Innovation in learning
by a McGraw-Hill
Continuing Education School

Na
139-OM-701(501)

DIGITAL ELECTRONICS

THE MODEL 832
DIGITAL COMPUTER

REFERENCE MANUAL

10KX

McGraw-Hill Continuing Education Center

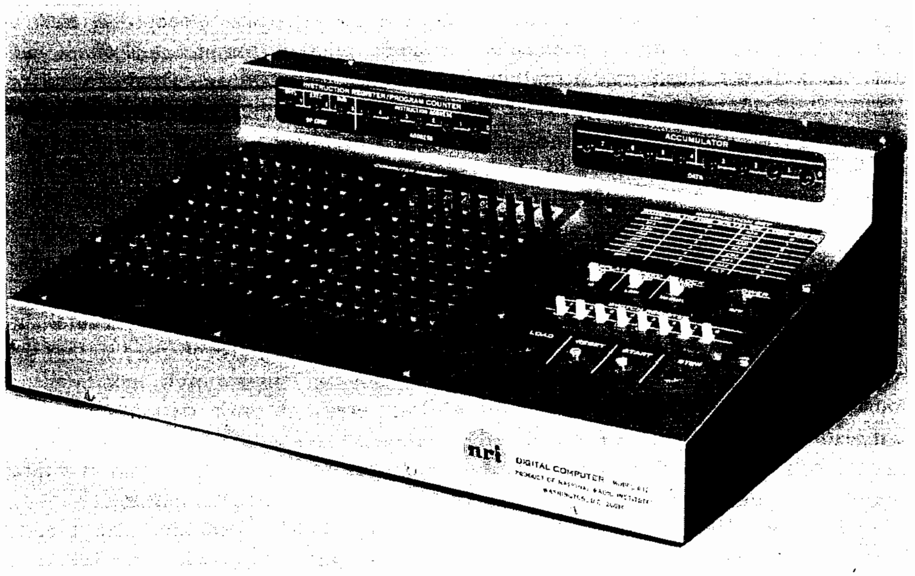
THE MODEL 832 DIGITAL COMPUTER

REFERENCE MANUAL 10KX

INDEX OF SECTIONS

NATIONAL RADIO INSTITUTE, WASHINGTON, D.C. 20016

1979 EDITION



The Model 832 is a small digital computer designed for training purposes. It fills the gap between those training devices designated as logic trainers for teaching digital logic and computer circuitry and the more complex and harder to program minicomputers. It was designed to minimize circuitry and simplify operation but at the same time maintain all of the major features of a real digital computer. Those using it in a training program can quickly and easily learn the fundamentals of digital computers including organization, operation and programming. Let's look at the major features of the Model 832.

The Model 832 was designed especially for training. The architecture, logic, and instruction set of the Model 832 were designed to optimize the computer's use as a training device. Its physical arrangement also serves this purpose. The large front panel contains two easy to read register displays as well as the memory switches that permit rapid access to one-half of the computer's memory so that programs and data can be quickly changed without the need for costly input/output devices. A minimum of controls are carefully placed to permit ease of operation.

The Model 832 uses modern SSI and MSI TTL integrated circuits. The use of 7400 TTL integrated circuits makes the Model 832 a reliable, low-cost digital computer with a minimum amount of hardware. The student works with the latest and most popular integrated circuits in use in digital equipment today.

The Model 832 allows the student to study detailed circuit operation of the digital computer at the gate level. The student can demonstrate all of the basic logical concepts and most widely used digital circuitry. At the same time the computer design permits the student to work at a higher level; he learns to write programs to implement popular algorithms for solving typical problems. The student can easily program the computer to perform virtually any operation. Many advanced programming concepts can be demonstrated.

The Model 832 was designed to teach digital computer fundamentals. The best way to learn computer organization and operation is to study a particular digital computer thoroughly in order to learn its operation and to become intimately familiar with its instruction set so that useful programs can be written. The Model 832 is small enough that anyone can readily learn its operation and master its instruction set to demonstrate all of the most important programming fundamentals. Regardless of which type of computer you eventually work with, you will demonstrate all of the important digital computer concepts with the Model 832.

GENERAL SPECIFICATIONS

Memory. Maximum memory size 32 8-bit binary words. Sixteen 8-bit words of memory are available in the programmable read-only switch/diode matrix memory. Another sixteen 8-bit words of memory are available in the form of a bipolar semiconductor read/write random access memory. The first sixteen 8-bit words of memory are represented by front panel switches so that program and data changes can be quickly and easily made without the need for costly input/output devices. This one feature alone makes the Model 832 computer one of the most valuable computer training devices ever built.

Control. Stored program, fully synchronous operation. There are two basic computer modes, instruction fetch and instruction execute.

Addressing. The Model 832 uses a single address format and direct addressing of memory. Other forms of addressing, such as indirect, indexed, and relative, can be demonstrated through programming.

Clock. Both low and high speed clock pulses are available. The high speed clock runs at approximately 250 kHz. The slow clock runs at approximately 2 Hz. The slow speed clock operation executes the program at a low rate of speed so that the student can observe each operation and logic event. A single step mode is also available so that the student operator can step the program a clock pulse at a time with a front panel push button.

Instructions. There are fifteen basic instructions, seven memory reference instructions, and eight generic or literal instructions. These include load/store, arithmetic/logic, jump/skip, shifts, and register transfers.

Input/Output. The input is via front panel binary switches; the output is by two 8-bit binary light displays.

Programming. Programming is strictly by machine language using hexadecimal notation.

Circuitry. Seventy-four type 7400 TTL integrated circuits, both the SSI and MSI type, are used to implement the Model 832. All integrated circuits are installed in sockets. This makes the IC's easy to replace should one ever become defective and permits demonstrations of trouble-shooting. All sockets and circuitry are mounted on highly reliable, double-sided G-10 printed circuit boards.

Basic Word Size. 8 bits.

Arithmetic. Binary representation of numbers with 2's complement representation of negative numbers. All processing is done serially. However, there are numerous internal parallel data transfers.

Size. The Model 832 is 22 inches wide, 7 inches high, and 14 inches deep.

Color. Blue cabinet, front panel of brushed aluminum with red trim and white lettering.

Cabinet. Made from heavy sturdy steel with an aluminum front panel.

Total Weight. 16 pounds.

Power Source. 95 - 125 VAC, 50/60 Hz, 25 watts.

Computer Operation

Fig. 1 (center fold) shows a complete block diagram of the Model 832. In this section we will analyze the basic operation of the computer using this diagram. You will also study the basic overall operating concepts of the machine.

The Model 832, like any digital computer, operates on a stored program concept. A series of instructions forming a program to produce some useful calculation or operation is stored in the computer's memory. The computer sequentially fetches these instructions from memory one at a time, analyzes them, and then carries out the operations specified by the instructions. The step-by-step sequence of instructions causes the desired operations and calculations specified by the program to take place.

REGISTERS

The various registers used in carrying out the operations specified by the instructions form the very heart of any digital computer. The Model 832 contains six operating registers. Let's define them before we continue with a detailed block diagram analysis.

A Register. The A register is the accumulator or main operating register in the computer. All of the instructions in the computer refer to operations that take place on data stored in the accumulator. The accumulator is the basic arithmetic logic register in the machine. It is eight bits in length. Its content is displayed continuously on the eight right-hand display lamps.

B Register. The B register is an auxiliary 8-bit register used to store intermediate results of program calculations. It is generally used to exchange data with the accumulator. A rotate A and B instruction causes the data in the accumulator to be transferred to the B register and the data in the B register to be transferred to the accumulator. In performing the logical AND instruction, the contents of the A and B registers are ANDed together with the result appearing in the accumulator.

E Register. This is another 8-bit auxiliary register which is used to store intermediate results in calculations. The contents of the accumulator can be shifted into the E register while the contents of the E register are shifted into the accumulator. This is done with a rotate A and E instruction.

In some versions of the Model 832 digital computer, the bipolar semiconductor read/write memory is an option. When this memory is not contained within the computer, the E register is wired as a read/write portion of the memory and simulates one memory location.

I Register. This 8-bit register is the instruction register. It holds the instruction word during the execution of that instruction. The I register is loaded serially from the memory. Its contents are then decoded to determine the operation to be performed. The I register contents can be displayed by the eight left-hand display lamps via a display multiplexer. During jump instructions (conditional or unconditional branches in the program) the five address bits of the instruction register are transferred to the program counter to tell the computer where to fetch the next instruction.

M Register. The M register is the memory address register. It is five bits in length and holds the address of the word to be fetched from memory. The word, of course, may be either an instruction word or a data word. The memory address register is loaded from the program counter during instruction fetch operations. Its contents are decoded and the desired instruction is loaded serially into the I register. During the execute phase of the operation, the M register is loaded from the 5-bit address portion of the instruction register.

P Register. The P register is the program counter. This 5-bit counter contains the address of the next instruction in sequence to be executed. It is normally incremented once for each fetch/execute phase. Its content is fed to the memory address register during fetch operations to determine the location of the instruction to be fetched and executed. The program counter is loaded from the instruction register during conditional or unconditional jump instructions to alter the sequence of the program. The output of the program counter is fed to a display multiplexer where its contents may be observed on the left-hand binary lamp display.

BLOCK DIAGRAM ANALYSIS

As shown in Fig. 1, the Model 832 digital computer is divided into three basic sections, the memory, the control section, and the arithmetic/logic section. The memory stores the program and the data. The control section sequentially examines the instructions of the program stored in memory and generates the control signals that carry out the specified operations. The arithmetic/logic unit manipulates the data according to the control signals generated by the control section to carry out the functions specified.

Memory. The memory in the Model 832 digital computer consists of a maximum of 32 8-bit memory locations where data and instruction words can be stored. The first half of the memory is the programmable read-only memory that is a switched diode matrix making up a total of sixteen 8-bit words. For each bit position there is a switch and a diode in the matrix. The 128 switches are on the left side of the front panel of the computer grouped as sixteen 8-bit words. The memory is programmed by setting the switches. Either data or instructions may be stored this way.

Data can be read out of the memory only. Data cannot be written into the memory under the control of the computer with a store instruction. However, the memory is fully programmable by setting the front panel switches.

The switched diode matrix programmable read-only memory is used in the Model 832 because of its convenience in programming the computer, particularly for training purposes. To keep the cost of the computer low, no input/output devices are used. Therefore, to get data into the memory quickly and conveniently without an input device, the memory was designed to provide rapid data entry and changes.

In addition, the contents of the memory can be readily observed by visually noting the positions of the switches. Because this portion of the memory is so easy and quick to change, many different programs can be entered and run in a short time.

Another advantage of the switched diode matrix programmable read-only memory is its nonvolatile characteristic. Since all the data is stored as mechanical switch settings, turning off the power does not destroy the contents of the memory.

Despite the small size of the programmable read-only memory, an amazing number of programming and computer fundamentals can be demonstrated with just sixteen words. There are very few basic computer concepts that cannot be demonstrated.

The more sophisticated programming experiments require additional memory space, however. For this purpose, sixteen additional 8-bit words of memory are used in the Model 832. This portion of memory is made up of a random access, read/write bipolar semiconductor memory. It consists of a total of 128 TTL flip-flops, one for storing each of the bits in the sixteen 8-bit words. These flip-flops are grouped by words and are addressable. Data can be read from the memory and easily written into the memory under program control with a store instruction.

The bipolar semiconductor memory is volatile, so whenever computer power is removed, all of the flip-flops storing the data will be disabled. When power is reapplied, the states of the flip-flops will come up in some random manner, thereby destroying any program previously stored there. For that reason, care should be taken not to remove power if you wish to retain the contents of the semiconductor memory.

