all conductors are arranged in close proximity. Since the read cannot be completely orthogonal with the  $X$  and inhibit, another basic requirement introduced complementary wiring with as much geometric symmetry as possible to realize balance and assist noise cancellation. Noise cancellation tech-



niques take on particular importance for the readout wire because of the large capacitive and magnetic coupling between drive and readout leads. Consequently, special noise cancellation wiring patterns are used.

A single ferrite sheet module is shown in Fig. 4. This unit with the four subassemblies, each containing three columns of sheets, is mounted with all holes in registration. The submodules are first fabricated with separate read conductors, each woven with special wiring patterns. These patterns (shown in Fig. 5) minimize coupling with the *Y* or inhibit drive lead by reversing the sense direction in a binary fashion. For example, the first two columns of read holes intersect the *Y* in the positive sense and the next two in a negative sense. For the next set of four holes the pattern is reversed. For the next set of eight holes the first eight-hole pattern is reversed, and so on. This scheme compensates for the variation

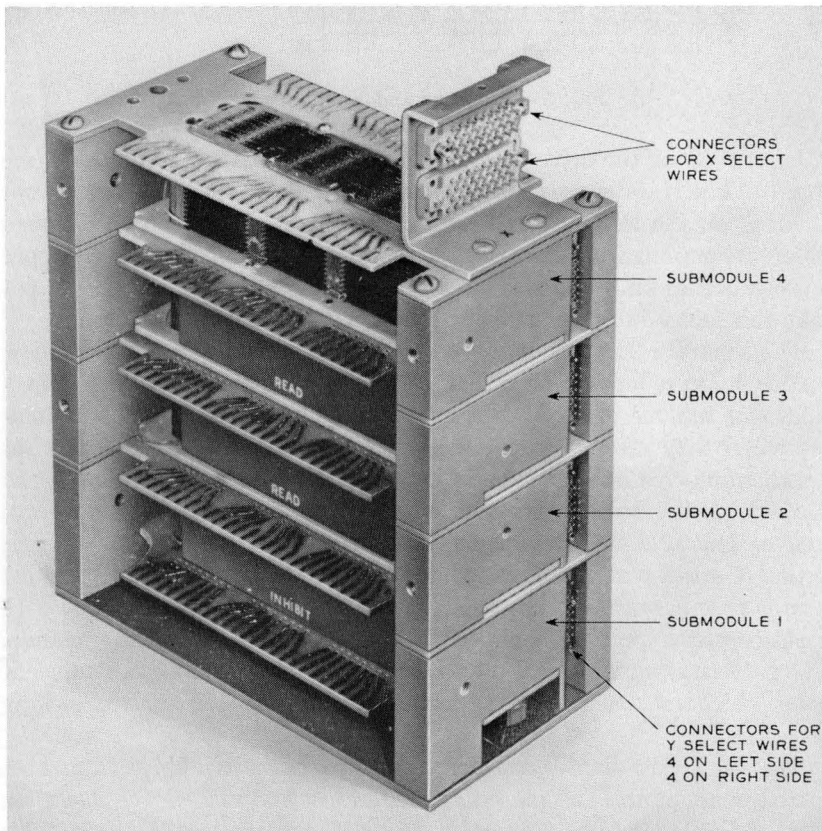


Fig. 4 — Module.

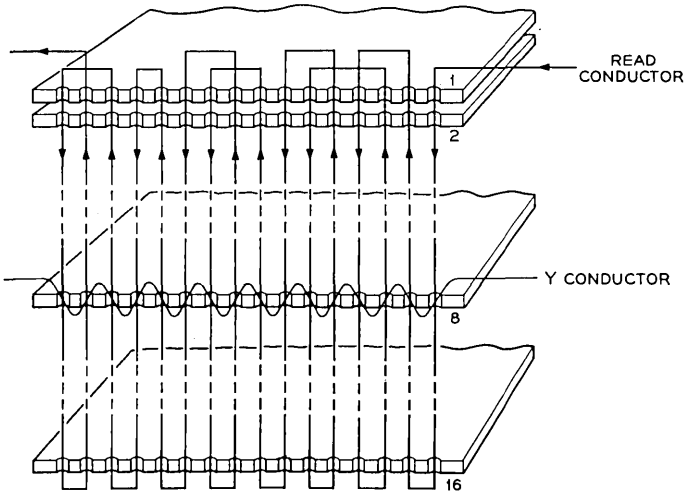


Fig. 5 — Readout and  $Y$ -axis wiring.

of current along the drive wire due to distributed capacitance losses (see Fig. 6). The shunt losses due to stray capacitance degrade the current entering on the left to that shown on the right. This difference causes different coupling from the drive to the readout wire and prevents proper noise cancellation. Simple inductive cancellation schemes which do not take this factor into account give less effective noise cancellation.

The separate 24 readout wires for each submodular assembly allow further noise reduction, as well as introducing flexibility of interconnections (for reasons to be described presently). Once the four submodules are wired they are mounted one above the other. The 24 inhibit wires are then inserted like the read wires, except that they pass up and down through all the holes of the four submodules in a simple up-and-down weave. The 32  $X$ -select wires are threaded orthogonally to the inhibit, with 2  $X$ -select wires per  $X$  plane, so that each  $X$  wire intersects a folded read wire only once on a sheet. Fig. 7 illustrates the  $X$  weave wired with a close-twisted pair arrangement both within and outside the module. The pattern also reflects the added requirement of coupling two adjacent holes which are not coupled by the same inhibit, to satisfy flux locking constraints.

The 64  $Y$  coordinates are derived from the individual ferrite sheet plated conductors. The plated conductors of the same level sheets on each of the three stacks are soldered together to form a  $Y$  plane, with 16  $Y$  planes per submodule and 64  $Y$  planes per module. Although a

careful screening process on individual sheets is implemented prior to assembly, occasional defective sheets might occasionally appear in a finished module. One additional  $Y$  plane is included per submodule to allow a defective sheet to be replaced by a spare. This substitution is employed only in the manufacturing process.

To summarize the makeup of the completed module, there are four  $Y$  connectors on one side for entrance and four  $Y$  connectors on the other side for exit, each connected to a submodular assembly of 16 coordinates, each coordinate containing 3 sheets. The twenty-four twisted-pair read-outs are brought out to four printed wiring boards, one at each submodule. One printed wiring board located at the bottom of a module provides access for 24 twisted-pair inhibits. Two connectors are located at the top of the module where the twisted-pair  $X$ -selected wires originate and terminate. Each module contains 64  $Y$  coordinates, 32  $X$  coordinates and 24 bit planes.

### 3.4 Memory Organization

Overcoming the prohibitive delta noise and the large, variable driver loads in an undivided or unduplicated memory, is a difficult goal for a

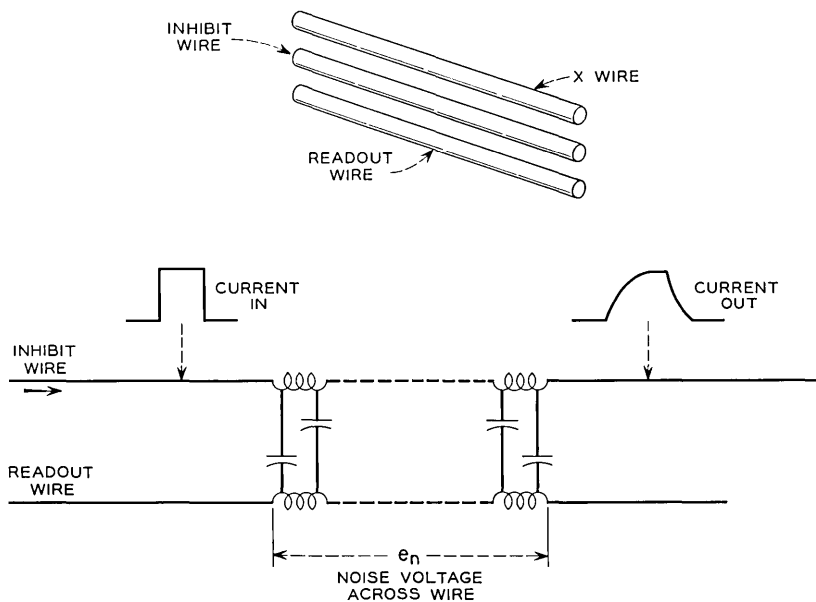


Fig. 6 — Inhibit transmission line.

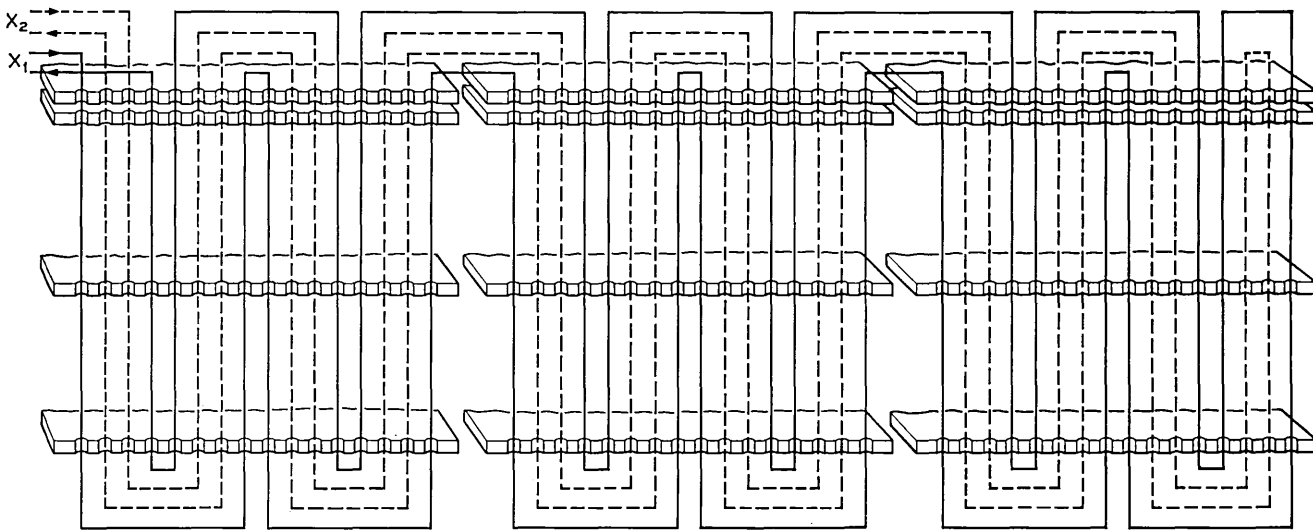


Fig. 7 — X-wiring module cutaway.

large memory designed for a 40-year life without potentiometer adjustments. Circuit schemes are introduced which drive large loads, stabilize currents, reduce delta noise via a post-write disturb, and shape current rise time. A novel scheme of measuring the noise with sample currents and subtracting the predicted noise during the read interval is also included. However, these appliques by themselves are insufficient to overcome the large delta noise of a full 8192-word coincident array with the extended life and reliability requirement of a telephone switching office.

