Like the single thread which showed the way through a maze in a classical tale, the stored program of No. 1 ESS guides telephone calls through the labyrinth of control equipment and switching equipment in a central office.
A anybody picking up and dialing a telephone at this moment starts a chain of switching actions that links his phone through a unique voice pathway to any one of the 88 million telephones in the United States. A million people may pick up their phones at the same time with similar results. If a few hundred of these people are served by the same central office, their calls are handled side by side although the connections for each one may be directed by separate groups of control equipment.

From the customer's point of view, No. 1 ESS will handle ordinary telephone calls in the same way an electromechanical office handles them. They will seem to go through side by side, and the actions set in motion when a call is originated will continue in an apparently unbroken sequence until the connection is made. Actually, No. 1 ESS works on only one call at a time; its enormous operating speed makes it appear to handle all calls simultaneously. Furthermore, while in an electromechanical system each action in the switching sequence triggers the action that follows it, in No. 1 ESS the program is the trigger. It may direct the system to send dial tone to a telephone originating a call, then make a connection to the switching network for a second call, go on to take down connections for a third call, and only then return to the first call to receive dialed digits. Thus any step in processing a single call is isolated, so to speak, from any other.

The actual circuit actions that accomplish each step in the processing take place in central control. Every 5.5 microseconds one of the 100,000 instruction words that make up the stored program directs central control in some basic action of call processing or automatic maintenance. The flow of actions follows a precise schedule because the program actually controls the system. Each specialized action outlined by a functional program, the executive control program, which decides when each of the first four are to be called into operation.

To smooth the flow of the whole process, the call store memory acts as a clearing house of information between the functional groups. The call store is divided into many sections, each with a number of words, or memory slots, called registers, hoppers, and buffers, in which information is deposited and withdrawn as it is needed. One or more registers is associated with each telephone call being processed. Input programs fill the hoppers with input information that is operated on by the operational programs. These, together with the subroutines, stock buffers with output information for the output programs. Thus the processing of any call is conducted in play between the program store and the call store.

A familiar starting point for describing how a switching system handles a call is the custom of lifting his telephone handset from its cradle. In No. 1 ESS, as in all other systems, that act signals the origination of a call. But to grasp what happens next, we must examine this information and decide what, if any, output actions are required in response to it. The operational programs also call on subroutines to fetch data that the output programs will use. Finally, the output programs make and release connections in the switching network and operate other circuits. During the time that central control is involved with the specialized actions outlined by a functional program, the executive control program, which decides when each of the first four are to be called into operation...
tral control to scan all ferrods. The object is to discover which ferrods have changed state since the last scan, 200 milliseconds before. Each time it discovers a change, the program temporarily stops the scanning action and writes the equipment number of the originating line in a hopper reserved for service requests. The system always completes the immediate action on one line before it goes to the next. Thus, though it will take the same action in all scanning times, it works on only one at a time. When all the ferrods have been scanned, the scanning program returns control to the executive program.

At this point, any line that has gone off-hook since the last scan is identified in the service request hopper. The executive program now decides on the next action. To take further action on the service requests stored in the hopper, it must schedule the input detecting program. But it may decide instead to give control to an output program; for example, to operate the line switching network. A single network control device can be used only once in about every 20 milliseconds. But to preclude any possible "traffic jam," the program directs any controller to operate only once every 25 milliseconds and in the interval directs actions elsewhere in the system. Meanwhile, the service requests wait in the input detecting program. Thus, the initial step in the call may have no direct connection in time with the next step. This may also be the case with any other sequential steps.

The drawing on page 216 traces all the program steps that occur as a call progresses from its initiation to a talking connection. As the call moves through the office, control is handed back and forth between the executive control program and the functional programs. A functional program may direct the executive control program to perform a specified task, and then return control to the executive program. The latter may direct the functional program to perform a specific action, such as dialing a number or waiting for a reply to be received. Meanwhile, the executive control program may also be processing other calls in the system, and in the interval directs actions elsewhere in the system. The executive control program is the "master of all the functional programs, the actual job of the task dispatcher is to examine the data stored in the buffers and to control the execution of the functional programs that are associated with the task.

The executive control program schedules the functional programs in a priority sequence, which is determined by the type of service requested and the urgency of the call. The sequence is triggered by the 7-bit revolving counter, which is executed every 45 milliseconds and a binary 1 appears at the leftmost position every 45 and every 60 milliseconds.