Now refer to the memory section in Fig. 1. The programmable read-only memory is shown as a block, as is the bipolar semiconductor memory. The circuitry used to address a single word in either portion of the memory consists of a memory address multiplexer, the memory address register, and a memory address decoder. The memory address multiplexer accepts two parallel 5-bit words, one from the program counter and the other from the address section of the instruction register. The multiplexer causes either of these two 5-bit words to be stored in the memory address register.

During the fetch phase, when an instruction is to be taken from memory and analyzed the memory address multiplexer causes the output of the 5-bit program counter to be transferred to the memory address register. At this time the five input lines from the instruction register are inhibited. During the execute phase of the computer operation, when the instruction is carried out, the memory address multiplexer inhibits the program counter inputs and causes the five address bits from the instruction register to be stored in the memory address register. This is true for all memory reference instructions where an instruction specifies an operation on some data word stored in memory. The five bits from the address section of the instruction register designate the location of that operand.

The 5-bit address word stored in the memory address register specifies one of the 32 total memory words. These five bits from the memory address register are applied to a memory address decoder and directly to the bipolar semiconductor memory. The memory address decoder looks at the four least significant bits of the memory address register and selects one of sixteen words in the programmable read-only memory. These four bits from the M register are also applied to the bipolar semiconductor memory. The decoding for all sixteen words in this memory is contained within the integrated

circuit memory itself. The four bits then select one of the sixteen words in the bipolar semiconductor memory.

As you can see, two words in the memory have been addressed by the four bits in the memory address register, one in the programmable read-only memory and the other in the bipolar semiconductor memory. However, we want only one word to be read out. The two words addressed in the two halves of the memory are read out in parallel and are applied to memory multiplexers. These multiplexers are enabled by the fifth bit in the memory address register. Only one of the two multiplexers will be enabled. If the desired word is stored in the programmable read-only memory, then the associated memory multiplexer will be enabled. The multiplexer associated with the bipolar semiconductor memory will be inhibited.

The purpose of the memory multiplexers is to produce a parallel-to-serial data conversion. The outputs of the programmable read-only memory and the bipolar semiconductor memory are eight parallel bits. The multiplexers scan these eight bits one at a time, causing the data at the output to be in serial form. The outputs of the two multiplexers feed an OR gate whose output is the serial memory data.

It is not possible to store data under program or instruction control in the read-only memory. However, data can be stored in the bipolar semiconductor memory. This is done with a Store Accumulator instruction. The eight output bits from the accumulator are applied directly to the bipolar semiconductor memory. When a Store Accumulator instruction is executed, the contents of the accumulator are transferred in parallel and stored in the location addressed.

The Control Section. The control section of the computer is the unit that sequentially examines the instructions of the program stored in memory. It interprets their meaning and generates the necessary control signals to have the arithmetic logic unit carry them out. In Fig. 1 you can see that the control section consists of the program counter, the instruction register, the instruction decoders as well as the clock, the timing counter, the timing decoder and the fetch/execute, run/stop and reset flip-flops. A display multiplexer permits the contents of both the program counter and the instruction register to be displayed.

There are two distinct phases of operation of the computer, the fetch phase and the execute phase. In the fetch phase, the 5-bit address portion of the program counter is transferred to the memory address register. Then the instruction word stored in memory is read out and loaded serially into the instruction register. After this happens the computer goes into the execute phase where the arithmetic logic unit carries out the function specified by the instruction. During this time, the 5-bit address portion of the instruction word in the instruction register is transferred to the memory address register where the operand to be used by the instruction is addressed and read out serially.

Once the execute phase has been completed, the computer again goes into the fetch phase where the next instruction in sequence is read out and executed. This procedure continues until all of the instructions in the program have been executed and a halt instruction is recognized.

The basic source of timing and control in the computer is the clock oscillator. It generates high frequency rectangular pulses that are used to control all sections of the computer. These clock pulses cause all of the counters to count and registers to shift at the appropriate times. The clock is controlled by a run/stop flip-flop. When the flip-flop is reset, it stops the clock so that no pulses are produced. When this flip-flop is set, it is in the run state and allows the clock to generate pulses.

The run/stop flip-flop is in turn controlled by a clear (CLR) signal generated by the reset flip-flop. The reset flip-flop is operated by a front panel push button which is used to reset and clear this flip-flop as well as the program counter, instruction register, accumulator, and fetch/execute flip-flop in the computer. The run/stop flip-flop is also controlled by the halt signal. Whenever a halt instruction is decoded, it causes the run/stop flip-flop to be reset, thereby stopping the clock and terminating the program.

The clock output pulses feed a 3-bit binary timing counter. This 3-bit ripple counter drives the fetch/execute flip-flop. This flip-flop determines the phase of operation of the computer. When it is reset, the computer is in the fetch state. When this flip-flop is set, the computer is in the execute state. The output of the timing counter causes the fetch/execute flip-flop to toggle alternately between the fetch and execute states. The timing counter counts eight pulses and then causes the fetch/execute flip-flop to be complemented. For example, if this flip-flop is initially in the reset or fetch state, it will become set after eight clock pulses have occurred. After the second eight clock pulses have occurred, the flip-flop will again toggle from the execute to the fetch state.

The reason for counting by eight is because the word length used in the computer is eight bits. It takes eight clock pulses to cause the contents of an 8-bit register to be shifted in or shifted out. Therefore, all operations in the computer can take place during eight clock pulses. It takes eight clock pulses to load the instruction register with the 8-bit instruction word stored in memory. It takes an additional eight clock pulses to cause an arithmetic or logic operation to take place on the contents of the 8-bit accumulator.

In operating the Model 832 digital computer, the reset button on the front panel is pressed first to make sure the clock is stopped. This operation also clears the program counter, the instruction register, the accumulator register, the timing counter, and resets the fetch/execute flip-flop to the fetch state. The computer is now ready to run. When the start button is depressed it automatically sets the run/stop flip-flop, allowing the clock to operate. During this fetch phase the contents of the program counter (00000) are transferred through the memory address multiplexer to the M register. This means that the computer automatically looks for the first instruction at memory location 0_{16} in the programmable read-only memory. The first instruction word of any program should be stored there.

The instruction stored in memory location 0_{16} is read out serially and stored in the instruction register. It takes eight clock pulses for this operation to occur. At the end of eight clock pulses the timing counter toggles the fetch/execute flip-flop. This puts the computer in the execute state. The five address bits in the instruction word are transferred through the memory address multiplexer to the M

register. Here the memory word referred to by the previous instruction is read out.

For example, suppose that a Load Accumulator instruction is detected. The address specified by the instruction word is fed to the memory where the contents of the specified memory location are read out serially and transferred into the accumulator register. It takes a total of eight clock pulses for this to happen. Once these eight clock pulses have occurred the data word will be in the accumulator and the timing counter will recycle, resetting the fetch/execute flip-flop to the fetch phase.

During the execute phase, the program counter is incremented to 0001. This automatically causes the program counter to contain the address of the next instruction in sequence to be fetched. When the fetch phase occurs, the program counter contents are again transferred to the M register where the next instruction of the program in sequence is read out and stored in the instruction register.

The instruction register is monitored by the instruction decoder. This decoder looks at the op code (three most significant bits) of the instruction word. It produces eight output states, only one of which is enabled at a time. The output line enabled determines which instruction is to be executed.

If the operate (OPR) instruction is specified by the op code, it enables the operate function code decoder. This is another decoder that monitors the three least significant bits of the address portion of the word in the instruction register. Only one of the eight output lines of this decoder will be enabled. The output line enabled will define which of the eight possible operate functions will be executed.

The output control lines from the instruction and function decoders are combined in logic gates along with timing pulses generated by the timing decoder. The timing decoder monitors the three flip-flops in the timing counter. Eight output signals are generated by the timing decoder. The outputs of the timing decoder designated TØ, T1, T2 and T7 are the ones used to generate most of the control signals.

The Arithmetic/Logic Section. The arithmetic/logic unit (ALU) of the Model 832 digital computer is that section which carries out the operations specified by the instructions. Most of the instructions specify some arithmetic or logic operation. These arithmetic or logic operations will be carried out on the data contained in the accumulator, the B register, the E register and from data words in memory specified by the instruction. The various operations are performed by the register gating logic and the arithmetic/logic unit.

As an example of the operation of the ALU, consider the arithmetic operations of add and subtract. When an add operation is selected, the contents of the accumulator will be added to the word stored in the memory location specified by the instruction word. These two words are added serially, a bit at a time, in the arithmetic unit and the sum is stored back in the accumulator. When a subtract operation is specified, the contents of the memory location specified by the instruction are subtracted from the contents of the accumulator. Again the operation is performed serially, a bit at a time, and the difference is stored back in the accumulator. The carry flip-flop is used to store any carry condition that arises, so that the carry may be added to the next bit position.

The register gating logic block shown in Fig. 1 permits a wide range of data transfers between registers. Data may be exchanged between the accumulator and the B register or the accumulator and the E register. The logical AND operation also takes place between the A and B registers when it is executed.

The overflow flip-flop determines a specific overflow condition in the accumulator. Should a carry output be generated by the most significant bits of the word in the accumulator and memory during an add operation, an overflow condition is indicated. This may or may not be a genuine overflow or error condition, depending upon the data in the register and the operation being carried out. However, the overflow indication is given so that it can be detected should the operator require this detection. The overflow is displayed on the 8-bit lamp display shared with the 5-bit program counter.

Word Formats

Figs. 2 and 3 show the instruction and data word formats used in the Model 832 digital computer. Both are eight bits in length.

INSTRUCTION WORD FORMAT

Fig. 2 shows the instruction word format. The 8-bit word is divided into two sections, the op code and the direct address. The bit positions are numbered Ø through 7 from right to left. Bits 7 through 5 (7:5) are used to define the instruction op code. This permits a total of eight basic instructions for the computer. Seven of these instructions are memory reference instructions in that the operation they specify calls for an operand from some location in memory. That location in memory is identified by the address portion of the instruction word which is made up of bits 4 through Ø (4:Ø). These five bits permit us to define a maximum of $2^5 = 32$ memory locations. All memory locations are addressed directly. There is no immediate, indirect, indexed or literal addressing in the Model 832. However, some of these modes of addressing can be demonstrated through programming.

The eighth instruction defined by the op code is the operate (OPR) instruction. This instruction defines operations that do not require addressing a location in the memory for an operand. These include logic operations, shifts, skips and register transfers. Whenever the OPR instruction is selected, the bits of the address portion of the instruction word are used to further define the exact function to be carried out. Bits 2 through Ø (2:Ø) can specify eight functions. Bit 3 is not used. Bit 4 defines the direction of the shift operation specified by the function code. A binary 0 represents a right shift and a binary 1 represents a left shift.

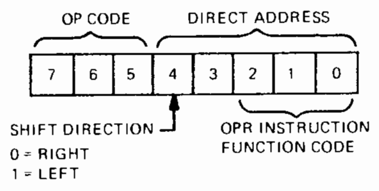


Fig. 2. Instruction word format.

DATA WORD FORMAT

Fig. 3 shows the data word format for the Model 832. As with the instruction word, the data word bits are numbered 7 through Ø from left to right. In the Model 832 digital computer, the data may be either binary bit patterns or numbers representing a specific magnitude. In either case, the pure binary code is used.

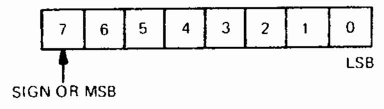


Fig. 3. Data word format.

When the data word defines a number, there are several ways in which these numbers can be represented. These are positive integers, signed integers or signed fractional numbers. Table I shows how numbers are represented in each of these three number systems. In each case the largest and smallest numbers possible in each scheme are indicated. The decimal,

NUMBER REPRESENTATION SYSTEM	DECIMAL	OCTAL	HEX	BINARY
Positive Integer				
Largest Number	+255	377	FF	11111111
Smallest Number	0	000	00	00000000
Signed Fractional (2's Complement)				
Largest Positive Number	+127/128 (.9921875)	177	7F	0.1111111
Smallest Negative Number	-128/128 (-1.0)	200	80	1.0000000
Signed Integer (2's Complement)				
Largest Positive Number	+127	177	7F	01111111.
Smallest Negative Number	-128	200	80	10000000.