Consequently, in this system the inhibit, read, and  $Y$  coordinates are subdivided and duplicated. The duplication is incorporated in such a manner as to minimize delta noise and reduce interactions between the coordinates, and cost reduction techniques are developed by utilizing common controllers as much as possible and placing the minimal hardware in the duplicate sections.

The interaction and delta noise minimization is illustrated by means of a Venn diagram (see Fig. 8). The matrix shown represents one of the 8192-word bit planes containing 64 horizontal  $Y$  lines and 128 vertical  $X$  lines. The matrix is split into 16 sections, each a 16-by-32 coordinate array with the upper 8 sections coupled to the A-read wire while the lower 8 are coupled to the duplicate B. The sections labeled  $\alpha$  are driven by the  $\alpha$  inhibit, while the remainder are driven by the duplicate  $\beta$ . The one-half length horizontal line, representing a  $Y$  current into the memory, shows the  $Y$  duplication, while the full-length vertical line represents an  $X$  current into the memory. The horizontal and vertical line segments indicate how the memory should be organized to minimize the coupling

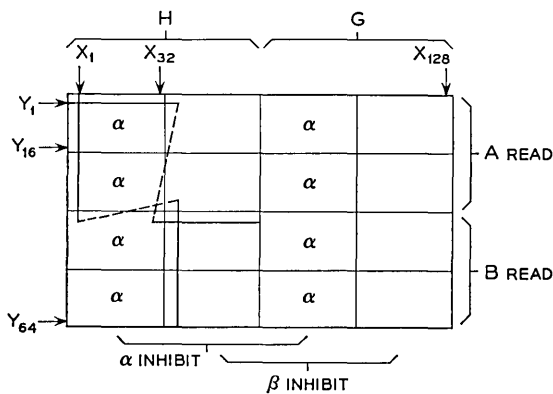


Fig. 8 — Bit plane, Venn diagram; minimum interaction and delta noise.

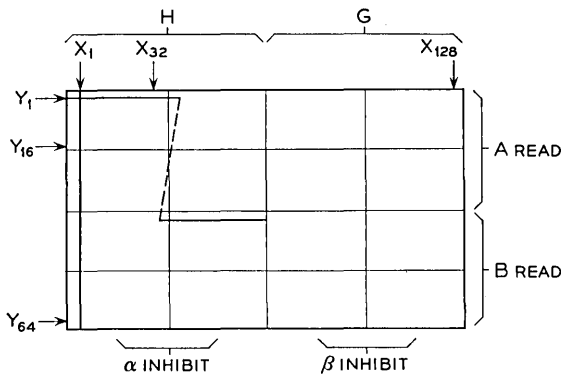


Fig. 9 — Bit plane, Venn diagram; compromise scheme.

of half-drive holes to the read wires and the coupling of inhibit currents to the  $X$  and  $Y$  wires. Since this organization is highly related to the module construction and fabrication, it is difficult to completely realize the objective; a compromise scheme results (see Fig. 9) which reduces terminal and solder connections.

Fig. 10 is a block diagram of the full 8192 words formed from 4 modules and shows the  $Y$ , inhibit, and read duplication. The first submodule read conductors are connected in tandem to the first, second, third and fourth modules. A jumper at the end cross-connects two adjacent read conductors to form a folded read plane. Each bit plane is split into two segments. One segment is derived by interconnecting the first and second submodules of all four modules, while the second segment is derived from the third and fourth submodules. Each segment is terminated by a separate preamplifier with 24 preamplifiers in the upper A half and 24 preamplifiers in the lower B half. The 2 preamplifiers per bit plane are recombined by an exclusive OR gate into a decision-level circuit.

The duplicated 64  $Y$  coordinates are interconnected between modules so that  $Y$  currents through submodule 1 of module 0 will travel to submodule 3 of module 1. This type of cross connection reduces the  $Y$ -to-read noise by one-half. The  $Y$ -access source is an 8-by-8-by-2 three-dimensional matrix with current steering logic to decide if the G or H half is chosen. Through this technique the more expensive selecting switches and current generators are shared by both  $Y$  halves. Only the selecting diodes, some transformers, and two switches must be doubled.

The 128  $X$  selections, with 32  $X$  coordinates per module, are not duplicated. However, the noise coupling between the  $X$  and read has been cut in half because of the A and B read segments. For this access an

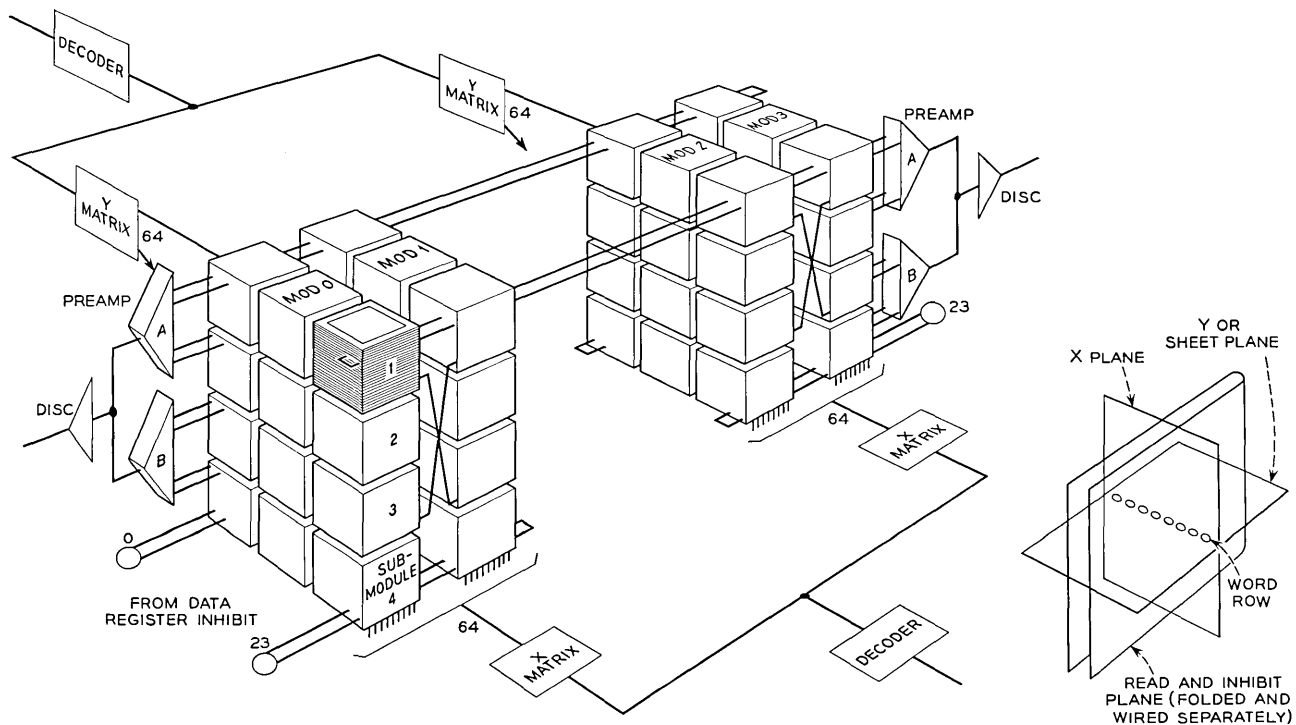


Fig. 10 — Module interconnections.

CALL STORE

8-by-8-by-2 matrix, identical to the  $Y$  matrix, introduces significant cost reductions.

The 24 separate write back or inhibits also are duplicated because of the large inhibit wire inductance. One set of 24 inhibit drivers is coupled to the first two modules and the second set is coupled to the last two modules. This duplication is orthogonal to the read split to minimize the intersections between read and inhibit.

#### IV. MEMORY CIRCUITS

##### 4.1 *Readout*

The readout complex is a circuit block which equalizes near- and far-end read signals to the preamplifier output. It contains two 3-stage high-gain transistor feedback preamplifiers with an EXCLUSIVE-OR gate for recombination of the read segments. DC restoration is used to reduce bias drifting caused by long bursts of unidirectional inputs, along with a scheme that measures the noise before sampling and subtracts the undesired noise from the output. The complex also accepts either positive or negative signals and samples or strobes the signal into a bidirectional gated-oscillator decision-level circuit (see Fig. 11).

Transmission line characteristics causing unequal response of read signals from near- and far-end points can create severe degradation of performance. To equalize the read response for both gain and phase, two transformers are used to couple both ends of the read wire, matching the characteristic impedance and balancing the transmission. The secondaries are connected in series to reconstruct the full source amplitude without attenuation.

The EXCLUSIVE-OR gate, dc restoration, and special noise measurement and subtraction functions are combined by a simple resistance-capacitance and bidirectional switch scheme (see Fig. 12). In this scheme the preamplifier capacitor output is shorted to ground by a clamp unless the preamplifier is chosen; if chosen, it is enabled only during the read interval. The small time constant formed by the low preamplifier output impedance and the low resistance in series with the capacitor allows quick discharge for dc restoration. After the clamp is open, the time constant is increased and the network transmits the preamplifier output with little attenuation. By opening the clamp during the early noise peak, the noise is measured and stored across the capacitor. During the subsequent sampling interval, the capacitor voltage subtracts from the input noise voltage, reducing the noise or undesired signal amplitude. The clamp is



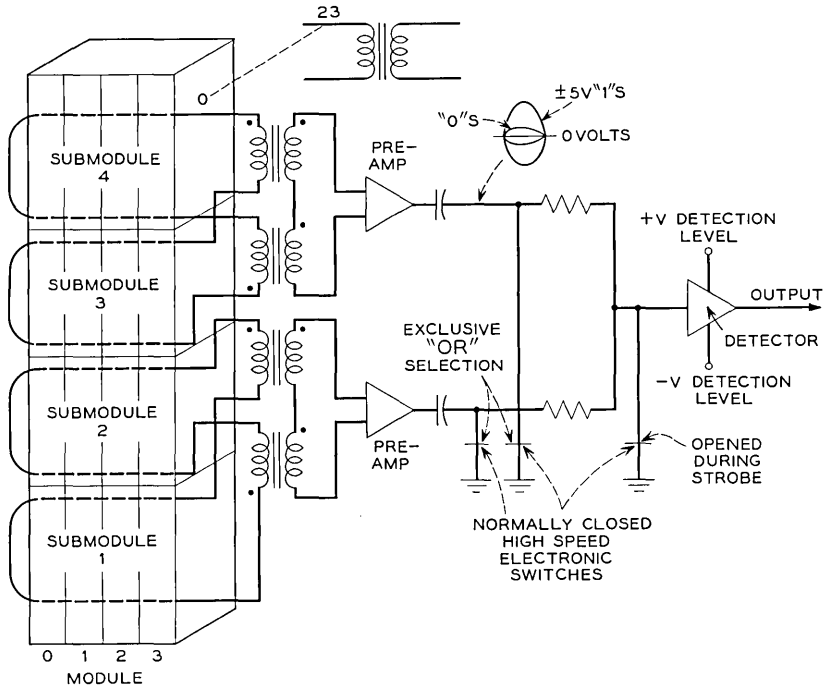


Fig. 11 — Read circuits.

closed shortly after the read sample is taken, to continue the dc restoration.