The handling of the input programs underlines an idea that is central to the method of processing telephone calls: Certain programs can be delayed while others are executed. Simple as this idea seems, the specialization in the handling of input and output programs must be executed in real-time, service requests must be dispatched with the least possible delay. Programs such as routine maintenance, on the other hand, can be deferred to slack traffic hours. All programs, therefore, are woven into a hierarchical order based on their relative importance.

A basic priority order is assigned to "task dispatcher" programs, specialized programs associated with specific functions in the call. The task dispatchers are active links between the executive program and the "task" programs that carry out the processing routines in the data buffers. Because the buffers accumulate information of varying degrees of importance, they must be unloaded according to a strict priority scheme. Therefore, the task dispatcher programs are grouped in six classes. The highest priority is class A, followed by classes B, C, D, and E. Each class is run more frequently than the class below it, and the class below it is run more frequently than the class directly below it. The highest priority, class A, is run most frequently, with the lowest priority, class E, run least frequently. The priority order is determined by the type of service requested and the urgency of the call. The sequence is triggered by the 7-bit revolving counter, which is executed every 45 milliseconds and a binary 1 appears at the leftmost position every 45 and every 60 milliseconds.

The handling of the input programs underlines an idea that is central to the method of processing telephone calls: Certain programs can be delayed while others are executed. Simple as this idea seems, the specialization in the handling of input and output programs must be executed in real-time, service requests must be dispatched with the least possible delay. Programs such as routine maintenance, on the other hand, can be deferred to slack traffic hours. All programs, therefore, are woven into a hierarchical order based on their relative importance.

A basic priority order is assigned to "task dispatcher" programs, specialized programs associated with specific functions in the call. The task dispatchers are active links between the executive program and the "task" programs that carry out the processing routines in the data buffers. Because the buffers accumulate information of varying degrees of importance, they must be unloaded according to a strict priority scheme. Therefore, the task dispatcher programs are grouped in six classes. The highest priority is class A, followed by classes B, C, D, and E. Each class is run more frequently than the class below it, and the class below it is run more frequently than the class directly below it. The highest priority, class A, is run most frequently, with the lowest priority, class E, run least frequently. The priority order is determined by the type of service requested and the urgency of the call. The sequence is triggered by the 7-bit revolving counter, which is executed every 45 milliseconds and a binary 1 appears at the leftmost position every 45 and every 60 milliseconds.

The handling of the input programs underlines an idea that is central to the method of processing telephone calls: Certain programs can be delayed while others are executed. Simple as this idea seems, the specialization in the handling of input and output programs must be executed in real-time, service requests must be dispatched with the least possible delay. Programs such as routine maintenance, on the other hand, can be deferred to slack traffic hours. All programs, therefore, are woven into a hierarchical order based on their relative importance.

A basic priority order is assigned to "task dispatcher" programs, specialized programs associated with specific functions in the call. The task dispatchers are active links between the executive program and the "task" programs that carry out the processing routines in the data buffers. Because the buffers accumulate information of varying degrees of importance, they must be unloaded according to a strict priority scheme. Therefore, the task dispatcher programs are grouped in six classes. The highest priority is class A, followed by classes B, C, D, and E. Each class is run more frequently than the class below it, and the class below it is run more frequently than the class directly below it. The highest priority, class A, is run most frequently, with the lowest priority, class E, run least frequently. The priority order is determined by the type of service requested and the urgency of the call. The sequence is triggered by the 7-bit revolving counter, which is executed every 45 milliseconds and a binary 1 appears at the leftmost position every 45 and every 60 milliseconds.
The sequence moves to the next class.

Interject programs may be “interjected” at any point in the sequence. One example, is the 1000 millisecond timing program which keeps a record of the time and the date and maintains a 10 × 100 millisecond timetable. This program is run every 100 milliseconds on an interject request from the 20 × 5 millisecond timetable. A program schedule in the 10 × 100 millisecond timetable may be executed when it becomes due (“times out”), or it may be rescheduled at a lower priority by writing a binary 1 (called setting a flag) in certain control registers. These flags direct the executive control program to execute specified task dispensers or to consult timetables that have periods larger than one second. This method of cascading timetables allows the executive control program to schedule programs that repeat as often as every 100 milliseconds or as seldom as once a week.