Table I. Number representation in the Model 832.

octal, and hexadecimal binary representations for each are given for convenience.

In the positive integer number representation system there are no fractional or negative numbers. The smallest number represented is 0, the largest is 255. As you recall, the largest number that can be represented by a binary code is equal to $2^n - 1$, where n is the number of bits. This gives us $2^8 - 1 = 256 - 1 = 255$. For many of the calculations and demonstrations in the Model 832 this number representation is the most convenient to use.

In the signed fractional number representation system, also known as the 2's complement system, the binary point is assumed to be between bits 6 and 7 of the data word. This means that bit 7 is used as a sign bit where a binary 0 equals a positive sign and a binary 1 represents a negative sign. The remaining bits are used to define the magnitude of the number. With the binary point to the left of the seven bit number, all of the numbers will be fractional. The largest positive number, then, is +127/128 or in decimal form .9921875. The smallest negative number is -128/128 or -1. This means that all of the numbers represented by the data word in this system are between +1 and -1. A wide range of numbers can be represented by using this system to scale the numbers to represent other numbers smaller or larger than the fractional value.

In the signed integer number representation the 2's complement representation of numbers is also used. Here the binary point is assumed to be to the right of the number. This permits us to represent the range of numbers between +127 and -128. Since bit 7 is used as the sign bit, this leaves a total of 7 data bits. The maximum that can be represented with the 7 bits is $2^7 - 1 = 128 - 1 = 127$.

The Instruction Set

There are 15 basic instructions that the Model 832 digital computer can execute. The 3-bit op code in the instruction word defines eight instructions, seven of which are memory reference instructions. The eighth is a generic instruction that further defines eight functions or operations that do not require memory reference. In this section we completely define the instruction set for the Model 832 digital computer.

MEMORY REFERENCE INSTRUCTIONS

Load Accumulator (LDA) Op Code 000

The load accumulator instruction is used to transfer the contents of the memory location specified by the 5-bit address in the instruction word into the accumulator (A) register. When this instruction is executed, the contents of the designated memory location will be transferred serially into the accumulator register.

Store Accumulator (STA) Op Code 001

The store accumulator instruction causes the contents of the accumulator register to be stored in the memory location designated by the 5-bit address in the instruction word. Because of the nature of the memory of the Model 832, the STA instruction has no effect when addresses 00000 through 01111 are used. The first 16 words of memory defined by these addresses are read-only memory locations. They are programmable in that they can be set to the desired contents with the front panel programming switches. However, you cannot automatically write into these locations from the computer itself. For that reason when an STA instruction specifies an address in this range, nothing will take place. However, it is possible to write data into the last 16 words of memory, addresses 10000 through 11111, with the STA instruction. When this instruction is executed, the contents of the accumulator will be transferred in parallel to the bipolar semiconductor memory and stored in the designated location.

In some versions of the Model 832 computer, the semiconductor memory is an optional feature. For those models of the computer without this memory, the E register within the computer is wired to act as a single memory location into which data can be written. This single register acts as a memory location that is addressed with bit 4 of the instruction word. When this bit is a binary 1 and the STA instruction is specified, data will be transferred from the accumulator to the E register in serial form for storage. Any address between 10000 and 11111 will cause the contents of the accumulator to be stored in the E register. When the optional memory is available, the E register is not used for this purpose.

Add (ADD) Op Code 010

The ADD instruction causes the contents of the accumulator (A) to be added to the number stored in the memory location designated by the address portion of the instruction word. The sum will be stored back in the accumulator. The number previously stored in the accumulator will be lost. The addition process takes place serially, a bit at a time. The addition process is strictly algebraic in nature and it may be in any of the three formats that were designated earlier.

Subtract (SUB)
Op Code 011

The SUB operation causes the contents of the specified memory location to be subtracted from the contents of the accumulator. The difference will appear in the accumulator. The number stored there previously, the minuend, is lost. The operation is carried out serially, a bit at a time.

Unconditional Jump (JMP)
Op Code 100

This instruction is an unconditional jump or branch instruction that causes the computer to execute instructions out of their normal sequence. When the JMP instruction is executed, it causes the address designated by the jump instruction to be transferred to the program counter. This means that the next instruction to be executed will be that instruction that is contained within the memory location specified by the jump instruction.

Jump-On-Minus (JOM)
Op Code 101

This instruction causes the sequence of executing instructions in the computer to be changed if the accumulator contains a negative number. The computer circuitry monitors bit 7 of the accumulator and interprets it as a sign bit. If it is a binary 0 the number is considered to be positive. If bit 7 is a binary 1, the number is negative (minus). When the JOM instruction is executed, the circuitry looks at bit 7 of the accumulator. If it is a binary 0 then no change will take place. The computer will continue to execute the instructions of the program in normal sequence one after another. However, if bit 7 is a binary 1, the 5-bit address specified by the JOM instruction will be transferred to the program counter so that the next instruction to be executed in sequence will be taken from that address.

Jump-On-Zero (JOZ)
Op Code 110

This conditional jump instruction causes the computer to execute instructions out of sequence if the number in the accumulator is zero. All eight bits of the accumulator are monitored by the circuitry and when all are binary 0's, a logic signal is generated to indicate this condition. When the JOZ instruction is executed, the circuitry looks at this zero indication signal. If it is a binary 0, indicating that the accumulator is not zero, then the computer will continue to execute instructions in the normal sequence. If the zero indication signal is a binary 1, it tells the logic circuitry that the content of the accumulator is zero. The address of the next instruction executed will come from the address specified by the JOZ instruction. This address is transferred from the instruction register to the program counter. Instructions following this will be in sequence starting from that address.

Operate (OPR)
Op Code 111

The OPR instruction is a generic instruction not requiring reference to a memory location. For this reason the bits of the instruction word normally used to designate an address are used to form a function code that specifies a specific operation to take place. There are eight operations that can take place with the OPR instruction. These are defined by a 3-bit function code, bits 2 through Ø (2:Ø) in the instruction word format. These eight operations are defined next.

OPERATE INSTRUCTIONS

Rotate A and B (RAB)
Function Code 000

When this instruction is executed, the contents of the accumulator (A) register will be exchanged or rotated with that of the B register. The contents of the accumulator will be shifted out serially into the B register while the contents of the B register are shifted into the accumulator. This instruction is specifically designed for the Model 832 digital computer to conserve memory space. It permits the storage of intermediate results during calculations without reference to a memory location. The contents of the accumulator can be saved in the B register with this instruction and then later recalled by again executing the instruction.

Complement Accumulator (CMA)
Function Code 100

Executing this instruction causes the 1's complement of the contents of the accumulator to appear in the accumulator. The binary number in the accumulator is shifted out a bit at a time, inverted, and then shifted back into the accumulator. When this instruction is executed the number in the accumulator is lost but its complement appears in the accumulator where all 1's have been changed to 0's and all 0's have been changed to 1's. The 2's complement version of a number in the accumulator can be produced by executing the CMA instruction and then by adding one with an add instruction.

Logical AND (AND)
Function Code 010

Executing this instruction causes the contents of the A register to be logically ANDed with the contents of the B register with the result appearing back in the accumulator. The original contents of the accumulator are lost, but the original contents of the B register are recirculated and saved. Executing this instruction produces the pure logical AND operation a bit at a time in serial form. This instruction can be used for masking and editing operations on data words.

Rotate A and E (RAE)
Function Code 110

Executing this instruction causes the contents of the accumulator (A) register to be rotated or exchanged with the contents of the E register. This instruction is similar to the RAB instruction except, of course, the E register is used instead of the B register. When this instruction is executed, the contents of the accumulator are shifted out serially into the E register while the contents of the E register are shifted serially into the accumulator. This instruction is valid only on computers containing the optional semiconductor memory expansion. On those computers without the semiconductor memory, this instruction causes binary 0 to be shifted into the accumulator. Thus, for these models the RAE instruction could be called clear accumulator (CAC).

Shift Accumulator (SHA)
Function Code 001

The SHA instruction causes the contents of the accumulator to be shifted one bit position either to the right or to the left, depending upon the state of bit 4 in the instruction word. If this bit is a binary 1, the shift is to the left. If this bit is a binary 0, the shift is to the right. The bits shifted out of the accumulator to the right or to the left are lost. Binary zeros are shifted into the register in either right or left shifts. This is a logical rather than an arithmetic shift where all bits of the accumulator are shifted, including the sign bit.

Skip-on-Condition (SKC)
Function Code 101

The Skip-on-Condition instruction causes the next instruction in sequence in the program to be skipped and the following instruction executed if a specified condition is met. The condition monitored in the Model 832 digital computer can be wired to sense practically any logical condition in the computer. It can be used to skip on accumulator Ø, accumulator minus, overflow, or any other logical condition that can be monitored. By using an external signal, the program sequence can be controlled by outside events. If the specified condition is met, the next instruction after the skip is passed over and the following instruction is executed. This is done by incrementing the program counter by one so that the next instruction in sequence can be skipped automatically. If the condition is not met, the normal program sequence will take place and the next instruction in sequence will be executed.

Decrement Accumulator (DCA)
Function Code 011

This instruction causes the contents of the accumulator to be decremented by one. In other words, one is subtracted from the contents of the accumulator when this instruction is performed. This instruction permits down counting operations in the accumulator.

Halt (HLT)
Function Code 111

Executing the halt instruction causes the computer to stop. When this instruction is executed, clock pulses are inhibited at the end of the execute phase to prevent further operation.

This basic instruction set for the Model 832 has been selected to make maximum use of the simple architecture of the computer. The various instructions have been selected to optimize computer operation with a small number of memory locations. Despite the small memory size, the flexible instruction set permits a fantastic variety of programs to be run.

This built-in flexibility also permits the instruction set to be modified to optimize the use of the computer for some particular application. Depending upon the application, it is possible to eliminate those instructions rarely used and replace them with other instructions of more value to the application. For example, the additional rotate register E may not be

needed, in which case the RAE instruction can be eliminated. In its place a clear accumulator or increment accumulator instruction may be substituted. Another example is to eliminate the subtract instruction and modify the logic so that the AND instruction is made into a memory reference instruction. Since subtract operations can be performed with the add instruction by manipulating the data properly, in some applications it may be more desirable to have the AND instruction as a memory reference. In this case, it frees the 010 function code and permits other generic instructions to be added.

Displays and Controls

All of your communications with the Model 832 digital computer will be via the displays and controls. The various switches and push buttons provide the input data and the control signals to the computer. The display lamps provide the outputs. In this section you will learn to use each of the controls and interpret the displays. Pay particular attention to this section, as proper operation of this computer is fully dependent upon your knowledge and understanding of the displays and controls.

DISPLAYS

The Model 832 digital computer has two incandescent binary light displays. Fig. 4A is the display which appears on the right of the computer. Fig. 4B appears on the left of the computer. Both are eight bits in length. They permit the computer operator to observe practically all of the most important logic states in the machine during its operation. Like any other binary light display, an *off* lamp indicates a binary 0 and an *on* lamp indicates a binary 1.

The 8-bit display on the right monitors the contents of the accumulator register. The bits are labeled Ø through 7. The data in the accumulator can be either a binary number or simply a bit pattern. For binary numbers bit Ø is the least significant bit position. Bit 7 represents the sign bit in 1's and 2's complement representation of negative numbers.