The bidirectional sampled signal is level-detected by means of a diode bridge arrangement (see Fig. 13). Input currents of either polarity in excess of the bridge currents back bias the diodes and raise the impedance, which changes the detecting amplifier from conditionally stable to unstable. The unstable state causes a severe oscillation, detectable as a "one" output. Use of a gated-oscillator technique with an amplifier normally biased in the class A region along with a current threshold detector significantly improves the sensitivity.

#### 4.2 Current Drivers

A single transistor of the type available in No. 1 ESS was not capable of driving the large  $X$  or  $Y$  load with a 0.5-microsecond rise time. It was necessary to combine transistor circuits so that larger induced voltages could be accommodated. The solution chosen (see Fig. 14) combines 4

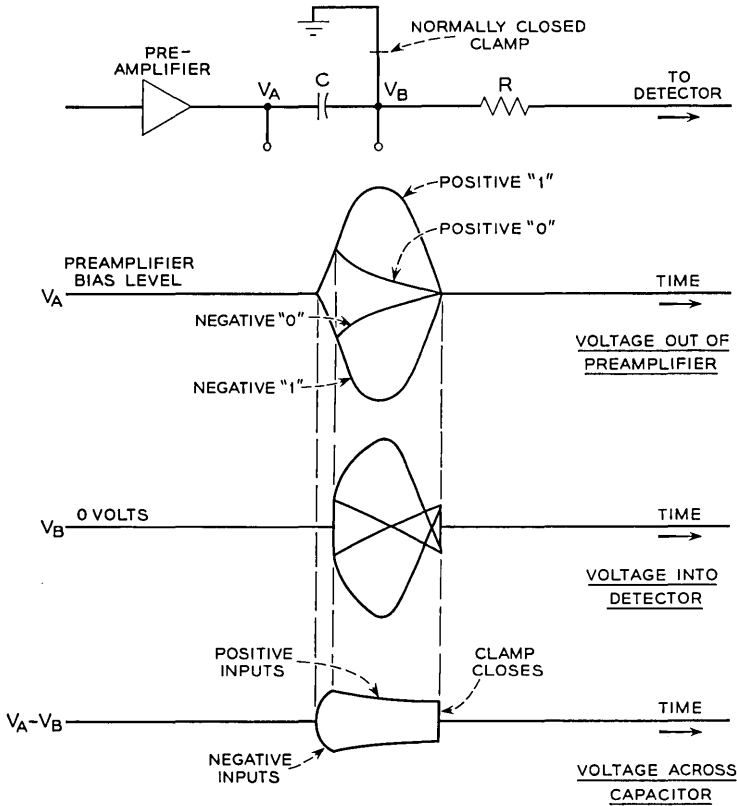


Fig. 12 — DC restoration noise reduction scheme.

separate read current drivers by means of 4 transformers with primaries in parallel and secondaries in series. A similar set of 4 write current drivers uses different windings of the same transformers; the write current drivers are fired after the read currents are turned off. The circuits, each a pulsed-current source of approximately 250 ma, combine to supply a 250-ma, 0.5-microsecond rise-time pulse in the read secondary during read time and in the write secondary during the subsequent write interval. This scheme not only distributes the load voltages but statistically improves stability, since the secondary current is equal to the average source current minus the average shunt transformer losses. The current standard deviation is one-half of a source current driver deviation, because the average of  $n$  independent but equal variables has a standard deviation reduced by  $\sqrt{n}$ .

Repetition rate effects are eliminated because the unbalanced transformer flux of the read interval is recovered by the reverse polarity drive of the subsequent write interval.

One added feature in this scheme is the two-step current rising shape. Rather than stagger the start of the  $X$  and  $Y$  currents so that the early  $X$  noise expires first, both currents are turned on in coincidence with a 125-ma first step and a 125-ma final step. This staircase is useful in three ways. First, the load voltage transients are smaller. Second, the early quarter-drive currents sample the coordinates without fully switching the selected address and the resultant readout noise is stored in the read preamplifier capacitor; during the subsequent full drive when the actual address is gated, the capacitor voltage is subtracted from the second step incoming signal, causing a significant reduction in delta noise. Third, the first step of the current staircase acts as a partial pre-read disturb, removing part of the undesired delta noise and precharging stray capacitance.

The variation of coercive force with temperature necessitates a reduction of drive current as the ambient increases. Currents should track within  $\pm 2$  per cent of an optimum value at any given temperature. To

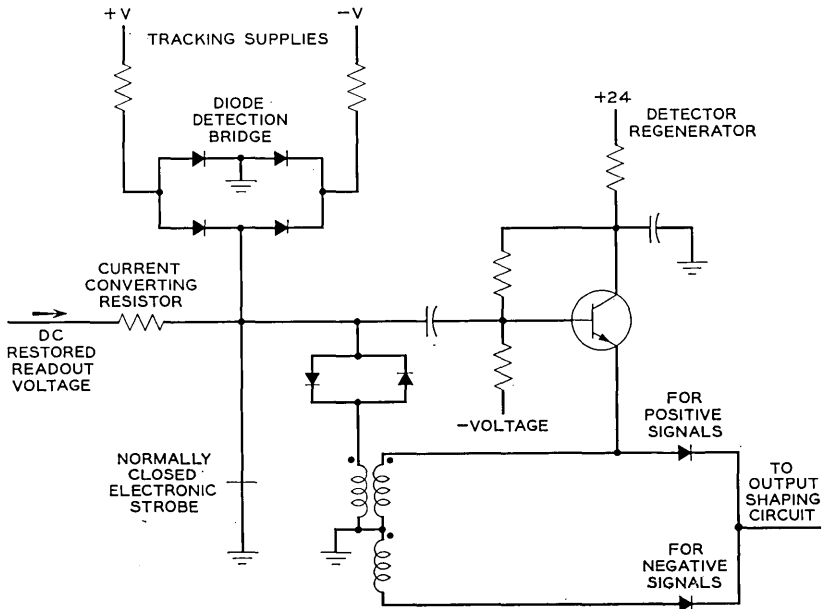


Fig. 13 — Detector.

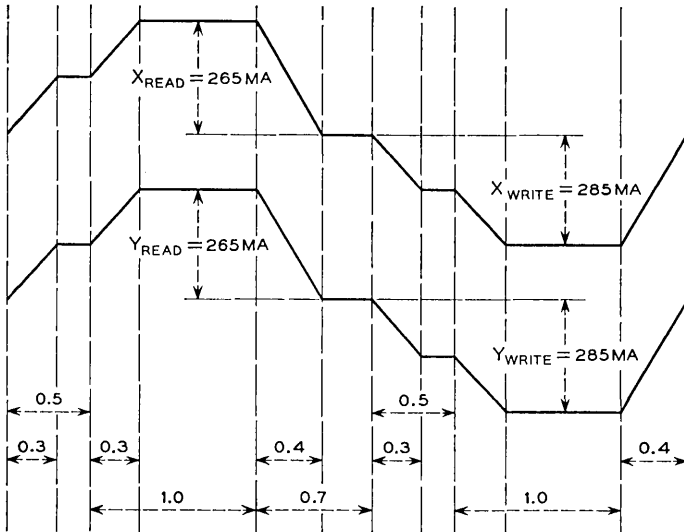
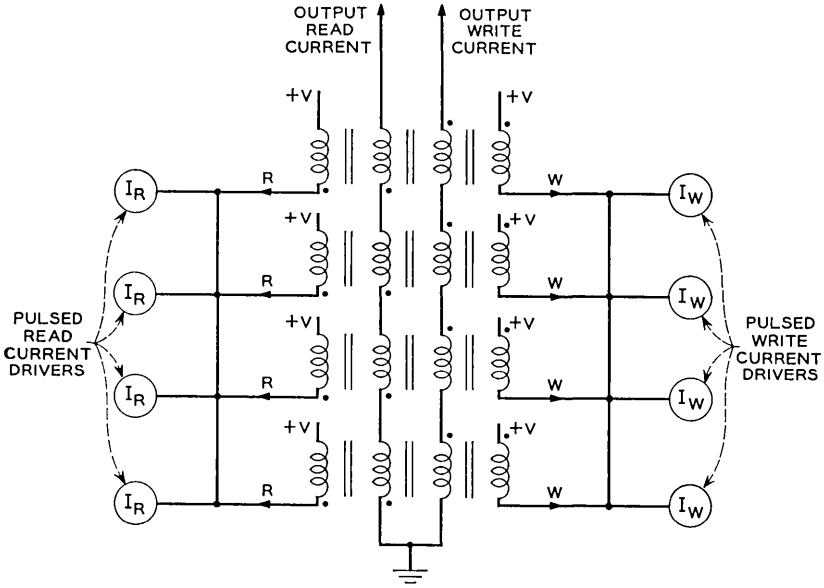


Fig. 14 — Current drivers. All values are  $\mu\text{sec}$  except those indicated otherwise.

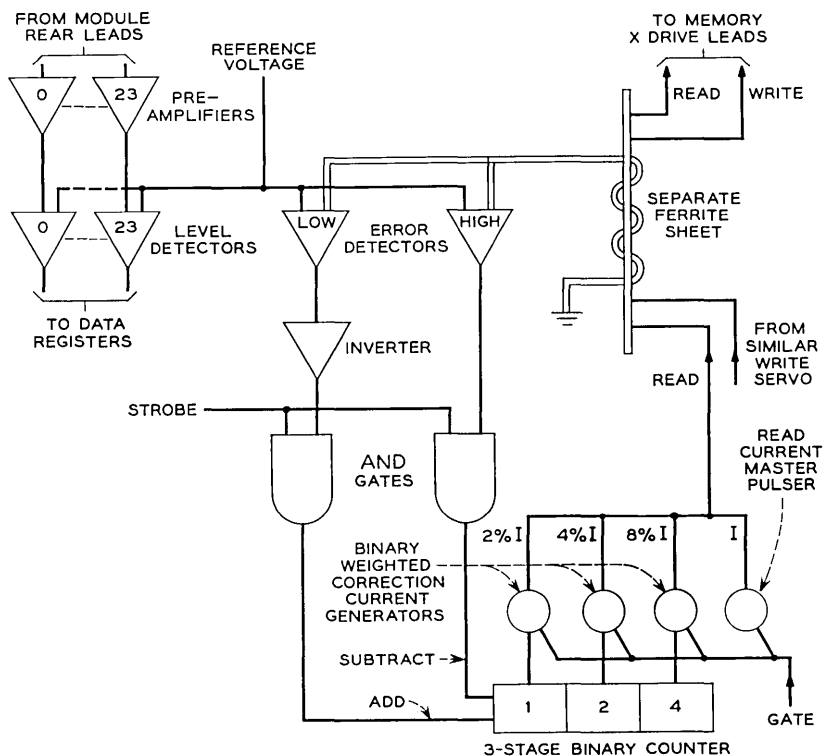


Fig. 15 — Current stabilizing servo loop.