All the individual threads of the program that we have discussed are woven together in the call store memory. Every call, in its progress through the office, is assigned to a call store register which consists of memory slots for the temporary storage of input and control data. (See the drawing on this page.) Some stages of a call require more memory space than others. During digit scanning up to ten digits may be stored, for example. Thus, there are different sized registers for different stages of a call. However, the program subroutines handle all registers in the same fashion because the first three words of all registers, called the state word, the queue word, and the link word, respectively, are alike.

The state word identifies the register by its function—originating, outpulsing, disconnecting, etc.—and its particular state. The latter is shown by an index called the program tag. When the register receives an input from a task dispenser, the state word selects a group of four task programs appropriate to the particular stage of processing. The exact program is selected from the four by reference to the type of task dispenser that delivered the input entry. Thus, instead of following a complex branching process to narrow down the choice of a program by many either-or decisions, the register simply looks for the proper program in a table.

The state word selects a group of four task programs appropriate to the then current state of the call. The link word links the register to a waiting queue when needed peripheral equipment, such as a ringing trunk, is tied up on another call. In this case, when the trunk becomes available, the queue administration program assigns the equipment to the register on the queue and transfers control to the task program selected by the state word.

The link word links a number of registers together if more than one is required for a call. This may occur, for example, on a call that is transferred to a remote station outside the central office. At one stage in this procedure an originating register, a temporary transfer register, and an automatic message accounting register are linked together. Any number of registers can be linked together by placing the address of the state word of the second register in the link word of the first register, the address of the state word of the third register in the link word of the second register, etc., finally closing the chain by placing the address of the state word of the first register in the link word of the last. Buffer entries to any one register in the chain can be passed to all of them. When a register receives a report of a buffer entry, its program tag is changed so that future entries will select the task program appropriate to the then current state of the call.

The No. 1 ESS program is obviously the operating intelligence of the system. Its structure is determined mainly by the very large call handling capacity demanded of the system, and the fact that it must respond to service requests in real time (as they occur). Although specific data such as equipment quantities and translations, which change from office to office, are not written into the program, they are easily added to the memory system in any office. If the details of a special service or feature is changed, or a new one is added, the program is easily modified to incorporate it. From this stems the considerable, but quite realizable, claim that No. 1 ESS can handle telephone features and services that have not yet been foreseen.

![Simplified layout of a call store register. 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 scan word is not always used. Its function is to link a call store register to a scanning register when the program orders that some point in the system be scanned.](image)
The Control Unit of No. 1 ESS operates in the rapidly changing environment of a tolling address system. Its main task is to continuously monitor and control that environment so that all customers can be served efficiently and accurately.

A. H. Doblmaier

There are three major classes of instructions to central control. The first consists of orders for central control to sense the state of the environment. For example, an order may require examining the ferrod line scanner associated with customers' lines in order to detect requests for service. Central control detects any change of state in the line as a binary zero or binary one signifying off-hook or on-hook. In general, scanning is used to detect inputs to the system.

A second class of instructions processes the input data. Processing steps may include manipulating the data within central control, storing the results in the call store and recalling them when they are needed, and obtaining auxiliary data from the program store. But the processing of data may not be a perfectly straightforward progression from one step to the next. Central control's real power lies in its ability to decide, at any stage in the sequence, whether to continue through the program in an uninterrupted line or to jump to another area. Central control makes these conditional transfers on encountering certain conditions. It may, for example, look at the contents of a register and, according to what it finds, either transfer or continue the program.

The third class of instructions generates outputs from central control for operating relays in a trunk circuit, closing a network path, etc. As a rule, a single instruction governs a single operation. The complete set of instructions, however, is general enough so that individual instructions can be combined in various ways to implement any combination of office features.

In processing instructions, central control's first step is requesting information from the stores. It may retrieve service requests from the call store and ask for the processing instructions stored at a particular "address" in the program store. In return the program store sends central control a two-part instruction stored at that address. The first part of the instruction tells central control what to do with the information being processed. The second part, the address, tells where to do it.

Central control carries out the actions through various registers and flip-flop circuits, most of which are functionally designated. Program store addresses are generated in a program store address register and the instructions are received by an order word register. Call store addresses are generated in an index adder and data from the call store is received by a data buffer register. An instruction decoder is attached to the order word register. The decoded output of the order word register, combined with an appropriate clock pulse, controls the gating of information through the registers of central control. (The diagram on page 224 shows the organization of central control.)