This display is fixed and will always monitor the accumulator contents. Notice that the display is divided into two 4-bit groups for convenience in viewing the contents of the accumulator as a hexadecimal number. For example, if the content of the accumulator is 0101 1100, this number would be represented by the hexadecimal digits 5C. Refer to the Appendix for hexadecimal conversion tables.

The left-hand display on the computer front panel permits the operator to monitor a wide range of program states.

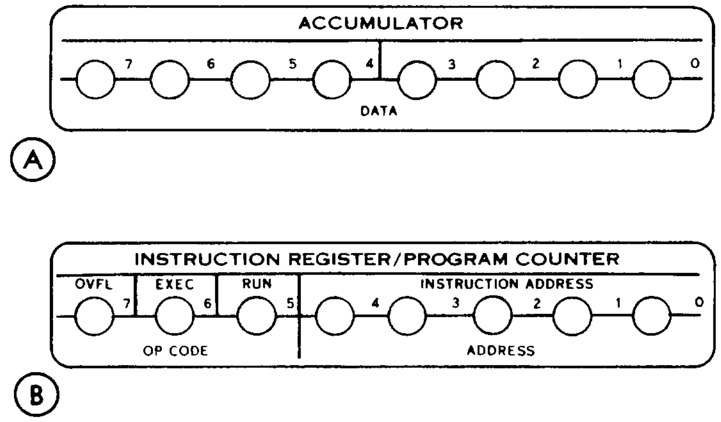


Fig. 4. Model 832 displays.

The part of the computer monitored by this display is determined by the display switch. With the switch in the down or IR position, the display monitors the instruction register. Notice that the labeling on the display divides it into upper and lower halves with the lower half marked according to the instruction word format. Bits 7 through 5 (7:5) of the instruction word represent the 3-bit op code. Bits 4 through Ø (4:Ø) designate the address referred to by the op code. These five bits represent the direct address referenced by the seven memory reference instructions.

For the operate (OPR) instruction bits 2 through Ø (2:Ø) represent the function code. Bit 4 designates the direction of shift operations when the operate instruction is being used. The instruction repetoire table on the front panel of the computer below the accumulator display gives complete details.

When the display switch is in the up or PC position, the left-hand display monitors the program counter which contains the instruction address or the address of the next instruction in sequence to be executed. Bit positions 4 through Ø (4:Ø) contain this 5-bit address. The remaining three bits, 5, 6, and 7, monitor three operating conditions in the computer. Bit 5 monitors the RUN state of the machine. When automatic operation is selected by the front panel switches, the run light will be on while the clock is generating clock pulses. When the computer is reset or is caused to halt due to the execution of a halt instruction, the RUN lamp will be off.

The bit 6 position is the execute (EXEC) state which tells the operator when the computer is in the execute phase. During the Fetch state the EXEC lamp will be off. During the EXEC phase the bit 6 lamp will be on.

Bit 7 monitors the overflow state of the arithmetic section of the computer. During any arithmetic operation involving additions and subtractions, if there is a carry out of the bit 7 position in the accumulator, the overflow lamp will light. There are numerous ways of determining whether a true overflow condition exists when this lamp lights. The interpretation is up to the operator depending upon the data word format he is using. The OVFL indication simply shows a carry out of the bit 7 position and it may be interpreted as desired.

There are several special features about the displays which you should understand. First, whenever the computer is in the reset state all of the lamps in both displays will be off.

Whenever power is applied to the computer, the flip-flops in the registers monitored by the displays can come up in any random state. Therefore, the displays will show this random state. By depressing the reset push button, the displays can be cleared.

The displays do not change when the clock in the computer is not running. When the clock is running at low speed, where clock pulses are approximately two pulses per second, you can observe the data changing in the registers by watching the lights flash on the display. The slow clock speed is slow enough to permit you to readily monitor the changes in the display for each clock pulse. However, when the computer is running at high speed on the fast clock, the display lamps will all be on and will glow dimly.

CONTROLS

The control switches and push buttons on the front panel of the Model 832 digital computer are for the most part self-explanatory. In this section we describe the exact function of each of these controls. See Fig. 5.

Power. The power switch is a red control used for applying 120 VAC to the computer power supply. No power is applied to the computer with the switch in the down or off position. With the switch in the up or on position, the power supply is enabled.

Clock. This two-position white switch selects either the automatic or single step modes of clock operation. With the switch in the up or AUTO position, clock pulses will be supplied to the computer circuitry by the main systems clock. With the switch in the down position the computer is in the single step mode. Clock pulses are applied to the computer by the single step manual push button located in the lower right-hand corner of the computer panel.

Speed. This white switch selects either a low or high frequency clock. It is effective only in the AUTO clock mode. With the switch in the down position, a low frequency clock is selected. The low speed clock frequency is approximately two pulses per second. With the speed switch in the up position the high frequency clock is selected. This clock runs at approximately 250 kHz.

Display. This white switch selects the circuitry displayed by the left-hand display on the computer panel. With the switch in the down or IR position, the left-hand 8-bit display shows the contents of the instruction register. With the switch in the up or PC position the display monitors the program counter as well as the states of the overflow, fetch/execute, and run/stop flip-flops.

Switch Register. These eight white switches form a switch storage register from which data can be entered directly into the accumulator. By setting these switches to a desired number, this number may be loaded into the accumulator by depressing the LOAD push button. The bit switches are labeled Ø through 7 from right to left in accordance with the accumulator numbering where the right-hand switch is the least significant bit. A switch in the down position represents a binary 0 and a switch in the up position represents a binary 1.

Load. This black push button causes the contents of the switch register to be loaded into the accumulator. The moment this switch is depressed, the accumulator contents will be equal to that of the switch register. The accumulator display will show the result.

Reset. This red push button resets the instruction register, program counter, accumulator register, run/stop flip-flop and the timing counter in the computer. This button is a complete systems reset. This is the normal beginning state for the computer operation. This reset button control is completely asynchronous in that any time that it is depressed, it will cause the clock to stop and all registers to be reset regardless of the current state of operation.

Start. This red push button sets the run/stop flip-flop and causes clock pulses to be generated. If the computer is running, pressing the Start push button will reset the run/stop flip-flop, allowing you to halt the computer. Operating the Start push button once more will allow the program to continue.

Step. This black push button generates a single clock pulse each time it is depressed. This button functions only when the clock switch is in the step mode. This step control permits the operator to generate clock pulses at any desired manual rate. This convenience

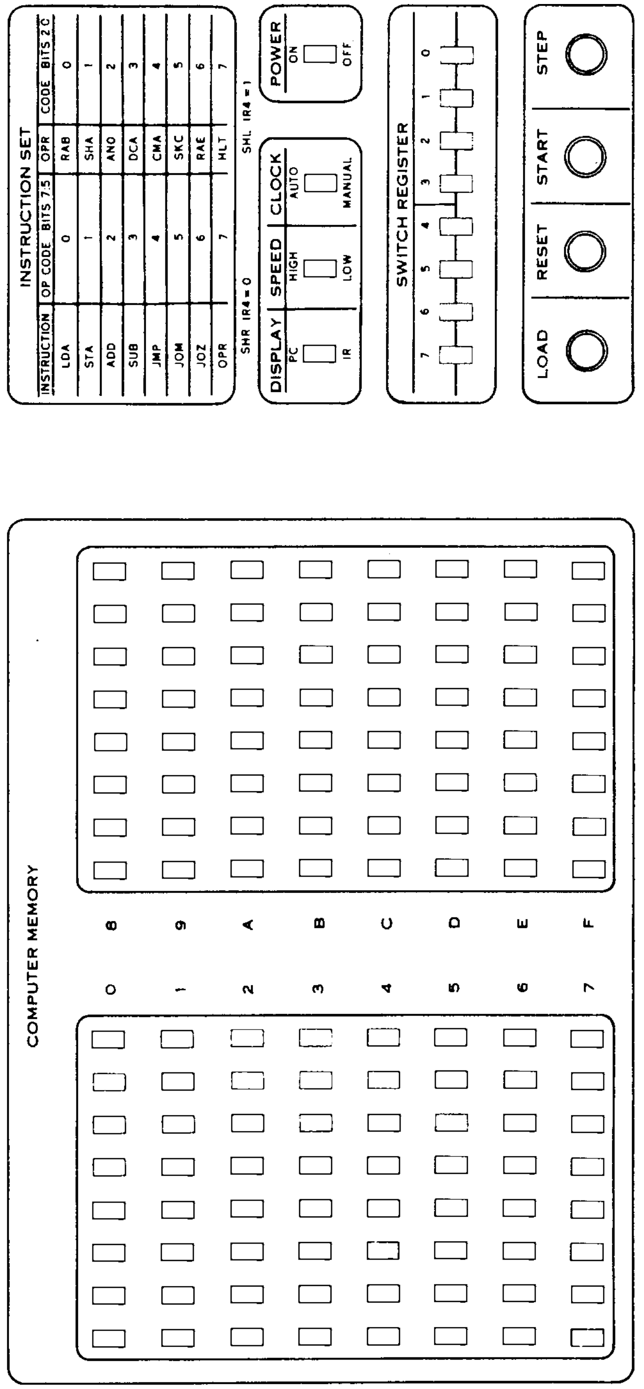


Fig. 5. Controls on the Model 832.

permits the operator to sequence the program a clock pulse at a time so that he can view the results of each and every clock pulse change.

Computer Memory. The 128 switches on the left-hand side of the computer front panel are not really controls in the normal sense. These switches make up one-half of the computer memory of the Model 832. The switches are grouped as sixteen 8-bit words numbered Ø through 9, A through F in hexadecimal form. These are the hexadecimal addresses for these words of the memory. A switch in a down position represents a binary 0 and a switch in the up position represents a binary 1. These switches may be set to store instructions and data in the computer memory. Normally the computer, after it has been reset, will fetch the first instruction from memory location 0. For this reason the first instruction to be executed should be located in memory location 0.

Programming and Operating The Computer

The Model 832 digital computer, like any digital computer, must be properly programmed before it can perform any useful function. All standard digital computer programming techniques are fully applicable to the Model 832, as long as the programmer confines himself to the given instruction set. Knowing the instruction set and basic computer programming fundamentals you can program virtually any problem that will fit within the memory. The problem to be solved or function to be performed is analyzed and broken into a number of simple steps. Each of these steps is then implemented with an instruction or group of instructions. The instructions are then written in sequence to form a complete program. The program is entered into memory and the computer is operated to produce the final result. In this section we are going to give you several suggestions which will be quite useful when you begin programming and operating the Model 832 digital computer.

PROGRAMMING

Programming fundamentals are covered completely in experimental manuals 10K and 11K. The experiments outlined in these manuals permit you to demonstrate all of the most important programming fundamentals on the Model 832 digital computer. You should immediately learn or review the programming fundamentals outlined in these manuals. Once you have performed the programming experiments in these manuals you will be completely familiar with the use of the Model 832. You will then want to try your hand at programming a variety of problems on the computer.

The Model 832 is programmed in machine language. That is, as the programmer you will enter the binary data representing each instruction and data word by manipulating the switch register and memory switches. However, in writing programs for the Model 832 we recommend the use of hexadecimal notation. Once you are intimately familiar with the hex system you can readily convert between binary and hex numbers. We recommend that you memorize the binary and hexadecimal equivalents for all 4-bit combinations. A table outlining the conversion equivalents is given in the Appendix. You should write the programs in hexadecimal notation and then you will have no difficulty in converting them into their equivalent binary numbers when you enter the program into the computer.

Programming the Model 832 is most easily accomplished by using a programming coding form similar to that shown in Table II. The left-hand column lists the memory address of each location in the memory. This address is given in hexadecimal notation.

Memory addresses Ø through 9 and A through F are the memory locations in the programmable read-only memory, accessible only by the front panel switches. Memory locations 1Ø through 1F are the sixteen locations in the bipolar semiconductor memory.