achieve a constant readout over the ambient range of 32°F to 115°F, this optimum current must be varied by 20 per cent. The precise tracking over this large current range is accomplished by means of an electronic analog-to-digital servo loop (see Fig. 15). Two turns of the read and write drive currents pass through 16 holes of a separate sensing sheet. An equivalent 8 turns couple the sum of the 16 switching signals out of this sheet into a high- and low-level detecting circuit. If the signal is lower than a bias voltage, a one-step add command into a 3-stage binary counter is initiated. If the signal is high, a one-step subtract command is initiated. The three counter stages directly control three auxiliary current circuits whose amplitudes are binary weighted at 2, 4, and 8 per cent of the nominal value. This system automatically adjusts itself so that the sensing ferrite sheet peak output signal is constant at the high-low detector bias voltage level. However, since the ferrite sheet coercive force

changes with temperature, the currents are automatically adjusted to maintain constant peak switching voltages. This adjustment corresponds exactly to the required variation of drive current with temperature changes. Another significant feature is the stabilizing nature of the ferrite sheet, which, in addition to the extremely low aging drift, is due to the nonlinearity of the device. The product of the 16-hole switching voltage ( $V$ ) and the switching time is approximately constant for full drives, but as the current ( $I$ ) is raised, the voltage increases in the following manner:

$$dV/V = 4dI/I.$$

A 2 per cent variation in input current results in an 8 per cent variation in output signal. In this application the ferrite device amplifies the hard-to-detect small current drift and develops a large voltage variation which is easier to detect.

The drift effect of the high-low detector bias voltage is also reduced by tracking this supply with the readout detector level voltage. A downward voltage drift reduces the servo loop current, which in turn lowers the memory output signal. However, compensation for the decreased output signal is achieved by a simultaneous lowering of the readout detector threshold.

These current driver schemes are used both for  $X$  and for  $Y$ . Each contains an individual servo loop with independent controls. The  $X$  and  $Y$  read and write currents are also threaded through another sensing ferrite sheet, with the resulting 500-ma sum of the  $X$  plus  $Y$  read currents resetting 16 holes and the 500-ma sum of the  $X$  plus  $Y$  write currents setting the same 16 holes. The amplified voltage output of the 16 holes is used as a timing source for the readout clamp and sampling signals. Because of this arrangement the critical timing signals track the current rise-time variations.

#### 4.3 Access

In this system the  $X$  and  $Y$  access circuits are identical. The 1-out-of-128 coordinate selections for the  $X$  (constructed from two sets of 1-out-of-64 selections circuit), is realized by a circuit identical to that which derives the two sets of 1-out-of-64 for the  $Y$ . Costs are minimized by employing a 3-dimensional  $8 \times 8 \times 2$  diode array as the selecting source; the 1-of-128 selections can be decoded by 18 transistor switches. The positive read and negative write current bidirectional requirements, which normally necessitate a duplicate set of switches, have been satisfied with unidirectional circuits by employing a four-winding transformer-

coupled switch scheme in which a common selection of all the secondary windings determines the direction of current.

Fig. 16 illustrates a 1-out-of-64 access grouping with current drivers of the type described in Section 4.2. One of the eight horizontal and one of the eight vertical dc switches are simultaneously closed by decoding the input binary address. During read time, a closed read ac switch directs the current through the primary read winding of the selected horizontal transformer to the closed horizontal switch. This generates a pulse in the read secondary winding through the selected load, down a column line, through the selected vertical transformer, and back to the read secondary winding of the horizontal transformer. The vertical transformer is chosen by first shorting the primary to ground through the selected vertical switch and then establishing a +24-volt bias on the unselected column lines, so that all diodes in the unselected seven columns are back biased. During the subsequent write interval a closed write ac switch directs the current through the write primary winding of the chosen horizontal transformer. This reverses the current in the secondary loop and recovers the unbalanced flux generated during the read interval. The necessary biases to uniquely select one of the column lines are derived by precharge switches which insert a -24-volt bias on all unselected columns.

The combined group of the 8 horizontally arranged transformers is called an "input marker" and the combined group of the 8 vertically arranged transformers is called an "output marker." The set of 8 horizontal and vertical dc switches is shared by two groups of input markers, output markers and diode matrices. (See Fig. 17.) Each group is uniquely selected by a primary ac switch. By this technique only the passive markers and diode matrices must be duplicated when extending from a 1-out-of-64 to a 1-out-of-128 scheme.

A highly significant byproduct of the access is the precharge feature. The stray and interwinding capacitances of unselected memory coordinates can cause severe current leakages into unselected locations until these capacitances are charged. Leakages, which destroy the fidelity of the current rise-time shape, occur even if an ideal system with ideal switches is used. This system precharges all the capacitances until the unselected diodes are back biased.

#### 4.4 *Inhibit*

The 48 inhibit drivers are simple pulsed-current sources whose amplitudes are determined by a 5-volt reference bias. The voltage is designed

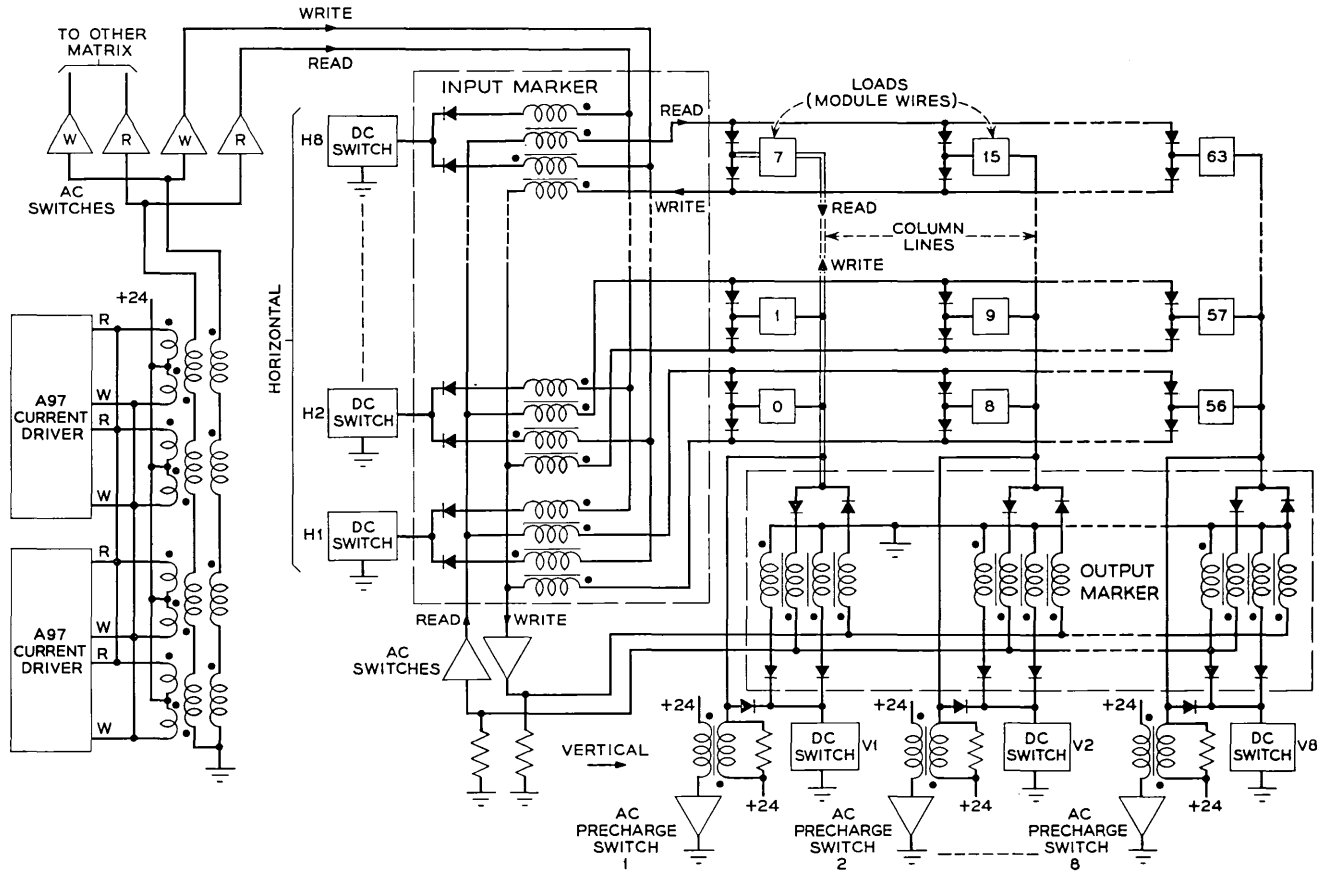


Fig. 16 — 1-out-of-64 selection matrix.



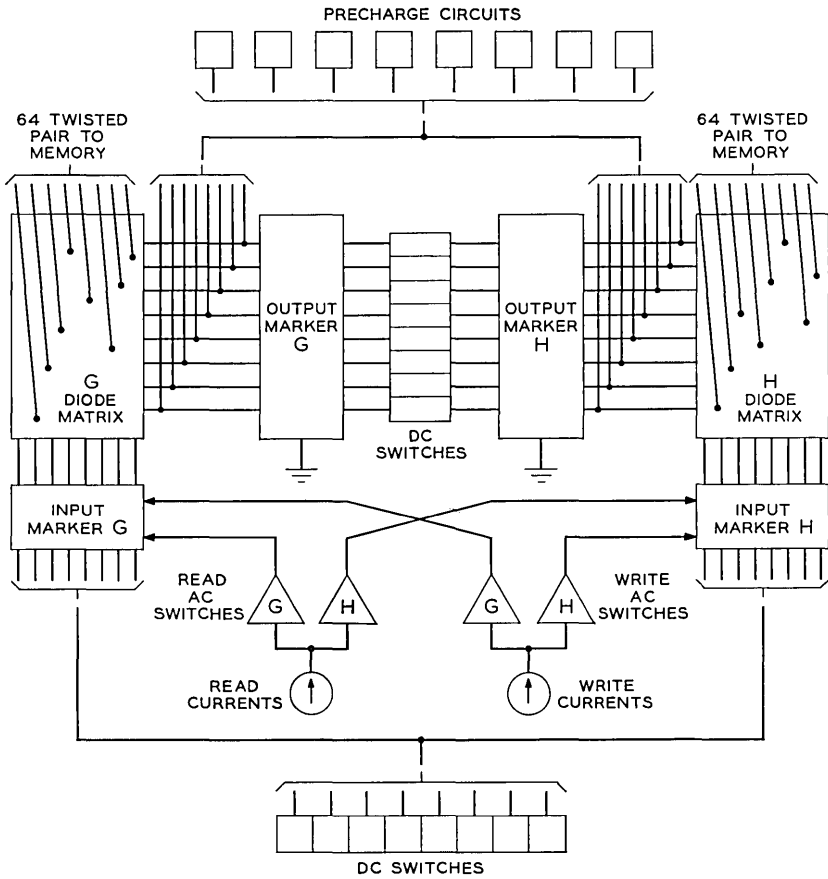


Fig. 17 — Input markers, output markers, and diode matrices.

to vary with temperature to properly compensate the inhibit current with ambient temperature. The major feature of the inhibits is the application of a post-write disturb (see Fig. 18), which reduces delta noise by referencing all holes with a half-read drive.