There are eight general purpose flip-flop registers within central control connected together by two common buses. One or more of these (designated B, F, J, K, X, Y, Z, and I registers) usually is involved with processing an instruction. Designed to accommodate the basic word length of central control, each register is 23 bits long and the interconnecting buses each comprise 23 parallel information paths. A call store word is actually 24 bits, so the data buffer register is 24 bits long. The 24th is a parity check bit which drops out in the data buffer register.

A program sequence of instructions to central control might proceed as follows:

One. Take a memory address from an index register and read the word containing the first dialed digit of a call. Mask out all bits in the word except the four used for the first digit, and store the word in an accumulator register. (Masking, done in a mask and complement circuit, consists of changing all bits in a word, except the significant one to zeros.)

Two. Compare the word just placed in the register with the value 10 (the number of pulses counted when a customer dials zero).

Three. If the two compared quantities are equal, transfer to the program at an address specified in the instruction. If the quantities are not equal continue with the present program sequence.

Thus, at the third instruction central control decides if the customer has dialed zero (i.e., if the compared quantities are equal.) If so, it transfers to a program connecting the line to an operator. If not, it continues reading up dialed digits.

At any stage in a sequence central control may go to the registers for information. For instance, one of the flip-flop index registers may contain the address of a block of call store words needed in accumulating dialed information. One word from this block may be transferred to a second register where it can be used by the program. The task of central control, therefore, is to find that one word in the block of words and gate it to the appropriate register.

To begin, the program store sends central control an instruction to perform this task. That instruction might be paraphrased: 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.

Control center acts on instructions stored in the call store and program store memories to process a telephone call or to diagnose a trouble spot. There are over 12,000 logic packs, black squares at right center are groups of ferrod line scanners.
The actual instruction has three parts: first, the operation (fetch data from the memory and store it in the X register); second, a constant that identifies the pertinent word in the block; third, the address of the Y index register. All program instructions are written mnemonically. Memory to X register, for example, is written MX. If the constant in the second part of the instruction is, say, 3, then the instruction is written MX 3, Y, to find the pertinent word in the block of words. Central control sends two inputs to the index register. One input is the constant of the instruction, the other is a variable representing the contents of the specified index register. The index adder adds the two and the output of the adder is the address of the desired word.

In processing data, central control manipulates arithmetic quantities and logic quantities, both of which are couched in the binary language of digital machines. Arithmetic numbers contain 23 bits, designated 0 to 22. The bit at the right, called the least significant bit, the one at the left, the most significant bit. Bit 22, the sign bit, distinguishes between positive and negative numbers. For a positive number, the sign bit is 0. Negative numbers are expressed as the "ones" complement of their positive equivalent, and their sign bit is 1. For example, -1 is 000...001; -2 is 111…110.

An arithmetic circuit in central control adds, subtracts, compares, shifts, and rotates binary numbers. (Neither call processing, nor any other system function requires division or multiplication.) In addition to these arithmetic operations, the arithmetic circuit performs the logical operations AND, OR, and EXCLUSIVE-OR. (The role of the logic functions in line scanning is shown on page 200.)

In many cases, particular bits of information are not located in the most convenient bit positions in a word. This may occur, for example, if several small data words are packed together into adjacent bits of a larger memory word. Masking can be used to isolate a word within a larger word, but it does not affect the relative bit position. To move these bits into a more convenient position, the word can be transferred to an accumulator register in the arithmetic circuit which shifts or rotates the bits to other positions in the word.

In rotation, bits are passed through one end of the accumulator register and reinserted at the other end so that no information is lost. One bit, or a few, or all the bits in a word can be rotated as necessary, and it can be done from left to right or in the opposite direction. (Actually, the word is rotated as a unit by any specified number of places.) Shifting, the bits are forced out of either end of the register, but the vacant positions are then filled with zeros.

Another useful function in central control data processing is the rightmost one function. It consists of detecting and identifying a binary one in the midst of a number of binary zeros in a word. The one may signify, for example, an idle trunk. If central control is seeking a trunk over which to complete a call, the one indicates an idle trunk in the group, the zeros, busy trunks. The word storing the states of the group of trunks is gated to the accumulator. A rightmost one detect circuit then gates the position of the rightmost one to the F register through the bus. (F stands for "first one").