MEMORY LOCATION	CONTENTS	HEX CONTENTS	COMMENTS
0	LDA 6	06	42
1	ADD C	4C	42 + 27 = 69
2	SUB 1E	7E	69 - 35 = 34
3	STA 12	32	
4	HLT	E7	
5			
6	42	2A	
7			
8			
9			
A			
B			
C	27	1B	
D			
E			
F			
10			
11			
12	(34)	22	RESULT
13			
14			
15			
16			
17			
18			
19			
1A			
1B			
1C			
1D			
1E	35	23	
1F			

Table II. Recommended program coding format.

Adjacent to the memory address is another column designated Contents. This column describes the instruction or data word stored in each location. In this column is written the mnemonic for each instruction in the program along with the hexadecimal memory address referred to by those memory reference instructions used and the decimal value of data numbers to be used.

The next column to the right gives the 2-digit hex value of each instruction and data word. You can use the decimal-to-hex conversion chart in the Appendix to help you arrive at the values here. Then using this column and mentally converting to binary, you enter the program into memory.

The right-hand column in the programming form is a Comments column. It permits you to make specific notations about the operation taking place at each instruction step.

To illustrate the operation and programming of the computer, an example problem is shown in Table II. As you recall, the Model 832 always looks for the first instruction to be executed in memory location Ø. Here the computer first interprets the LDA 6 instruction. This tells the computer to load the accumulator with the contents of memory location 6. As you can see in Table II, location 6 contains the number 42.

Once the content of memory location 6 is loaded into the accumulator, the next instruction is executed. There is an ADD C instruction in memory location 1. It tells the computer to add the contents of memory location C to the contents of the accumulator and store the result back in the accumulator. The content of memory location C is the number 27. This instruction will cause 27 to be added to the number 42 now in the accumulator with the result, 69, being stored back in the accumulator. Note that the Comments column in Table II indicates this for reference.

The next instruction in sequence is then executed. This is an SUB 1E instruction in memory location 2. This causes the contents of memory location 1E, which is the number 35, to be subtracted from the contents of the accumulator, 69. The difference, 34, will appear in the accumulator.

The next instruction in sequence, stored in memory location 3, is then executed. This is an STA 12 instruction. It tells the computer to store the contents of the accumulator in memory location 12. Therefore, the contents of the accumulator, 34, designated in parenthesis, are stored in memory location 12. The next instruction, memory location 4, is then executed. This is an HLT or halt instruction so the computer operation stops. The final result of the calculation (the number 34) is still in the accumulator and will be displayed on the accumulator lamps.

It is a good idea to use the format shown in Table II to program each one of the problems you wish to solve. By writing a neat program in this form you will have complete documentation for every problem that you solve.

There are many common operations that you may wish to perform on the Model 832 digital computer which are not fully implemented by the instruction set. Typical of these are multiply and divide operations or logic operations such as the OR function. These functions can be readily performed but must be done by program subroutine. Several of the most popular subroutines are given in the Appendix. Other methods for performing these operations can be implemented, but the ones given in the Appendix are recommended because they use the least number of memory locations, thereby saving as much space as possible for other instructions.

OPERATING THE COMPUTER

Once you have written the program you will enter it into the computer memory and cause the computer to execute that program. In this section we will show you exactly how to do this.

Loading the Memory. Loading the programmable read-only memory is extremely simple. This is only a matter of setting the switches to the proper binary bit positions as they correspond to the hexadecimal contents of the words in your program. Simply refer to the hexadecimal Contents column of the program coding form, translate it into the equivalent binary numbers and set the switches on the appropriate memory word. If the program is short enough to be contained within sixteen words, the entire program can be stored in the switch memory. If the program requires more than sixteen locations or if store-into-memory operations are required, then it will be necessary to load the semiconductor read/write memory. If the program is long enough and requires using the semiconductor memory, the semiconductor memory should be loaded first. This is done by using a simple programmed subroutine set into the memory switches. Once you have entered all of the required data into the semiconductor memory using this program, the remaining part of the program can be set into the switch memory as described earlier.

The procedure for loading the semiconductor memory is to write a simple program that will cause the desired data to be loaded into a specific memory location. This is done by first loading the accumulator with the instruction or data word to be stored in a specific memory location. This is done by setting the switch register on the front panel to the desired number and depressing the LOAD push button. This will cause the contents of the accumulator to be the same as that of the switch register. Then in memory location Ø you will store an STA instruction that will cause the contents of the accumulator to be stored in a memory location designated by the five address bits in that STA instruction word. The five address bits of the STA instruction word are set to a memory location in the 10000 to 11111 range as desired. In memory location 1 a halt instruction is written. Once the accumulator has been loaded with the appropriate word and the address portion of the STA instruction set into the location where the word is to be stored, the Start button is depressed. This will cause the computer to execute the program and store the word in the accumulator and the desired memory location and then halt automatically. The computer should be in the high speed automatic clock mode for this purpose. Once the word has been loaded, the Reset button is depressed to clear the accumulator and again ensure that the computer will execute the instruction stored in memory location Ø when the Start button is depressed.

The next word to be loaded is set into the switch register and then loaded into the accumulator with the Load push button. The desired address is entered into the STA instruction by the front panel switches from memory location 0. The Start button is then depressed to cause the program to execute the store instruction and place the desired word in the designated memory location. This procedure is repeated until all of the necessary memory locations are loaded. Then the remaining part of the programming operation can be completed by setting the memory switches to the desired positions.

Once you have loaded the semiconductor memory you may wish to check the contents to be sure that you have made no mistakes. Again this can be done by writing a simple subroutine that will cause each memory location in the semiconductor memory to be displayed on the accumulator lamps one at a time. By using the slow speed clock mode for this operation, you can readily observe the contents in each memory location of the semiconductor memory and check it against the programming coding sheet.

The simple subroutine shown will perform this operation.

```
0      LDA 1Ø
1      LDA 11
2      LDA 12
3      LDA 13
4      LDA 14
5      LDA 15
6      LDA 16
7      LDA 17
8      LDA 18
9      LDA 19
A      LDA 1A
B      LDA 1B
C      LDA 1C
D      LDA 1D
E      LDA 1E
F      LDA 1F
```

You can program this subroutine into the programmable read-only memory. Then depress the Reset button to clear the machine. Put the computer in the automatic operate mode on low speed. Then depress the Start button. The contents of each of the memory locations 1Ø through 1F will be displayed one at a time during the fetch phase. Since there is no HLT instruction, you will have to press the Reset button to stop the computer.

As the binary content of each memory location is displayed, you can mentally convert it into hexadecimal notation and compare it to the hexadecimal contents designated by the program.

It is important to remember that the semiconductor memory is a volatile memory and all memory data will be lost if computer power is removed. If you wish to retain a program, simply leave the computer power on. The computer draws very little power, and therefore can be left on indefinitely. No damage will occur with continuous operation.

Operating the Computer. Operation of the computer is simply a matter of depressing the Reset button and then the Start push button. This will cause the program to be executed instantaneously if the high speed automatic clock mode is selected. By selecting the low speed clock in the automatic mode, you can watch the program being executed. You can see each instruction word loaded into the instruction register and the contents of the accumulator as each instruction is carried out. By following your programming coding sheet as the computer is operating, you can see how each instruction is loaded and executed.

To execute the program yourself, a clock pulse at a time so that you can stop at any given point, you should select the step mode of operation. As you depress the Step push button, one clock pulse will be generated. Remember that it takes eight clock pulses to cause any given computer operation to take place. You can count the number of clock pulses as you depress the button. In starting the program you will initially depress the Reset button to clear the machine. Then you depress the Step push button eight times. This causes the first instruction to be fetched and loaded into the instruction register. The next eight clock pulses cause the instruction to be executed. You should remember that all operations take place with eight clock pulses and it is convenient for you to count these clock pulses as you watch each instruction being fetched and executed. If you do not count the clock pulses in 8-bit groups, you can end up at some intermediate point where the results being displayed on the lamps are essentially meaningless.

There are several useful tips that you may wish to use in programming and operating the Model 832. First of all, most programs that you will write will start with the LDA instruction. The reason for this is that it is necessary to initially load the accumulator with data in order to perform the necessary operations on that data. The first instruction in your program, therefore, will almost always be an LDA instruction. However, in order to conserve memory space it is a good idea to eliminate as many instructions for your program as possible. The LDA instruction can generally be eliminated by using the switch register and Load push button. Before you start the program, you load the accumulator with the necessary data to be used simply by entering it into the switch register and depressing the Load push button. This eliminates the initial LDA instruction and permits the first instruction in your program to be some other operation.

Remember, if you use this procedure, it may still be necessary to provide a storage location for the word you are placing in the accumulator manually, especially if this word is referred to elsewhere in the program. If it is not, then you save an additional memory location. Efficient use of memory is good programming practice and practically a necessity with the Model 832.

Of course, you may wish to start the program at some memory location other than memory location Ø. In such a case, simply load a jump (JMP) instruction in memory location Ø that will cause the computer to branch to the first instruction of the program anywhere in memory.

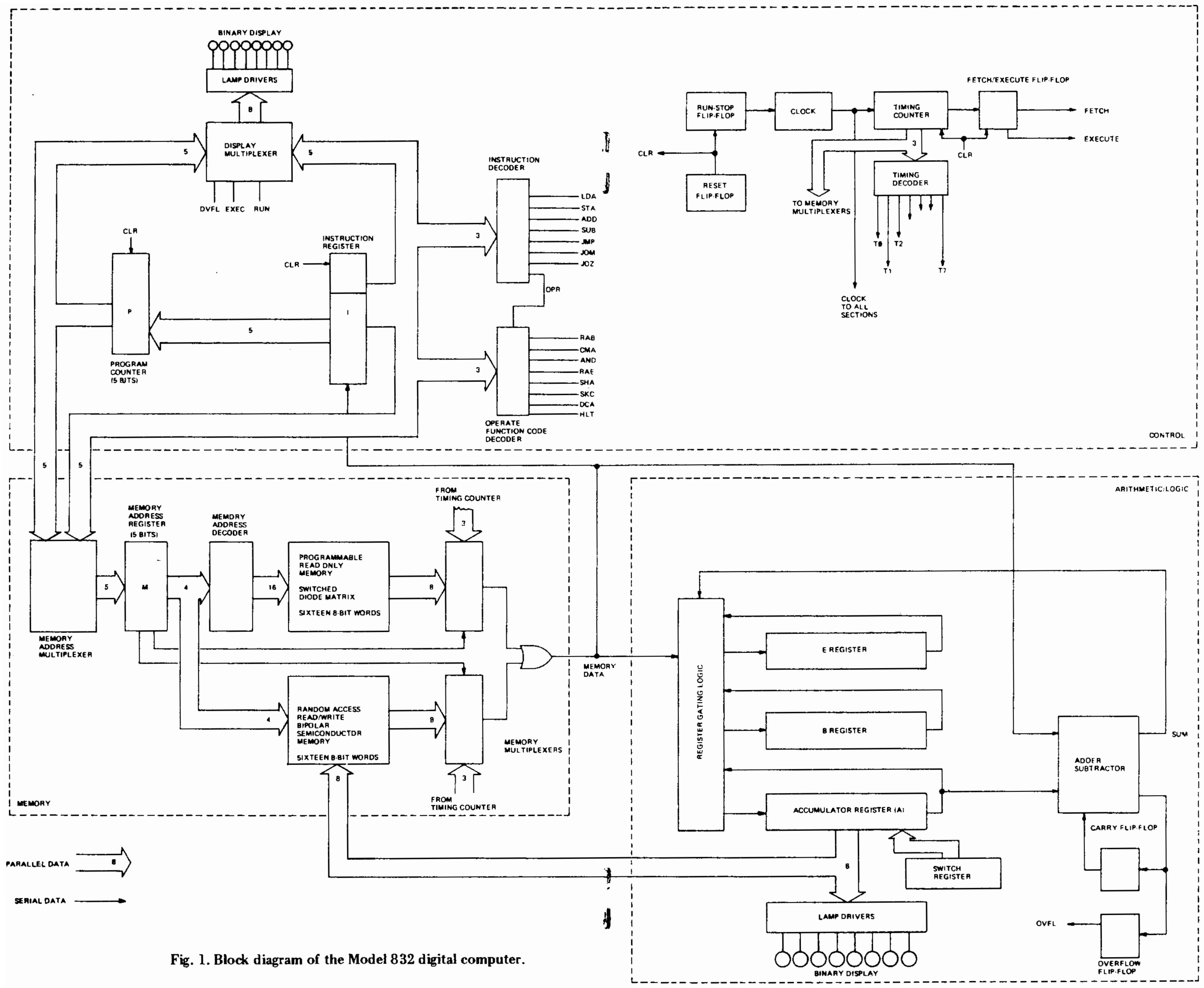


Fig. 1. Block diagram of the Model 832 digital computer.