V. LOGIC AND SEQUENCE CIRCUITS

5.1 General

As shown in Fig. 1, the No. 1 ESS call store is a self-contained memory system that can read or write information, as ordered, in any externally

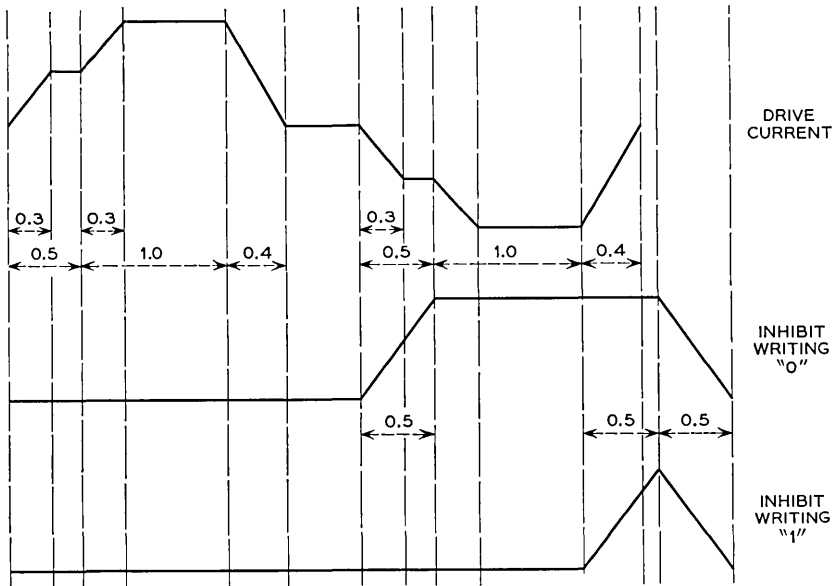


Fig. 18 — Inhibit current with post-write disturb. All values are  $\mu\text{sec}$ .

specified address location. It contains logic circuits such as address decoding and data registration, and timing circuits typical of memory systems. However, it also contains a wealth of maintenance and checking circuits not typical of memory systems. Its operation can be considered as an independent subsystem with self-diagnostic facilities. Much of the logic design is concerned with the operation of the call store in the larger system. Of particular interest is the common bus communication which provides duplication of memory and eliminates the need for expensive private buses in central offices with widely varying memory requirements. Other important sections are the automatic maintenance circuits, including fault-recognition and trouble diagnosis, which integrate the call store operation with the system maintenance plan.

### 5.2 Common-Bus Communications

Inputs are transmitted over a duplicated bus shared by all call stores. Each store is preset to receive from one of the buses by an internal flip-flop called the "R<sub>0</sub> flip-flop," which is controlled in each unit by a separate input channel. A call store will consistently receive from the same bus unless a trouble condition arises.

All call stores on the common bus receive and register every order.

An order is processed and the answer returned if the 6-bit input code matches the store name. In effect, the code is part of the information address and from the system point of view the 6-bit code and 12-bit address make up an 18-bit address for the entire office temporary memory file. For convenient bookkeeping, however, the office temporary memory is broken up into 4096 word blocks. The 12-bit address selects one word location within a block and the 6-bit code selects the memory block within the office. Recall that for growth economics the 8192-word capacity call store is divided into two 4096-word blocks called the "G half" and "H half" ("Gee" and "Haw" being a mule driver's terminology for "right" and "left" respectively). The code name of the H half is wired in a store at installation and is called the "fixed name." The G-half code name, called the "variable name," is contained in 6 call store flip-flops which are under system control and can be changed in a 5.5- $\mu$ sec cycle.

Fig. 19 shows three call stores on a duplicated call store bus. As shown, all three units are set to receive from bus 0. Answers from the H half of each store are sent back on bus 0. G-half answers are returned on bus 1. A read order with a code of 1 will activate CS1 and CS3. Each store sends the answer on a separate bus, where central control match circuits can check for a fault in communication or memory action.

Routing of answers from the G and H halves is controlled by four answer flip-flops in each store. The answer flip-flops can be set in any combination, so that either half may answer on either, neither, or both buses. Answer flip-flops, like the variable-name flip-flops, can be set in a 5.5- $\mu$ sec cycle by a control mode order from the central control.

### 5.3 Bus Receivers and Registers

All inputs to the call store are routed to register flip-flops, and the essential memory circuits use the semi-de outputs of these registers.

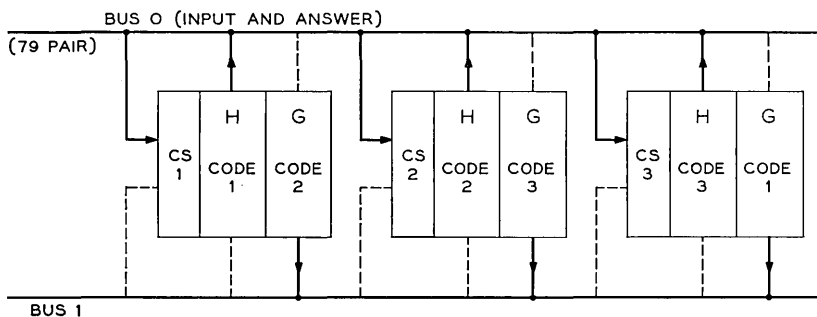


Fig. 19 — Call stores on bus.

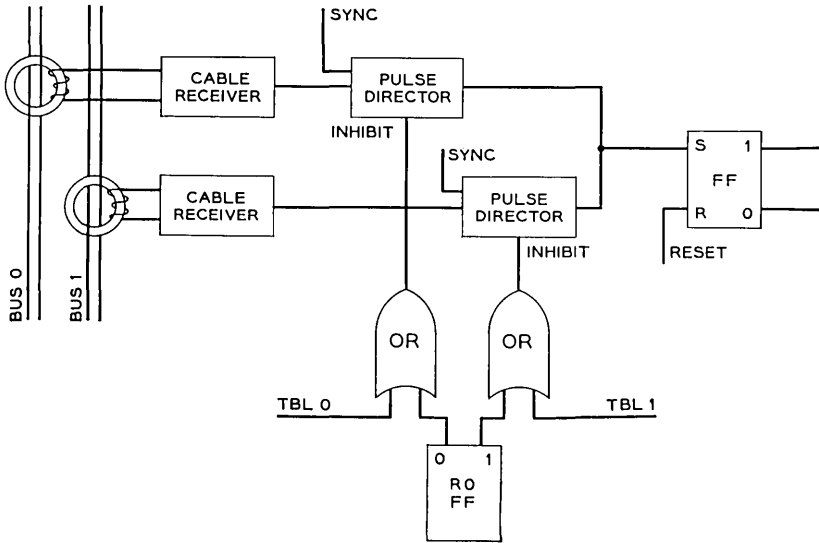


Fig. 20 — Bus input path.

Fig. 20 shows a typical input path. The inputs from the separate buses are transformer coupled to cable receivers and transmitted to "pulse director" circuits. The pulse directors gate the inputs with their associated sync pulse and may inhibit the input. Bus selection is made by inhibiting the unselected bus input at the pulse director. Also shown in the figure are the TBL0 (trouble on bus 0) and TBL1 inhibiting functions. These leads are controlled by a separate communication channel and can be used to shut off inputs from either or both buses.

The outputs of the pulse directors are logically added to set the register flip-flops. All inputs, code, mode, address, order, and data are single-rail, and the register flip-flops are reset by internally generated pulses at the end of each cycle.

The diagram in Fig. 21 gives the complete contents of the input and answer buses along with the nominal timing.

Each wave of information on the input or answer bus is accompanied by a time-coincident sync pulse. The receiving circuits require the coincidence of the sync pulse and the information to protect against false operation on bus noise.

#### 5.4 Modes

In the "normal" mode two call stores respond as a regular memory to every order. Under trouble conditions it is necessary to communicate

with a single store in order to determine the location and nature of the trouble. For this purpose, two maintenance modes are used in which the call store responds to a code match with the fixed name only. Thus two maintenance modes are necessary to define the H or G half of a given store. These are designated "H maintenance mode" (HMM) and "G maintenance mode" (GMM). The call store operates as a normal memory in the HMM and GMM; however, only the fixed name is used for selection and the store answers and receives on the same bus.

The fourth and final mode of operation is the control mode. Here the call store does not operate as a memory — no ferrite is switched. In the control mode, the write operation sets flip-flops or initiates conditions in the call store and the read operation sends the state of flip-flops or other test points to the system via the answer bus. For example, setting the 6 variable-name and 4 answer flip-flops is a control mode, write operation. The first 20 bits of the 24 data-input leads carry the set and reset information for the 10 flip-flops. A corresponding control mode read operation returns the state of the variable name and answer flip-flops to the central control on the normal answer bus.

### 5.5 Orders

The read and write orders are received in a redundant code over two bus leads. In the normal and maintenance modes,  $R,W = 10$  is a legitimate read and  $R,W = 01$  a legitimate write. The inputs  $R,W = 00$  or  $11$  are trouble conditions; however, drive currents have started before the

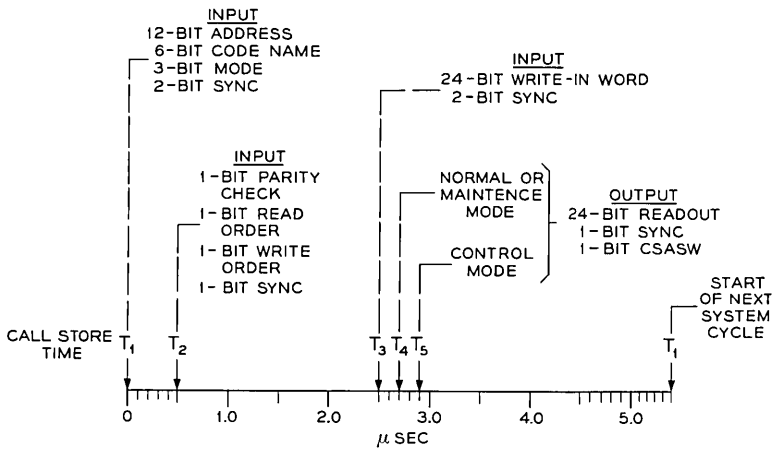


Fig. 21 — Timing of input-output signals.

order is decoded and cannot be interrupted. Illegitimate orders are therefore converted to read orders and no ASW is returned with the answer. A good write order,  $R, W = 01$ , is also converted to a read order if the ASW circuits detect any trouble. Thus information is safeguarded by forcing a regenerate sequence if any ASW failure is detected.