Sometimes, when a task is performed on a number of successive memory locations, it is convenient to set an index register to the binary value of the first memory address in the sequence, then to add 1 to the register to each succeeding word. This function, called "incrementing" is particularly useful in "looping" programs. For example, the instruction MX 3, Y gates the contents of the memory at the address Y + 3 into the X register. But MX 3, YW gates the contents of the memory at Y + 3 into the X register and increments Y by 1.

The index register may also be modified by changing it to an indexed quantity. In the instruction MX 3, YW, the W is the indexed quantity. This instruction changes the index register, Y, to the value of W which is the original value of Y, plus 3.

A final index register modification sets up the memory address to a specified quantity. MX 300, SY reads the memory at address 300 into the X register and places the quantity 300 into the Y register. (In the program mnemonics SY means set up Y.)

Two general approaches are common in the design of machines like central control— asynchronous and synchronous. An asynchronous machine works on the basis of a continuous flow of actions, each one triggering the next. A signal verifies that one action is completed before the
Timing diagram for non-coupled central control

The 5.5 microsecond cycle attained with this overlap operation approaches the ultimate cycle time for central control. No additional significant increase in processing rate would be gained by increasing the complexity of central control.

is checked by the call store. A second parity bit is returned by the call store and a data parity checker affirms the overall parity. Various check signals from peripheral units are received in special registers and analyzed by match circuits.

In the event of a fault, or an error, central control usually tries to repeat the operation on which the failure occurred. A persistent fault triggers an interrupt and central control tries an alternative routine. For example, if a fault appears to stem from a program store the central control may attempt to execute a special program stored in the call store and designed to uncover the defective program store. If these procedures do not work, central control attempts to reorganize the system and establish a working configuration. As a final measure, an emergency action sequence may take over control of the system and exercise different combinations of duplicated units and buses until it finds a working system. (For details see A New Approach To System Maintenance in this issue.)

Central control, then, is the executive part of No. 1 ESS. Although the flexibility of the system, as the preceding articles have emphasized, derives from the stored program technique, the effectiveness of the program itself depends on the efficiency of central control.

The overlap mode of operation in the central control of No. 1 ESS. The machine processes two instructions simultaneously. As one arrives at the auxiliary buffer word register, the address for the following one is sent to the program store. The 5.5 microsecond cycle attained with this overlap operation approaches the ultimate cycle time for central control. No additional significant increase in processing rate would be gained by increasing the complexity of central control.

The overlap mode of operation in the central control of No. 1 ESS. The machine processes two instructions simultaneously. As one arrives at the auxiliary buffer word register, the address for the following one is sent to the program store. The 5.5 microsecond cycle attained with this overlap operation approaches the ultimate cycle time for central control. No additional significant increase in processing rate would be gained by increasing the complexity of central control.
Memory Devices

R. H. Meinken and L. W. Stammerjohn

All the telephone switching logic and any other data vital to the completion of telephone calls are stored as binary bits in the memory cells of two new memory devices in No. 1 ESS—the ferrite sheet and the permanent magnet twistor.

Continuous pulse testing is performed on all twistor wire for the permanent magnet twistor memory to ensure that the signal levels are uniform for binary one and binary zero readouts. L. L. Venskike performs the test on wire to be fabricated into the device.

228 Bell Laboratories Record

June 1965

The basic operation of the memory devices in No. 1 ESS consists of storing a binary digit or "bit" in a specific location, called a memory cell, and recalling it on command. When memory cells are combined in large arrays, millions of bits can be stored in a single memory. Some bits are recalled after only a few millionths of a second. Others may be stored for years and recalled repeatedly as often as they are needed. To understand the operation of large memories, we can start with the behavior of a single cell.

A material or a device that has at least two stable states has memory. An ordinary light switch, for example, has two stable states, on and off, and by its position "remembers" a manual command. Electromagnetic switches or relays remember an electrical command, and they have functioned as memory devices in the telephone plant for many years. These devices, however, cannot operate at the high speeds demanded by modern electronic systems.

Memory elements prepared from specially developed magnetic materials have a number of properties that make them eminently suitable for high-speed systems. The fundamental property, their two stable states, is expressed as opposite directions of magnetization. But more significantly, these elements can be made to switch very rapidly from one direction of magnetization to the other. Furthermore, the coercive force of the material provides a threshold of operation that prevents small amounts of extraneous energy from changing the direction accidentally.