Logic Circuitry Analysis

All of the logic circuitry for the Model 832 digital computer is contained on ten printed circuit boards. The circuitry in the Model 832 is made up predominantly of TTL SSI and MSI integrated circuits. Several DTL integrated circuits and discrete component circuits are also used. In this section we are going to analyze in detail the circuitry contained on each of the ten printed circuit boards in the computer. The operation of the power supply will also be discussed.

The computer circuitry is divided into four basic groups: the memory, control, arithmetic, and power supply. The computer memory and associated circuitry is contained on three circuit boards labeled M1, M2, and M3. The control circuitry is contained on three circuit boards labeled C1, C2, and C3. The arithmetic circuitry is contained on two printed circuit boards labeled A1 and A2. The remaining two boards are lamp drivers. The power supply circuitry is wired to terminal strips on the bottom plate of the computer cabinet. Fig. 6 shows the position of each of these circuit boards in the computer.

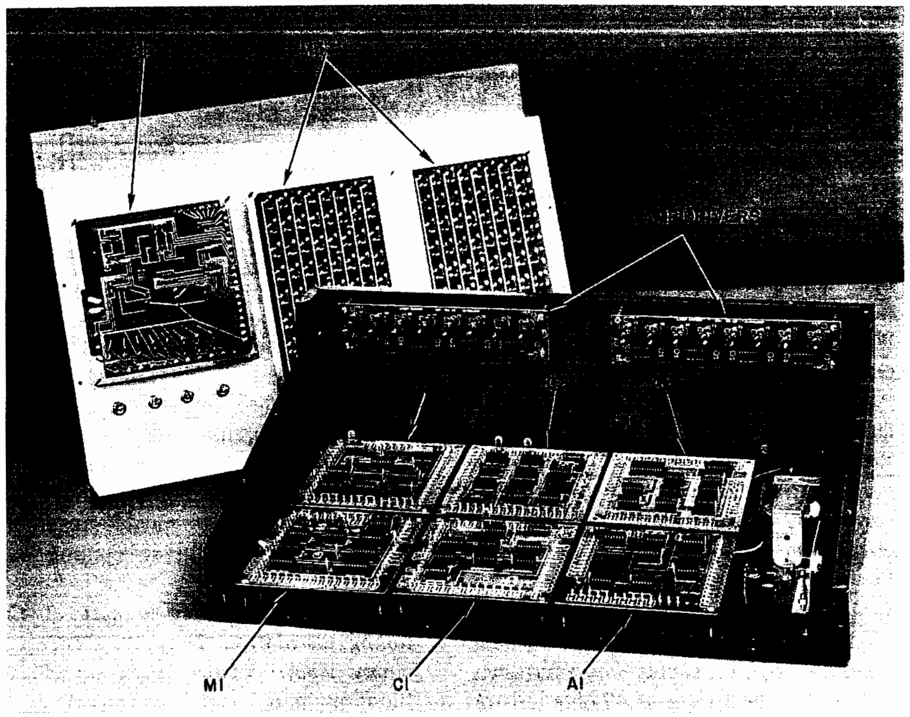


Fig. 6. Circuit board identification inside the computer.

Except for the lamp driver boards and the power supplies, the circuitry for each printed circuit board is contained on a large separate logic diagram. Refer to these as each board is discussed. A complete list of signal names and their meanings appears in the Appendix. Refer to this for further clarification of each as it is referred to in the text.

THE LAMP DRIVERS

There are two lamp driver printed circuit boards in the Model 832 computer. Each contains eight incandescent lamps and the associated driver circuitry. Fig. 7 shows the circuitry contained on each board. The lamp is a type 49 incandescent lamp rated at 2 volts at 60 ma. It has a bayonet base and mounts in the socket on the printed circuit board. The lamps are easily removed for replacement.

Each lamp is driven by a DTL lamp driver. These lamp drivers are contained within a type 15844 integrated circuit. This is a dual 4-input NAND gate with open collector output. The inputs are all tied together to form a single input. Whenever this input is a binary 1, the output transistor in the DTL gate is turned on. This supplies current to the lamp. Whenever the input is a binary 0, the output transistor is cut off and the lamp is out. There are two such lamp drivers in each of the four integrated circuits on the printed circuit board. The integrated circuits are powered by the +5 volt supply in the computer. The lamps themselves are supplied by a separate lamp driver power supply which furnishes approximately 5 volts. A 47-ohm resistor in series with each lamp drops the voltage for the lamps to about 2 volts.

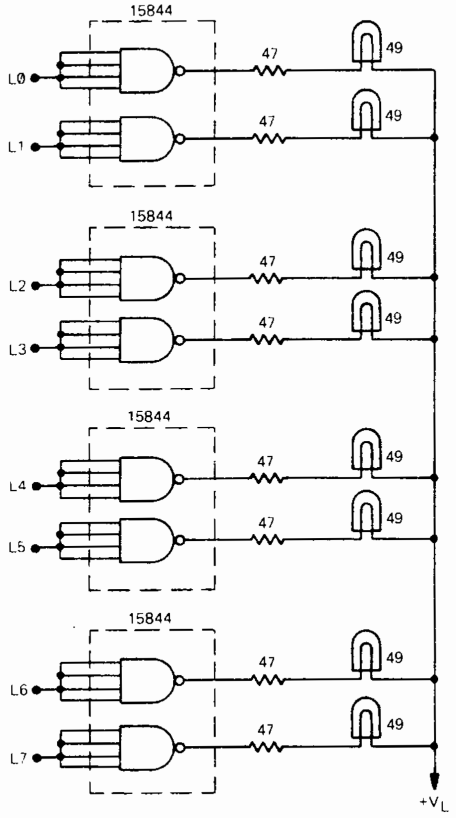


Fig. 7. Lamp driver circuits.

THE M1 CIRCUIT BOARD

Refer to the large separate diagram outlining the circuitry on the M1 circuit board. This board contains the memory address register, the memory address decoder for the programmable read-only memory, and the timing counter and distributor.

The timing counter is part of the timing distributor. This is a three-stage binary ripple counter made up of J-K flip-flops in IC5 and IC6. Outputs A, B, and C are applied to a type 7442 decoder, IC7. This decoder monitors the counter output and generates a series of eight output pulses designated TØ through T7. The outputs of the decoder are active low, meaning that the enabled output line is low while all others are high. The complement signals are made available by inverting them with the gates in IC8. Notice that only the timing signals TØ, T1, T2, and T7 are used.

The output of the C flip-flop in the timing counter drives another flip-flop designated as the fetch/execute flip-flop. This flip-flop, designated D, determines the operate state of the computer and is toggled once every eight clock pulses. Clock pulses are applied to the A flip-flop in the timing counter. Fig. 8 shows the waveforms for the timing counter, timing distributor and associated circuitry.

The memory address register is made up of IC3 and IC4. IC4 is a 7475 quad-latch. This is a 4-bit register made with type D flip-flops. IC3 is a 7400 IC connected as a type D flip-flop. Together the two IC's form a 5-bit storage register. This register holds the address of the word in memory to be addressed. The output of IC3 designated MA4 is the most significant output bit.

The M register receives its input data on lines ADRØ through ADR4. Data is loaded into the register during the TØ time interval produced by the timing distributor. Note the connection between the TØ output of IC8 and IC3 and IC4. During TØ time, the data appearing on the ADRØ through ADR4 lines is strobed into the register.

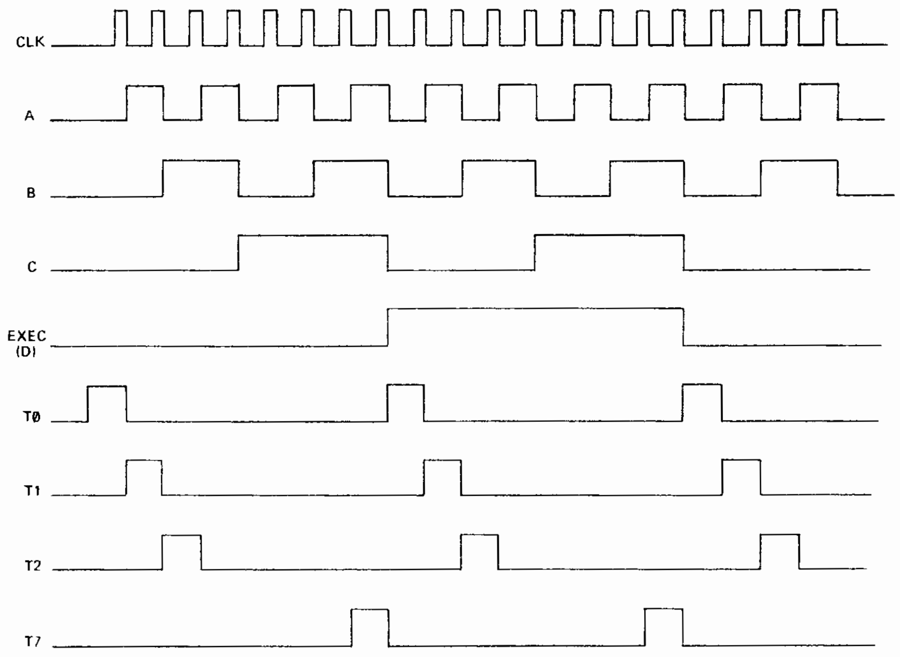


Fig. 8. Pulses for timing counter and distributor.

The four least significant bits in the M register are designated MAØ through MA3. These are applied to two 7442 IC's, IC1 and IC2. These decoder IC's look at the 4-bit address and cause one of the sixteen memory word select lines to go low. These output lines are labeled WØF through WFØ and are used to select one of the sixteen switch-diode matrix memory locations on the M3 circuit board. All of these output lines are high except for the selected one which is low. Notice that the MAØ through MA3 lines are also made available externally. These are used to address a word in the bipolar semiconductor memory as well.

THE M2 CIRCUIT BOARD

Refer to the large separate diagram for the M2 circuit board. You will need to refer to the diagram for the M1 circuit board as well while discussing the circuitry on the M2 board. The M2 board consists of two memory data multiplexers, IC9 and IC14. These are used to convert parallel data into serial data. Also on this board are the two bipolar semiconductor memory IC's. Each contains 64 flip-flops arranged as sixteen 4-bit words. Combining the two gives us a total of sixteen 8-bit words. Also on the M2 circuit board is the E register, an 8-bit shift register, IC11, and the associated circuitry. This register is used as an addressable memory location in those models of the 832 computer not containing the bipolar semiconductor memory. In those models that do contain the semiconductor memory, the E register is used as an auxiliary storage register that can be accessed by an RAE instruction.