In the control mode all four combinations of the read-write order leads are used. Control read and control write operations are used, and the corresponding order bits  $R, W = 10$  and  $01$  are required. The control functions are further defined by a control address which uses the four least significant address bits. To test the control address circuits themselves, however, it is desirable to have a control order that is independent of address so that all combinations of address inputs can be tested. To this end, the normally invalid order combinations  $R, W = 00$  and  $11$  are used to read test points associated with the addressing circuits. These special orders check the  $Y$  and  $X$  portions of the address registers as well as the dc switches in the memory access circuits.

### 5.6 Addresses

In the memory modes, the 12-bit call store address defines one word in the half-store, 4096-word memory block. This block is generally visualized as a square  $64 \times 64$  array in an  $XY$  plane. The six least significant bits constitute the  $Y$  address. The memory access system described earlier further divides the six  $Y$  bits into two groups of three bits each. Each three-bit group is decoded into a one-out-of-eight selection by the dc switches. Selection of the  $X$ -drive switches is similarly derived from the six most significant address bits.

In the control mode the address bits select one of the possible control operations in a manner analogous to the selection of a memory location. Since only a few control locations exist, only the four least significant address bits are needed for control addresses. For example, the name and answer flip-flop control address is octal 3. Therefore (see Fig. 19) if we send on bus 0: control mode, code = 3, address = 3, and order = read, the right-hand call store will reply that its variable name is octal 1 and that its answer flip-flops are set to return H-half answers on bus 0 and G-half answers on bus 1.

### 5.7 Data Channels

Call store data are handled by 24 identical channels, one of which is shown in Fig. 22.

During a read order in the normal or maintenance mode the output of

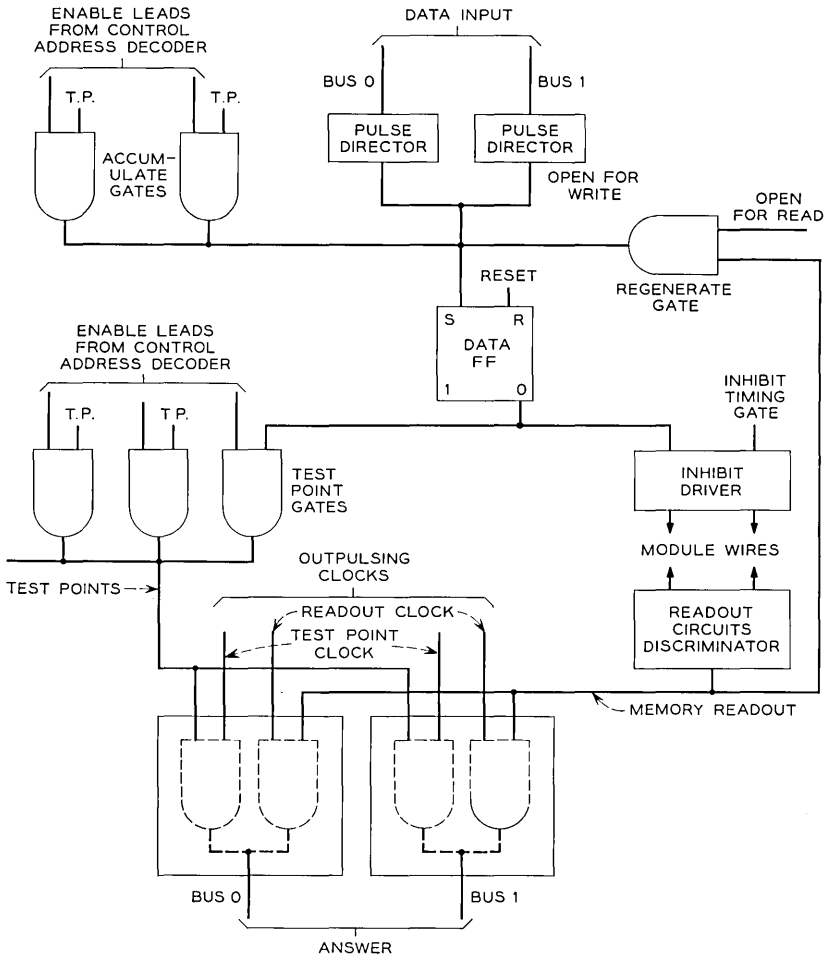


Fig. 22 — Data channel.

the discriminator is gated onto the selected bus by a readout clock pulse. If the mode is normal the four answer flip-flops select the answer bus and either, neither, or both buses could be pulsed. In addition, on a read order the regenerate gate is open and the discriminator output sets the data register. If a 1 is read from memory the data register flip-flop will be set, the associated inhibit generator will not fire, and a 1 will be rewritten in the memory during the write portion of the call store cycle. When a 0

is read out from memory the data flip-flop remains reset, the inhibit generator fires, and a 0 is written in that bit.

For a write order in the GMM, HMM or normal mode the sequence is the same except the regenerate gate is closed, the input data pulse director is enabled, and the data register is set or not set from the bus. On a write order the read part of the call store cycle is used to erase existing information.

Both input data and readout are single rail; the data registers are reset by an internally generated pulse. One of the data register reset pulses is generated at the end of the write-back drive current, 0.5  $\mu$ sec before the end of the inhibit gate. Therefore all data flip-flops are reset while the inhibit gates are still open, and as a result, all inhibit drivers fire, producing a post-write disturb signal.

In the control mode the data channels are used to transmit test point information to the central control. The flip-flops themselves serve as pulse accumulating registers in two control read cases.

For semi-dc conditions that exist throughout a cycle, or at least exist when the output is gated on the bus, the test point gates are used. Pulses that occur at odd times during the cycle are checked by first giving a control write order to accumulate the pulses in the data register. This enables one of the accumulate gates shown on Fig. 22. The timing pulses, if and when they occur, set the data register flip-flops, whose contents can be read out if the next order is a data register control read.

## VI. MAINTENANCE CIRCUITS

### 6.1 *General*

The maintenance circuits are designed to accommodate a system philosophy of fault recognition and trouble location quite different than those used in most information processing machines. The prime objective is to maintain the continuity of system operation. Thus the first action is to locate the faulty store and remove it from service. Detailed diagnosis which will isolate the fault to a specific plug-in package can then be accomplished on a deferred or low-priority basis. The possibility of 40 or more call stores in an office accents the demand that diagnosis be automatically programmed and as complete as possible.

Flexible switching provides rapid rearrangement of the bus configuration, and the variable-name facility allows the location of memory blocks to be reassigned to retain a duplicate copy of priority information. Because of the complete information duplication and the flexible communication, the system can survive even if half of the stores fail.



The existence of a trouble is generally noted by mismatch between duplicate stores, parity failure, or failure of "all-seems-well" (ASW). Mismatch and parity are clearly over-all checks that include the entire communication link between central control and the stores. Furthermore, they indicate trouble on the bus or in a complete call store rather than detecting failure of some circuit within a store. The ASW circuits do detect specific conditions within a unit, but the system's use of the ASW pulse does not elaborate on the nature of the failure; it merely specifies which store is in trouble.

Table I lists the maintenance facilities in a call store along with their primary functions.

The duplicate bus answer matching as well as the input and answer bus selection were described previously. These insure alternate routing to carry on communication in spite of failures and allow cross checks between channels to weed out faulty links. The maintenance mode orders, which enable the system to choose specific addresses in specific stores, have also been described. The functions of the maintenance mode orders as well as the remaining maintenance facilities are discussed in the following sections.

### 6.2 *All-Seems-Well (ASW)*

There are six conditions tested by the ASW circuit:

- (1) The order combinations  $R, W = 00$  and  $11$  cause ASW failure. In the normal mode they indicate trouble. In the control mode, these combinations are legitimate but they still cause an ASW failure. However, this failure is intentional and is used to test the all-seems-well circuit.
- (2) Parity failure causes ASW failure.
- (3) An invalid mode input causes ASW failure.
- (4) Simultaneous operation in the control mode and a memory mode causes ASW failure.

TABLE I — CALL STORE MAINTENANCE FACILITIES AND FUNCTIONS

All-seems-well (ASW)	}	fault recognition in communications
Address parity (over code and address)		
Duplicate bus answer matching	}	over-all fault recognition
Stored parity (code, address and data)		
Input and answer bus selection	}	communication diagnosis
Fault flip-flops		
Control mode orders	}	logic diagnosis
Maintenance mode trouble location		
Monitor bus	}	memory circuit diagnosis
Scan points and central pulse distributor		
		store status test

(5) Simultaneous operation in both the G and H halves of the store causes ASW failure.

(6) A permanent code name match causes ASW failure.

The all-seems-well circuit makes all six tests during every call store operation, normal or otherwise. If there are no irregularities this circuit returns an ASW pulse along with the readout onto the answer bus. However, if a trouble is detected the store answer is transmitted without the ASW pulse and the call store is forced into read and regenerate operation regardless of the input order. This restriction on memory writing protects information from being destroyed when the incoming orders are incorrect and is particularly useful with address parity error detection.

### 6.3 *Fault Flip-Flops*

Five fault flip-flops are included in a call store to detect and register the occurrence of intermittent or transient faults. Either maintenance programs or a maintenance man using teletypewriter control can use the fault flip-flops to locate the general area of intermittent or marginal troubles.

The fault flip-flop indications provided are:

- (1) all-seems-well failed,
- (2) both the G and H halves were selected simultaneously,
- (3) the store tried to operate in the control mode and a memory mode simultaneously,
- (4) test point data were sent along with memory readout on the answer bus, and
- (5) the code match circuit has a permanent decoder output (PDO).

Item (5) is particularly troublesome. A PDO means that a store will answer an order with any input code and mask the bus with unwanted signals. If PDO is detected, a call store will set both of its own trouble flip-flops, thus taking itself out of the system. This is the only case where the call store tampers with system status or bus selection.

### 6.4 *Monitor Bus*

The monitor bus is an 8-pair bus common to all call stores, on which test points may be connected through relay contacts to a ferrod scanner. In the call store the monitor bus is used exclusively to test regulated voltages. The outputs of two voltage regulator boards of the same type are connected through current limiting resistors to opposite sides of a bus pair, and the ferrods detect current flow if the output voltages are different and out of tolerance.

### 6.5 *Address Parity and Stored Parity*

Every memory mode order (normal or maintenance mode) includes a parity bit over the code and address. This parity is checked in the call store. Failure of parity causes failure of ASW and will convert a write order to a read order. Central control is informed of the trouble when it fails to receive the ASW answer pulse and can take corrective action. Since the write order is not executed, information in the incorrectly selected address is not destroyed.

Another parity bit is transmitted to the call store during memory write orders. This parity is over code, address and data. It is stored in memory as the 24th bit and returned to central control when the word is read. This parity is not processed in the call store and is treated as any other data bit. However, central control interprets the parity and checks for errors.