In the last few years, magnetic core memories have been used widely in digital systems. Both the permanent magnet twistor memory and the ferrite sheet memory of No. 1 ESS are related to these. In the simplest form of core memory, thousands of cores are assembled in a coordinate frame of wires. Two wires intersect orthogonally (in the X and Y directions, that is) at each core. A core can be magnetized in either a clockwise or counterclockwise direction; one way represents a binary zero, the other a binary one. A general description of the operation of core memories is a good foundation on which to build a comprehension of the twistor and ferrite sheet memories of No. 1 ESS.

To store or "write" a digit in a core memory, we control, or select, the direction of a core's...
Hysteresis loop of the ferrite material which composes the memory cell of the call store memory showing X, Y, and inhibit fields and one and zero flux states. The inhibit (write zero) field, added to the write one field, prevents the memory cell from switching and leaves the bit in the zero state.

magnetization by passing currents through the wires. Half the value of the current required to switch the core is applied to an X-column wire, the other half to a Y-row wire. Since the value of the current in each wire is less than the threshold of operation, the magnetization of a core is not changed if only one of the wires passing through it carries current. Only the core at the intersection of the two wires that receives their coincident currents is affected. After it has been set in either direction, a core remains magnetized without any applied current.

To recall or "read" the digit stored in a core, coincident currents are used again. If a core at the point of coincidence is already magnetized in the direction that would be induced by the current, it does not change. But if it is in the opposite state, it switches, and the reversal of the magnetic flux induces a voltage pulse in a "sensing" wire. Reading the memory, then, consists of sensing the pattern of voltages and no-voltages.

In this description, we have reduced the reading operation to its simplest level, the single bit. A memory that could read out only one bit at a time would be thoroughly inefficient for a high-speed system, however. Bits are actually strung together in parallel combinations, called words, and read out of the memory in aggregate. The structure and length of the binary words in any memory is determined by the requirements of the system. In core memories, it is possible to form words of almost any bit length by connecting the cores in proper wiring patterns.

Up to this point, we have been dealing with characteristics that are true not only of No. 1 ESS memories, but of the memories in many kinds of digital systems. Beyond these general ideas, however, the No. 1 ESS memories differ in significant ways from other memories and from each other. Electrically and mechanically, each is designed to suit its unique functions. These functions, described by the names temporary memory or call store for the ferrite sheet and semipermanent memory or program store for the twistor, have been discussed elsewhere in this issue. (See Features and Services.) Now let us take a detailed look at the most important principles in the design of first the ferrite sheet and then the twistor. After that we will consider some of the problems involved in achieving reliability and economy for these devices.

The Ferrite Sheet Memory

The mechanical construction of the ferrite sheet memory is modular. Four modules compose one call store. A module contains 192 separate ferrite sheets arranged in three adjacent stacks of 64 sheets. The storage capacity of each call store in a No. 1 ESS office is 8192 words of 24 bits each.

A single ferrite sheet contains 256 holes on a 16 by 16 grid. The material surrounding each hole is a memory cell similar to a single core in the core memory. This material is a bistable ferrite (i.e. it has a "square-loop" magnetization or hysteresis curve) that is uniform in composition. It was especially developed for these characteristics to permit coincident-current operation.

The ferrite sheet memory is similar to the core arrangement that has been described. Four conductors, called X, Y, sense, and inhibit windings, carry the currents used in reading information.
Reading a word out of the twistor memory involves accessing windings, biased ferrite magnets, word solenoids, twistor elements, and sense amplifiers. Magnetized cells (binary zero) are the blue bar magnets in this drawing. An unmagnetized one (binary one) are white. The fields in two intersecting access windings select a core (the one at the intersection of the blue X-access and Y-access wires) and induce a current in its copper strap. This generates a pulse of a few millivolts which the blue sense amplifier detects. Twistor elements under magnetized cells switch and generate a pulse of a few millivolts which the white sense amplifier detects. Twistor elements under unmagnetized cells flip into saturation and cannot switch. This generates a pulse which the blue sense amplifier detects and interprets as a binary zero.