The memory data multiplexer, IC9, looks at the 8-bit parallel output of the switch-diode matrix memory. Whenever MA4 is a binary 0, this multiplexer is enabled. The A, B, and C lines from the timing counter on the M1 circuit board sequentially enable the gates in the multiplexer so that the eight parallel input lines are sampled sequentially and fed to the output. The output, SD1, is fed to a NOR gate, IC10, where the signal MDA (memory data) is generated. This is the serial memory data output line.

The address lines of the semiconductor memory (IC15 and IC16) are tied in parallel and are fed from the memory address register bits MAØ through MA3. These two integrated circuits contain their own address decoding. The parallel output of the bipolar semiconductor memories is fed to the 8-bit memory data multiplexer IC14. Here the parallel output data is converted to a serial data word. Multiplexer IC14 is addressed when the $\overline{\text{MA4}}$ line is low. Notice that the parallel output lines from the semiconductor memories require external 4.7k pull-up resistors.

Multiplexer IC14 converts the parallel data from the semiconductor memory into a serial data word labeled SD2. When this is connected by a jumper to the M2 point on the NOR gate in IC10, the serial data can also appear at the MDA output.

To store data into the bipolar semiconductor memory, the word to be stored is applied to the eight input lines AØ through A7. These lines come from the eight parallel output bits of the accumulator register. A gate in IC12 on the M2 circuit board generates a signal that causes the data present at the AØ through A7 lines to be stored. Three signals, EXEC, STA, and T2, are required to store the data. This means that the computer is in the execute mode, a store instruction is being decoded, and timing signal T2 occurs. This generates signal STE which causes the data to be stored.

Note that IC14, IC15, IC16 and the associated components make up the optional memory expansion package. The basic Model 832 does not contain these components but they are available and readily added at a later date. They are particularly valuable in expanding the usefulness of the computer.

The E register on the M2 circuit board is made up of IC11 and IC13. IC11 is an 8-bit shift register with parallel output lines EØ through E7. The ECLR input permits the register to be cleared to zero when a positive-going signal is applied here. IC13 is a quad two-input NAND gate IC connected to permit the data in the E register to be recirculated or new data to be written in. The SMC line is the serial memory control line that determines whether the word in the register will be recirculated. If the SMC line is a binary 1, the data in the memory will be recirculated, shifting it out of the EØ line through the gates in IC13 to the register input on pin 1. The other gates are thereby inhibited. If the SMC line is a binary 0, the EØ input to the gates is inhibited and data will not be recirculated. Instead, the gate fed by line AØN will be enabled. New data can then be shifted into the 8-bit shift register on the AØN terminal. This data will come from the AØ line, the LSB position of the accumulator register.

To shift data out of the E register, the MA4 line must be high. This will enable the associated gate in IC10 and cause the data to appear at the SWM terminal. If a jumper exists between SWM and M2, the data stored in the E register will be shifted out and will appear on the MDA line. In the basic Model 832 with no semiconductor memory, the E register is addressed by any address containing a one bit in the MA4 position.

THE M3 CIRCUIT BOARD

To understand how the switch memory works, refer to the memory printed circuit board logic diagram that you received in an earlier text. Notice the large diode-switch matrix. Each bit position consists of a switch and a diode connected in series to one of the 128 points formed by the 8-by-16 matrix. Eight lines feed the input node connections to the 15844 IC NAND gates used.

There are two 4-input gates with an open collector output in each IC. We are using external 1k-ohm load resistors for these NAND gates on the memory printed circuit board. We are not using the input diodes on the NAND gate. These are connected together and enabled by a +5 volt line on the memory circuit board, as you can see in the figure. However, we are using the node input. The node inputs are pins 3 and 11, respectively. As you can see, the diodes in the matrix are connected to the gate node and are used to enable the gates, depending upon how the switches are set.

To fully understand how the memory works you should recall the method of operation of the NAND gate. If any one of the diode inputs to the NAND gate is brought to ground (or made a binary 0), the output of the NAND gate goes high. If all of the diode inputs to the NAND gate are open or at the binary 1 level, the NAND gate output will be a binary 0.

Now refer to the memory board diagram. When the WØF address input line goes low, it will bring the cathode side of the diodes connected to it to ground and will, therefore, force the output of the corresponding NAND gate high if the switch in series with the diode is closed.

Consider the NAND gate associated with output SØ in the figure. Notice the diode connected to the switch attached to pin 11, the node of the NAND gate. The cathode of the diode is connected to the WØF input. If this switch is closed and WØF is grounded, indicating that word Ø is addressed, the SØ output will be a binary 1. However, opening the switch will simply leave the node open and the NAND gate will have all of its normal diode inputs enabled. This will cause the SØ output to go low. As you can see, with the switch closed the NAND gate output is forced high and a binary 1 is produced. With the switch open a binary 0 is produced. This same action occurs for each of the eight bit positions for the word Ø in the memory.

As you refer to the memory logic diagram, keep in mind that all other address input lines at this time are at the binary 1 level. For that reason none of the other switches in the memory affects the state of the output. If a switch should be open, it will have no affect on the circuit. If a switch happens to be closed, it will also have no affect since all of the other address lines are at a binary 1 level at this time. These conditions enable the NAND gates in such a way as to permit only the diodes and switches in the addressed word to control the output state.

THE C1 CIRCUIT BOARD

Refer to the separate logic diagram for the C1 circuit board. This board contains the instruction register (I) and the instruction decode circuitry. The instruction register is a 74164 8-bit shift register, IC24. It is loaded serially from the memory. The input to the register is the MDA line, as shown. The register is loaded during the fetch phase of computer operation when the instruction to be executed is shifted into the instruction register. Clock pulses applied to the gate in IC22 are enabled by the fetch signal from the fetch/execute flip-flop. This causes the shift register to shift in the data from the memory. This register is also cleared to zero by the signal on the CLR line when the front panel Reset button is depressed.

The eight parallel outputs of the instruction register, IRØ through IR7, are fed to the C2 circuit board where they are applied to a display multiplexer for display on the eight display lamps. The instruction register outputs are also applied to two instruction decoders, IC21 and IC23. These are type 7442 IC's which decode the various instruction formats in the computer.

IC21 is an octal decoder that decodes the three op code bits of the instruction word stored in positions IR7, IR6, and IR5. The eight basic memory reference instructions are decoded by this decoder. The output lines are available for use in controlling logic circuitry in other parts of the computer. Six of the active low outputs of the instruction decoder are made active high by the hex inverter, IC18.

The two 4-input NOR gates in IC17 are used to generate two special logic signals designated JMZS and ASD. The JMZS signal is a logic one if any one of the three jump instructions is specified or if a store operation is indicated. The ASD signal is generated if an add, subtract, or decrement instruction is given.

When the operate (OPR) instruction is decoded by IC21, pin 9 on this IC goes low. This enables the function decoder, IC23. This decoder looks at bits IRØ, IR1, and IR2 in the instruction register and then decodes the various operate

functions. The decoder output lines are made active high by the inverters in IC19 and IC20.

Gates in IC22 are also used to generate special logic signals. One of these is designated ROA, which is a binary 1 if an RAB or AND instruction is decoded.

THE C2 CIRCUIT BOARD

Refer to the large separate diagram for the C2 circuit board. This board contains the program counter (P), the memory address multiplexer and the display multiplexer along with the associated logic circuitry.

The program counter is made up of two 74193 IC's. These are 4-bit binary counters that can be parallel loaded. All four bit positions in IC31 are used, but only the least significant bit position of IC28 is used, thereby making a 5-bit program counter. This counter can be cleared by the CLR signal when a positive-going level is applied.

The program counter is incremented by the clock signal when it is enabled by various other logic levels applied to the control gates in IC29. The program counter can be incremented in two ways. First the program counter is normally incremented each time an instruction is executed so that it contains the address of the next instruction in sequence to fetch. This is done during the execute phase and the TØ time interval. As soon as the instruction is fetched, the computer goes into the execute phase and immediately the program counter is incremented by a clock pulse during the TØ time interval. In order for the program counter to be incremented, the $\overline{\text{BRH}}$ signal must be high, indicating that a jump instruction is not being executed.

The other way that the program counter can be incremented is during a skip instruction. The SKC input to the gate in IC32 is high when a skip instruction has been decoded. The CDX input designates the condition upon which the skip instruction will be executed. This can come from any number of logic sources. However, if it is a binary 1, then a skip operation is designated and the program counter will be incremented an additional time. The program counter will still be incremented during the execute and TØ times as usual. However, when the skip instruction is executed, it will be incremented again during the T2 time interval. This will cause the computer to skip the next instruction in sequence and proceed to the next.

During a jump instruction execution the program counter is not incremented. Instead, the jump instruction causes the computer sequence to branch to a memory location specified by the address portion of the jump instruction in the instruction register. A jump instruction is sensed by the circuitry in gates IC25 and IC26. The $\overline{\text{JMP}}$ input designates an unconditional jump which causes the line BRH to be high, indicating a jump. The JOZ and ACZ signals indicate a jump on zero operation. The JOM and A7 signals indicate a jump on minus accumulator. If any of these conditions is met, the BRH signal will be a binary 1 and a jump operation will be initiated. This will inhibit the normal incrementing of the program counter by having the $\overline{\text{BRH}}$ signal inhibit the gate in IC29 from passing clock pulses to the program counter. Instead, the program counter is parallel loaded during the execute phase and at T2 time. The gate in IC26 that monitors these signals generates a parallel load program counter signal, LDPC, that causes the word stored in the address portion of the instruction register to be loaded into the program counter. This 5-bit word comes from lines IRØ through IR4 on the C1 circuit board.

The IRØ through IR7 lines from the instruction register on the C1 circuit board are also applied to the display multiplexer. The display multiplexer is made up of two 74157 IC's, IC33 and IC34. These quad two-input multiplexers permit a single set of lamp drivers to alternately monitor two 8-bit input words, one of them being the 8-bit instruction word. Signal LS controls which of the two 8-bit input words is monitored. If LS is low, the multiplexer display output will pass the signals representing the contents of the instruction register. These will be applied to the left-hand lamp driver board in the computer. If the LS signal is a binary 1, the contents of the program counter as well as three other logic signals are monitored. Bits PCØ through PC4 are displayed as well as the execute (EXEC) signal, the run (RUN) signal, and the overflow (OVFL) signal.

The memory address multiplexer is made up of IC27 and IC30. IC27 is a quad two-input multiplexer; IC30 is a single two-input multiplexer. Together these two integrated circuits form a 5-bit two-input multiplexer that drives the ADRØ through ADR4 input lines to the memory address register on the M1 circuit board. This multiplexer permits either of two 5-bit address words to be applied to the M register. The control for this multiplexer is the fetch signal. During the fetch phase of the computer operation, the content of the program counter is passed through the memory address multiplexer to the ADRØ through ADR4 lines. This means that the address for the instruction to be executed is taken from the program counter. The program counter always contains the address of the next instruction to be executed. During the execute phase the FETCH input signal is a binary 0. Therefore, the multiplexer then causes bits IRØ through IR4 to be applied to the ADRØ through ADR4 lines. During the execute operation a memory reference instruction will most probably be executed; therefore, the address of the word to be used in the operation is taken from the address portion of the instruction word in the I register. If an OPR instruction is designated, the function code stored in the instruction register is transferred to the M register. However, it has no meaning in an operate instruction operation, therefore no significant operation takes place. The word loaded into the M register is irrelevant and will be removed during the next fetch cycle.

THE A1 CIRCUIT BOARD

The AI circuit board contains the 1-bit serial binary adder-subtractor and the auxiliary B register plus associated gating circuitry. Addition and subtraction operations in the Model 832 digital computer are carried out by the type 7480 IC46. This single bit adder accepts two inputs and generates the binary sum output and an appropriate carry. This carry is stored in a flip-flop whose output is fed back to the carry input. The flip-flop stores any carry signal that occurs during the addition so that it may be added to the next most significant bit of the words being added. Two inputs are applied to the adder. These are shifted a bit at a time into the adder from their shift register storage locations.