### 6.6 *Maintenance Mode Trouble Location*

Diagnosis of the memory module and its associated circuits is primarily done by storing and reading information. The faulty package is located by noting at what addresses storage fails and/or in which data channel. For example, a bad diode matrix will cause errors along one  $X$  or one  $Y$  address. A bad input marker will cause 8 consecutive addresses to fail. Bad output markers cause every 8th address to fail.

To increase the selectivity of this type of diagnosis, associated circuits are packed on the same printed wiring boards, and common operations are grouped to cause recognizable failure patterns. As an example of the latter statement, the 24-bit data register is reset in groups of 12, the inhibit generators are pulsed in groups of 8, and the regenerate gate and input pulse directors are gated in two groups of 8 and 16.

### 6.7 *Scanner Points and Central Pulse Distribution*

The bus routing  $R_0$  flip-flop, the two error status trouble flip-flops, and the code match indication flip-flop have a very high degree of influence upon call store operation. Therefore the indication of state of these flip-flops is not trusted to the regular bus communication system. Instead the outputs are directly connected to a direct reading ferrod scanner.

The  $R_0$  flip-flop and the two trouble flip-flops are so critical that separate system control of the setting or resetting commands are required.

In these cases the set and reset inputs are directly connected to a central pulse distributor which can alter the flip-flop state.

### 6.8 *Control Mode Orders*

The control mode operations complete the survey of maintenance features. These are essential for automatic programming diagnosis through the central office facilities. They allow the system to interrogate the inner core of a call store; many circuit failure conditions do not have to be interpreted by evaluating memory readout, but can be directly tested. A test such as the "read  $Y$  dc switches" sends back onto the answer bus the binary condition of key circuits in the  $Y$  access, which indicates the contents of the  $Y$  section of the address register and the  $Y$ -access decoding circuits.

### 6.9 *Control Mode Operations*

- (1) Write data register,
- (2) read data register,
- (3) write name and answer flip-flops,
- (4) read name and answer flip-flops,
- (5) write — set current correcting servo counter to all 1's,
- (6) write — reset servo counter to all 0's,
- (7) read servo counter test points,
- (8) read fault test point group,
- (9) write — accumulate timing pulse group 1 in data register,
- (10) write — accumulate timing pulse group 2 in data register,
- (11) read  $Y$  dc switches,
- (12) read  $X$  dc switches.

## VII. EQUIPMENT

Equipment design of the call store consists of a highly functional layout in which the ferrite sheet memory modules, located behind a shield below the center of the frame (see Fig. 23), are surrounded by the access circuits at the sides and readout and inhibit circuits above and below. This permits the necessary short paths to be maintained for the readout connections while minimizing interference from the access and inhibit circuits. The logic control circuits and cable drivers are located near the top of the frame, below the terminal strips with which they connect. As indicated in the figure, the call store follows the general pattern of equipment design for No. 1 ESS.<sup>7</sup> With only a few exceptions, components are mounted on plug-in circuit packs. External connections

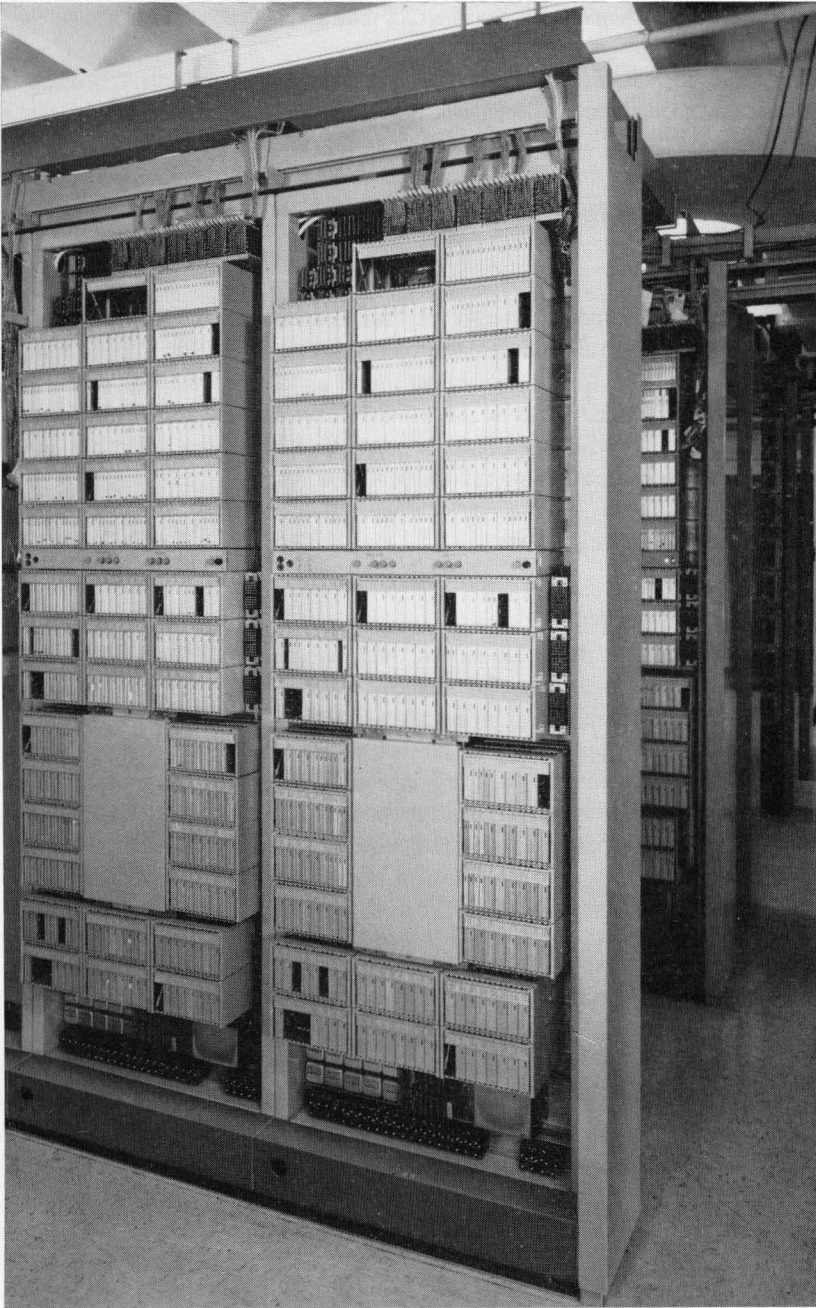


Fig. 23 — Two call stores in a No. 1 ESS installation.

are made with terminal strips and transformers at the top of the frame, while fuses, filters and other power control equipment are at the bottom. The control panel is of standard construction at a convenient height 52 inches above the floor, which is standard for No. 1 ESS. Connectors below the control panel, at the right, are for test purposes.

#### VIII. POWER

The entire external power supply for the call store comes from +24- and -48-volt central office battery. Emergency limits during commercial ac charging power failure can be as much as 20 per cent below the float voltages. This variation is accommodated by all circuits in the store.

Average power dissipation in a continuously addressed store is 470 watts. Unselected stores on the bus use 360 watts.

#### IX. OPERATING EXPERIENCE

##### 9.1 *General*

Experience with ten call stores over a combined operating time of 80,000 hours has been very favorable.

The large number of call stores to be installed in the many No. 1 ESS installations dictates complete circuit pack interchangeability without any final adjustments in working stores. To achieve this goal most circuits have been designed without potentiometers and yet maintain tight tolerances. Some packs, notably regulators, delays, current drivers and servo reference circuits, require potentiometers to compensate for semiconductor junction differences and initial parameter differences, but they are set at the factory and will not be adjusted in individual stores.

Temperature effects on the module over the range of ambients from 32°F to 115°F, although quite pronounced, are corrected through the joint action of temperature-sensitive regulators and the servo system to keep the memory output signals well within the margins of the decision circuits. The concurrent change in switching time for the ferrite material with change in temperature is adequately compensated by deriving strobe timing from the switching signal of a standard ferrite sheet.

Although the combined effect of component drift, temperature variations and reasonable circuit pack misalignment does reduce 1 to 0 output signal separation somewhat over the ideal case, the most severe effect is the delta noise generated by the particular pattern of information stored in the memory.

## 9.2 *Delta Noise Considerations*

### 9.2.1 *General*

No single information or addressing test pattern suffices to simultaneously tax both the memory and the circuits. Furthermore, it is desirable to facilitate circuit design by separating effects and measuring their relative magnitudes. To achieve this goal, random-access, random-content memories must be exercised by a series of different programs, each generating a different type of worst case. The accumulated results of all these tests are combined to calculate the design margins of the memory. The call store must have extremely good margins because of the intended 40-year life. The huge number of call stores and the vast complex of different program combinations in the many telephone offices throughout the country almost guarantee that any remote combination that is possible will occur. For this reason the call store performance must not be address-sequence dependent or information-content dependent; it must be impervious to any possible normal sequence of events.

### 9.2.2 *Delta Noise Definition*

The paramount problem in design of coincident-current memories is the delta noise due to accumulation of half-current selected memory locations. Consider first two wires through a single hole in a ferrite sheet (or through a single core). If a half-drive current were passed through one of these wires, it would develop a relatively large signal across the other winding by direct coupling. Now if we consider two windings through a pair of holes, the situation is quite different. The readout winding can be threaded through the two holes so that it couples the drive winding in the opposite sense at each hole. A half-drive current through the pair of holes would induce signals of opposite polarity in the readout wire, and the net signal would be zero. From this it follows that along any memory drive wire there must be an even number of holes (or cores); half of these holes must couple the readout wire with a positive polarity, while the other half couple the readout wire with a negative polarity. In a two-dimensional coincident-current memory both axes must satisfy the above condition. Therefore, it is convenient to think of the holes along a drive winding as forming "canceling pairs." For example, a square array of 1024 holes in a  $32 \times 32$  coincident-current memory plane would have 16 canceling pairs in each dimension.

At this point delta noise will be defined as the summation of the mag-

netic unbalance in canceling pairs due to the half-read drive currents. The large capacitively coupled signals are not as important, because very good cancellation can be achieved by proper wiring techniques and common mode noise rejection techniques. Magnetic unbalance can exist, however, and it is important to consider it from two separate sources.

The output signal from a canceling pair due to a half-drive current is zero if (1) both holes are in the same state (i.e., both 1's or both 0's) and (2) both holes have had the same past history of disturbances. The past history that is important is the polarity of the last half drive, either half read or half write. Condition 1 in the above statement depends only on information contents stored in the memory plane. Delta noise, arising when this condition is violated, will be called pattern noise or  $\Delta P$ . The past history of disturbances is a function of the sequence of operations and addresses used in the memory and will be called sequence noise or  $\Delta S$ .