Reading and writing operations are performed in the following manner. The memory element is selected, or addressed, by a coincidence of X and Y currents. The current pulses are bipolar; that is, a negative current pulse is always followed by a positive current pulse. The first, or negative part of the bipolar pulse, reads the memory cell. The positive part of the bipolar pulse writes into the cell. Current pulses are applied to the inhibit winding only during the writing operation.

In reading, the negative field created by the positive current pulse reverses the direction of magnetization in a cell in which a one is stored. The change in the magnetic flux induces a voltage pulse in the sense winding. The amplitude of this pulse is read as a binary one. A cell in which a zero is stored, however, is changed in its degree of magnetization only slightly by the negative field. A small reversible change induces a much smaller voltage in the sense wire. This pulse is read as a binary zero. After a cell is read and the negative field is removed, it returns to the remanent flux state that represents a binary zero.

To write a binary one into a cell that has been read, the positive field created by the positive current pulse is applied to the cell, reversing its magnetization. Positive current pulses in coincidence with the X and Y wires produce this field. To write a zero, however, really means to leave a cell essentially in the state resulting from the read operation. Therefore, the positive field applied to the cell must be weaker than the coercive force of the ferrite material. To achieve this, a negative current pulse is applied to the inhibit winding to create a negative field that opposes or cancels a part of the positive write field. Thus, the net field is not great enough to switch a large amount of flux in the cell. Following the write pulses, a negative pulse, called a post-write disturb pulse, is applied to the inhibit winding. This stabilizes the magnetization of the cell and improves its operating margins.

We have described the reading and writing operations on a single cell. To form the 24-bit word of the ferrite sheet, parallel operations are performed on 24 cells. Each cell, or bit, in a word is linked by common X and Y windings. Each inhibit and each sense winding links the bits in the same positions in each word of the module. In this way, a combination of one X, one Y, and one inhibit winding uniquely defines one bit in the memory.

A single module is wired with 64 Y conductors, 32 X conductors, and 24 interleaved inhibit and sense windings. Each path connects the cells it passes through in series electrically. Physically, each of these windings lies completely within a separate plane. Three planes containing one X, one Y, and one sense and inhibit winding are mutually orthogonal and their intersection defines a single bit or memory cell. The coincidence of an X and a Y winding, therefore, defines an entire word of 24 bits. A Y conductor, electroplated on each ferrite sheet during manufacturing, links all the cells in that sheet in series. Each Y conductor in the module is formed by serially connecting the conductors of the three sheets in corresponding levels of the three stacks. The X conductor for a single wire threaded up and down through the cells in a stack. Inhibit and series windings are threaded through the cells in a similar manner.

The drawing on page 232 shows the wiring patterns in a cutaway section of the module. Although coincident currents are required to switch a cell, a single pulse on either the X or Y conductor “half-selects” a bit and causes a small reversible change from the remanent flux of the magnetic material. This change generates a small voltage that interferes with the signal at the output of the conductor. In a large memory, the outputs of all half-selected bits linked by a common path add coherently and the sum of their amplitudes could exceed the wanted signal, causing errors in output information. To reduce this effect, the wiring patterns in a module are designed to cancel most of the outputs of the half-selected bits. Complete cancellation is not possible since the amplitude of this noise depends on the state of information stored in the half-selected bits. Thus, noise due to half-selected bits is a factor limiting the size of the memory.

The Twistor Memory

Now let us turn to the twistor memory. Its basic module is an accordion-folded stack of 64 flat planes, each having an array of 2816 twistor cells on each side. Into this stack are inserted 128 memory cards, each card facing each side of each plane. The cards, made of aluminum, bear a matching array of 2816 tiny permanent magnets (see the photograph on page 206) arranged in a grid of 44 rows and 64 columns. One program store (see page 221) contains 16 of these modules with a total storage capacity of 5.8 million bits of information. The store is organized into 131,072 words of 44 bits each. In contrast to the ferrite sheet memory, the twistor can be read an unlimited number of times without destroying the stored information. However, this does incur a penalty. Stored information cannot be changed electrically within the module. It is changed by temporarily removing the twistor cards and magnetizing or demagnetizing the permanent magnets.