### 9.2.3 Hysteresis Model

The 4-state hysteresis diagram shown in Fig. 24 illustrates sequence and pattern delta noise. This model assumes that a single half read will walk down a 1 hole from state 1W to 1R or a 0 hole from 0W to 0R. Further half-read pulses will not change the conditions, but a single half write will restore a 1 hole to 1W or a 0 hole to 0W. Experimental studies on small groups of cores, single ferrite sheets and the complete ferrite sheet module indicate that this model and its assumptions are very good. The different conditions of canceling pair noise and their relative amplitudes are given in Table II.

Pattern noise which is due to the unbalance of 1R and 0R pair depends on the information content of the memory and very little can be done about it. Asymmetric drive currents with write current larger than read current reduce pattern noise over the symmetrical-drive case, but it cannot be eliminated. The asymmetric drive apparently distorts the  $B$ - $H$  curve and tends to equalize the slope of walked down 1's and 0's. This effect is limited by partial switching due to the oversize write half drive and inhibit, which can deteriorate storage.

Sequence noise, on the other hand, can be eliminated by providing a uniform "last" operation for all holes. The most popular method of doing this is to use a "post-write disturb" pulse. Generally, post-write disturb is accomplished by firing all inhibit drivers after writing is completed. The inhibit drive is equivalent to a half read; therefore, all holes would be in the 0R or 1R state of Fig. 24. The cure has its cost, how-



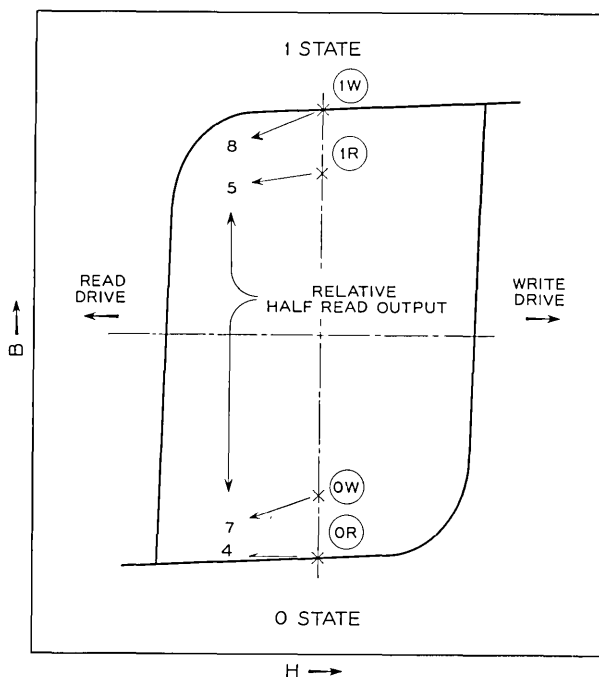


Fig. 24 — Four-state model B-H curve of ferrite sheet.

ever, in cycle time. Inhibit post-write disturb is particularly inefficient, because generally the inhibit winding is very long and has a high inductance. This makes it difficult to produce fast rise and fall on the post-write disturbs. The walk-down phenomenon, on the other hand, is very fast. Unlike hole switching, which for ferrite sheets requires 1  $\mu$ sec, walk-down from 0W to 0R or 1W to 1R is accomplished during drive current rise time alone.

Sequence noise requires consideration, even though a remedy is at

TABLE II — CANCELING PAIR STATES AND RELATIVE DELTA NOISE AMPLITUDES

State of Holes (Canceling Pairs)	Delta Noise	Relative Amplitude (16 Canceling Pairs)
0R vs 0R	none	1.6
0R vs 1R	$\Delta P$	16
0R vs 0W	$\Delta S$	83
0R vs 1W	$\Delta P + \Delta S$	100

hand, because sequence noise represents energy in the ferrite material, and if this energy is transferred by a post-write disturb or any other disturb, it must be provided for in the readout circuits. In other words, a post-write disturb will generate an output signal equal to the noise condition it corrects, and this signal must be dissipated before the next readout appears.

#### 9.2.4 Delta Noise Test Patterns

The basic test pattern for studying delta noise in a two-dimensional coincident-current memory is the pattern which writes all positive readout holes to the one state and the equal number of negative readout holes to the zero state. Fig. 25 shows a  $32 \times 32$  bit plane pattern that satisfies this requirement. It also represents the worst-case delta noise pattern for call stores, because the duplication of readout and  $Y$ -axis system reduces the plane of intersection of the  $X$ - and  $Y$ -access wires with a given readout to a 32-by-32 array.

The figure does not represent the relative location of holes on any physical plane. The  $X$ -drive wire, for example, may thread alternately through positive and negative holes, or through any grouping of positive and negative holes. Translation of the address assignments is necessary to produce the four-quadrant picture; any coincident-current memory can be represented in this way.

Ignore for the moment any sequence effect and suppose the pattern of Fig. 25 is continuously repeated. Each read does indeed include the

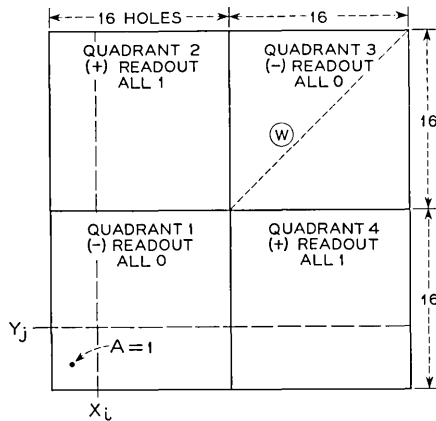


Fig. 25 — Address plane test patterns.

worst case of pattern-dependent delta noise ( $\Delta P$ ), but the noise component is always in the direction to enhance the signal separation by increasing 1's and decreasing 0's. For example, in quadrant 1, the 0's are negative and  $\Delta P$  is positive; in quadrant 2, both the 1 readout and  $\Delta P$  are positive.

To measure pattern-dependent delta noise that degrades the 1 to 0 ratio with consistency over every hole in the memory, a double-operation checkerboard pattern can be used as follows. Scan the worst-case delta noise pattern (which is a checkerboard) sequentially with two operations on each hole. First read a hole and write it to the opposite state. Second, read the hole and write it to the original state. On the second read, delta noise will be of a polarity to degrade the signal.

### 9.2.5 Sequence Patterns and the "Worst" Worst Case

Measurement of sequence-dependent delta noise not only adds complications to the testing procedure but also creates some pitfalls that must be avoided. The great opportunity to err comes from the fact that the polarity of the sequence-dependent delta noise can be independent of pattern; therefore it is possible to cancel sequence noise against pattern-dependent noise and obtain measurements that do not reflect the worst case of anything. For the call store the following statements apply:

Pattern-dependent delta noise from a 1,0 canceling pair will have the same polarity as the 1 hole.

Sequence-dependent delta noise from a half-read, half-write canceling pair will have the same polarity as the half-written hole.

To generate the "worst" worst possible delta noise on a test hole A the following operations are necessary:

- (1) Write the basic worst-case pattern as shown in Fig. 25.
- (2) Write the single test hole A to the opposite polarity. In the figure, A is shown in a quadrant of 0's, so it must be written to 1.
- (3) Write a 0 somewhere in the memory. This causes an inhibit current which half reads all holes in the bit plane and neutralizes the effect of writing the 1 in test hole A.
- (4) Write 1's along the diagonal W. This makes the "last" operation on every hole in quadrants 2, 3 and 4 a half write. The "last" operation on all holes in quadrant 1 is still the half read due to the previous inhibit pulse of step (3).
- (5) Readout test hole A. The readout will be a negative 1.  $\Delta P$  and  $\Delta S$  will both be maximum and positive.

The zero "worst" worst case can be generated in a similar fashion.\*

### 9.2.6 Results

Testing for "worst" worst case delta noise is extremely difficult, because practical test equipment becomes very complex. However, a partial implementation of this condition has been realized with the "worst" worst case along one axis and the worst case along the other axis. Measurements made on 10 stores over a temperature range of 32–115°F, with currents and circuit parameters perturbed within their expected life variations, have indicated signal-to-noise ratios in the order of 3 to 1.

Using post-write disturb to neutralize sequence noise, the 3-to-1 signal-to-noise ratio or, more accurately, 1-to-0 ratio, is measured with the double-operation worst-case checkerboard described in Section 9.2.4 and the drive currents at the extremes of the servo control ( $\pm 2$  per cent). The measurements are made by changing the discriminator level reference voltage until a 0 is detected as a 1 and until the first 1 is detected as a 0. The nominal value for the discriminator reference is 19 volts. Typical measured signal separations range between 6 volts to 26 volts and 10 volts to 29 volts. The major factors in separation variations are the memory modules themselves and the unclamp and strobe timing differences due to the variability in logic circuit delays.

## X. SUMMARY AND CONCLUSIONS

Using ferrite sheets with relatively high Curie temperature, an economical, fast, high-capacity temporary memory unit capable of operating over a 32–115°F temperature range has been achieved.

Many novel circuit features are incorporated to fully exploit the inherent device capabilities.

To achieve the high degree of dependability essential in an electronic switching system, it contains numerous maintenance facilities as well as provisions for rapid reassignment of memory function by means of a flexible intercommunication system.

The experiences with ten stores, in operation for more than 80,000 memory hours during the past year, have proven that the stringent design objectives necessary for reliable and dependable performance have been achieved.

---

\* +0 with  $+\Delta P$  and  $+\Delta S$  is impossible to generate. -0 with  $+\Delta P$  and  $+\Delta S$  is the worst practical case.

## XI. ACKNOWLEDGMENTS

The authors wish to acknowledge the contributions of many others to this development, particularly those of E. Ley for the store mechanical design, C. W. Thulin for his efforts in the ferrite sheet module development, J. E. Mack for his part in the early phase system design, E. H. Siegel for his novel circuit designs, C. G. Corbella and R. W. Rolund for their accurate analysis and evaluation, A. A. Stockert for his work on maintenance aspects, and R. W. Ketchledge for his counseling and guidance during the development.

## REFERENCES

1. Harr, J. A., Taylor, F. F., and Ulrich, W., Organization of No. 1 ESS Central Processor, B.S.T.J., this issue, p. 1845.
2. Downing, R. W., Nowak, J. S., and Tuomenoksa, L. S., No. 1 ESS Maintenance Plan, B.S.T.J., this issue, p. 1961.
3. Connell, J. B., Hussey, L. W., and Ketchledge, R. W., No. 1 ESS Bus System, B.S.T.J., this issue, p. 2021.
4. Meinken, R. H., Ferrite Sheet Memory, Proc. 1960 Electronic Components Conference, Washington, D.C., May 10-12, 1960.
5. Meinken, R. H., A Memory Array in a Sheet of Ferrite, Proc. Magnetism and Magnetic Materials Conference, October 16-18, 1956, p. 674.
6. *Digital Applications of Magnetic Devices*, ed. A. J. Meyerhoff, et al., John Wiley and Sons, New York, 1960, pp. 372-390.
7. Ferguson, J. G., Grutzner, W. E., Koehler, D. C., Skinner, R. S., Skubiak, M. T., and Wetheroll, D. H., No. 1 ESS Apparatus and Equipment, B.S.T.J., this issue, Part 2.