These magnets, which are rectangles of vanadium-aluminum, are the basic memory cells. A magnetized spot represents a binary one; a spot that is unmagnetized represents a one. A twistor wire—a copper wire that has been helically wrapped with a thin permalloy tape—runs along the twistor wire plane under each row of magnets. A copper strap runs under each column, intersecting orthogonally with a twistor wire directly under each magnet. The intersection of a copper strap and a twistor wire forms a twistor cell which senses the condition of the memory cell or magnet. Twistor wires are formed into two flat cables that link all planes in a stack. Each twistor wire (there are 44 in a cable) reads out one bit in a word. A copper wire, parallel to each twistor wire, completes the transmission path for output signals. Each copper strap (there are 64 to a plane) defines the bits in a complete memory word.

The pattern in which the twistor cell accesses the state of the vanadate magnet can be described quite simply. (See the drawing on page 232.) Magnetic flux in the permalloy tape wraps helically around the twistor wire and links both the wire and the copper strap at every intersection. Because this flux path closes through the air, the twistor cell is...
sensitive to ambient magnetic fields. Thus the strong field of the vicalloy magnet can affect the operation of the twistor cell. If the spot over an intersection is not magnetized, the field due to a current pulse in the strap will reverse the magnetization, i.e., switch the twistor cell. This induces a voltage pulse in the twistor wire which is read as a binary one. On the other hand, if the spot is magnetized, its external field saturates the permalloy tape at the intersection. Then the pulse in the copper strap cannot reverse the magnetization, i.e., switch the twistor cell. This makes the copper word strap more efficient and reduces its impedance.

Because it is grounded through the retaining spring structure and the frame of the module, the card also serves as a conducting ground plane which reduces common-mode noise in the twistor line. A permalloy sheet, placed under the copper straps, also contributes to the effectiveness of the strap currents. The permalloy concentrates the magnetic field of the strap current in the vicinity of the twistor cell. It serves also as a distributed magnetic shield that reduces the effect of unwanted ambient magnetic fields.

Reading and writing in the memories, then, can be described in terms of a single bit and the operations on a memory word are parallel interactions of the basic action. But the construction of the No. 1 ESS memories must be considered in terms of blocks of cells.

Among the primary design goals of both memories were low first cost and low maintenance cost. It would be prohibitively expensive to fabricate individual cells and then interconnect them. The solution to the problem of low first cost is to fabricate a large number of cells and their interconnections in a few common steps. Low maintenance cost depends upon building a device with long life and stable characteristics so that adjustment during service is unnecessary.

Memory cells must be exceptionally uniform, because if a single cell fails an entire module is rejected. Materials and processes were developed to promote uniformity. The memory structures were designed to support the cells so that changing ambient conditions affect them as little as possible. In common with older magnetic components, such as transformers, the No. 1 ESS memories show any appreciable aging due to changes in their primary magnetic characteristics.

In the ferrite sheet memory, 256 cells of each sheet are formed simultaneously. Ferrite powder is pressed into the sheet which is first fired and then metallized. During the last step, the Y conductor is plated directly on the sheet. The frame of the module is designed to hold all the sheets in alignment and support them without strain. This meets the mechanical objective and it also facilitates wiring. As a result of this method of manufacture, the ferrite sheet memory, in comparison to earlier core memories, requires fewer interconnections. The result of this is significant improvement in first cost and reliability.

In the twistor memory, the objectives of a minimum number of connections and the elimination of mechanical strain on the magnetic elements are achieved in one operation. The twistor wire cable, running continuously through a module, is laminated between sheets of polyester which hold the wires in precise position and protect them as well.

The memories for No. 1 ESS have been designed specifically for their application in a digital control system which must operate continuously in real time. The permanent magnet twistor provides a semipermanent memory device uniquely suited to the storage of program and translation data in this system. General purpose digital computers use no directly comparable memory device. The ferrite sheet memory, while its electrical properties are directly comparable to the core memories in general use, is an exceptionally compact memory containing about 350 bits per cubic inch. Excluding terminals and connectors, the density is 8000 bits per cubic inch. For comparison, the twistor memory contains about 250 bits per cubic inch.

In the last three years, the Western Electric Company has manufactured many twistor memory modules and ferrite sheet memory modules. Test experience has amply demonstrated the uniformity of the electrical performance in these memories and the efficiency of producing arrays of memory elements in common operations on a large scale.

One module of the ferrite sheet memory. An office of about 10,000 lines generally requires two or more cell stores (4 modules). A 65,000 line office with a high calling rate may contain 10 stores.