One input to the adder is always the least significant bit position of the accumulator, AØ. In carrying out add or subtract operations, one of the numbers to be used is always stored in the accumulator. The other number to be added is generally supplied from the memory via the MDA signal line. When an add operation is selected, data is supplied to the other input of the adder through the gate in IC44.

When a subtract operation is specified, another gate in IC44 is enabled so that data is supplied to the adder via $\overline{\text{MDA}}$. This is the 1's complement version of the signal on the MDA line. The Model 832 adder produces subtraction by complementing and adding the subtrahend. The data on the $\overline{\text{MDA}}$ line is the 1's complement of the MDA signal. However, 2's complement addition is carried out in this computer. Recall that the 2's complement of a binary number is obtained by first finding the 1's complement and then adding a one to the least significant bit position. This is done here by presetting the carry flip-flop. Whenever a subtract operation is detected, the preset input on the carry flip-flop in IC43 is preset through the 4-input gate in IC41. The 470-ohm resistor connected between pin 13 and ground of IC41 is low enough in value to cause that input to appear as a binary 0 level. This keeps the output of the gate at a high level. With the preset input on the carry flip-flop high, it has no effect on the state of the flip-flop. However, when the subtract signal occurs and goes from a binary 0 to a binary 1, the 500 pf capacitor connected to pin 13 on IC41 couples a positive-going pulse that appears across the 470-ohm resistor. This capacitor charges quickly but during this time causes a positive pulse to appear at pin 13. If the TØ, CLK, and Execute lines are high at this time then the output of the gate will go low, thereby presetting the carry flip-flop. This applies a binary 1 to the carry input (C_i) on the adder. This will cause a binary 1 to be automatically added to the least significant bit position of the two words to be added, one in the accumulator and the other the one's complement of the number stored in the memory. The result produces 2's complement addition (subtraction).

During the decrement accumulator (DCA) instruction, the inputs to the adder are supplied by the accumulator and the $\overline{\text{DCA}}$ signal applied to the gate in IC45. To decrement the accumulator we subtract one from the contents of the accumulator. To subtract one we must add the 2's complement version of the binary number one to the contents of the accumulator. The 2's complement representation of the number one (0000 0001) is simply 1111 1111. Adding eight binary 1's to the contents of the accumulator produces the same effect as subtracting one from the contents of the accumulator. This string of binary 1's is generated by the $\overline{\text{DCA}}$ signal. It will remain low during the execution of the decrement accumulator instruction, thereby applying the appropriate bit pattern to pin 12 of the adder.

The circuitry for the auxiliary B register is also on the A1 circuit board. This register is constructed of two type 7495 4-bit shift registers. These are IC39 and IC40 on the board. Data can be shifted into this register from the accumulator. The AØ and BN terminals on the A1 circuit board are normally connected together so that when a rotate A and B register instruction (RAB) is executed, data from the accumulator is shifted out of the AØ bit position of the accumulator into the B register via the gates in IC44.

Clock pulses are applied to both IC39 and IC40 during an RAB instruction. The ROA signal from the C1 circuit board enables the clock at this time to cause the shift operation to occur. At all other times the ROA signal is low, inhibiting the clock and thereby causing the data contained within the B register to remain stored there. Notice also that as data is shifted into the B register, the data stored there is shifted out via the BØ terminal. The serial data from the BØ output line is routed through the gates in IC45 and appears at the BEØ terminal when an RAB instruction is executed. This causes the data that is stored in the B register to be transferred into the accumulator and the data that is stored in the accumulator is transferred into the B register.

Recall from our discussion earlier of the instructions in the Model 832 digital computer that the AND instruction causes a logical AND operation to take place between the data stored in the accumulator register and the data stored in the B register. Both the accumulator and the B register are initially loaded, the accumulator with an LDA and the B register with an RAB. When the AND instruction is executed, the contents of the A and B registers are shifted into an AND gate and the logical output is shifted back into the accumulator. During this time, the data stored in the B register is simply recirculated. The BØ output is fed through the appropriate gates in IC44 back to the input. Notice that this gate must be enabled by the AND signal on pin 4, indicating that an AND instruction is being executed. The ROA signal is high at this time, indicating that an AND operation is being performed so clock pulses shift the contents of the B register.

The ASD signal applied to the clear input of the carry flip-flop is used to keep the carry flip-flop in the reset state except during add, subtract, and decrement instructions.

The overflow flip-flop monitors the carry output signal from the adder. The J input of this flip-flop looks at the carry output signal. The K input to this flip-flop is (OVFC), the overflow control line. Initially the flip-flop is reset. Should a carry output occur during the T7 time interval, it will indicate a carry out of the most significant bit position of the accumulator. This is considered as an overflow. During T7 both the J and K inputs will be high if a carry appears. Therefore, the flip-flop will be toggled by the clock signal. Since it is initially reset it will become set, thus indicating an overflow condition. The OVFL line will be high and, therefore, will turn on the lamp driver, indicating an overflow. The overflow output signal has no significance in the control over the computer. It is simply made available for monitoring should the operator desire to monitor any overflow state that may occur during the operation of the computer. The OVFL signal could be used as the condition to the skip instruction which was explained earlier.

The auxiliary register multiplexer made up of one-half of the circuitry in IC45 is used to supply either the BØ or EØ signals to the BEØ output line. When the RAB instruction is being executed, the contents of the B register are shifted out via the BØ line and appear at the BEØ output. If the RAE instruction is executed, the RAE signal will enable the appropriate gate in IC45, thus causing the contents of the E register to be shifted through the gates in IC45 to the BEØ terminal.

THE A2 CIRCUIT BOARD

The A2 circuit board contains the accumulator register and all the associated logic. Refer to the A2 logic diagram.

The accumulator register itself is the 24-pin IC, IC54. This register can shift right or shift left, be parallel loaded or cleared. It has eight parallel output lines labeled AØ through A7. The register can be loaded in parallel via the eight input lines labeled AIØ through AI7. These lines are connected to the eight accumulator switch register switches on the front panel and the C3 circuit board. Whenever the MLA push button is depressed, the contents of the switch register will be loaded into the accumulator.

Depressing the load accumulator button generates a signal designated MLA. This forces the outputs of the two gates in IC53 high, thereby making pins 1 and 23 of the 74198 accumulator high. These two signals designate the mode of the accumulator. which in this case is parallel load. At the same time with the MLA line going high it forces the output of the inverter in IC52 low. This causes a pulse to be transferred through the .1 mfd capacitor and to appear across the 470-ohm resistor connected to the input of the 15844 IC in IC56. This causes the output of this gate to go low. This gate is tied to the clock input terminal of the accumulator and, therefore, initiates the parallel load operation.

The eight parallel output lines of the accumulator, AØ through A7, are connected directly to the LØ through L7 inputs on the right-hand lamp driver board. This permits the contents of the accumulator to be continuously monitored. The eight output lines are also monitored by the gates in IC55. These gates plus two gates in IC53 form an 8-bit AND gate that detects the zero or reset state of the accumulator. If all of the flip-flops in the accumulator are reset and in the zero state, then the ACZ output line is a binary 1. The ACZ signal indicates a zero accumulator. This is used in the jump-on-zero instruction. The accumulator may be cleared to zero by initially depressing the reset push button on the front panel. This applies a high signal to the CLR input on the accumulator, thereby resetting it.

Data may be shifted into the accumulator serially from several different sources. The selection of the source is determined by the accumulator multiplexer made up of IC50 and IC47. By referring to the A2 logic diagram you can see that the data can be shifted into the accumulator from the memory (MDA) during a load accumulator (LDA) instruction.

In the basic Model 832 that does not contain the optional semiconductor memory, store operations out of the accumulator take place with the E register. Data is shifted out of the accumulator via AØ and into the E register. At the same time we do not wish to destroy the contents of the accumulator so they are recirculated. This recirculation takes place via the second gate in IC50. When a store accumulator instruction is executed, the STA line is high, thereby permitting the data shifted out of AØ to be entered back into the shift register. This circuitry is disabled when the optional semiconductor memory is installed.

The third gate in IC50 is enabled by the ASD signal from the C1 circuit board. This signal is present during an add, subtract, or a decrement instruction. During these instructions the sum produced by the adder is shifted into the accumulator.

During a complement accumulator instruction the IVA line will be high, enabling the fourth gate in IC50. This permits the complement of the accumulator to be fed back into the accumulator. The A∅ output is inverted by an inverter in IC42 on the A1 circuit board. The $\overline{\text{A∅}}$ output signal is applied to the fourth gate in IC50.

The 7460, IC47, is an expander input for the 7454, IC50. The two gates in this expander are used for the AND operation and register transfer operations.

During an AND instruction, the contents of the B register and the accumulator are ANDed together and the logical result is placed back in the accumulator. This is done with the upper gate in IC47. Notice that both the A∅ and B∅ lines are applied to this AND gate along with the AND signal which is present when the AND instruction is being executed. The logical output result is passed through this gate and the OR gate in IC50 and through the inverter in IC52 to the input of the accumulator.

The lower gate in IC47 is used to cause data to be shifted into the accumulator from either the B or E registers. The gate is enabled by the signal RBE which is generated by the left-hand gate in IC48. This gate produces a binary 1 output when either the $\overline{\text{RAB}}$ or $\overline{\text{RAE}}$ signals are low, indicating that one of these instructions is being executed. With either one of those signals low, the gate is enabled and data from either the B or E registers, as determined by the auxiliary register multiplexer on the A1 circuit board, will be shifted into the accumulator via the BE∅ line.

There are three ways that the clock input to the accumulator (ACLK) can be initiated. We have already discussed one of these ways. A single clock pulse is generated when the MLA input line goes low during a manual load of the accumulator operation. The other two clock pulse sources are supplied by IC51. For most operations the syst^m's clock signal is applied through this left-hand gate during the execute phase. Notice that the CLK and the EXEC signals are applied to the left-hand gate in IC52. There are two other signals also applied to this gate. One of these is the $\overline{\text{SHA}}$ which indicates that a shift operation is ***not*** designated. The other is a signal JMZSH that is produced as a result of any jump operation, a store operation, or a halt. During a jump, store, or halt instruction we do not wish to cause the contents of the accumulator to be disturbed. Therefore, when any jump, store, or halt instruction is executed, the JMZSH signal is low, thereby inhibiting the left-hand gate in IC52 and preventing clock pulses from reaching the accumulator.

During a shift operation (SHA) the SHA line is high, thereby causing $\overline{\text{SHA}}$ to go low and inhibit the left-hand gate in IC51. However, the right-hand gate in IC51 is enabled. During the execute phase and the T2 time interval for the timing distributor when a clock pulse occurs it will cause the accumulator to shift one bit position to the left or to the right, depending upon the state of the IR4 flip-flop in the instruction register. Recall that a binary 0 in the IR4 position causes a shift right operation and a binary 1 causes a shift left.

The IR4 signal is supplied to pins 9 and 10 of the gate in IC49. This gate is enabled during the execute phase and by the SHA signal which indicates that a shift operation is to be performed. The output of the gate, IC49, controls the gates in IC53 that set the mode of the accumulator. Normally the accumulator

will rest in the shift right mode since most operations require a shift right operation. During this time the output of IC49 will be high. This will cause the output of the right-hand gate in IC53 to be low and the output of the left-hand gate in IC53 to be high. This indicates a shift right operation. If a shift instruction occurs then SHA will be a binary 1. With IR4 low to indicate a shift right operation, the output of the gate in IC49 will be high, as usual, causing a shift right condition to occur. If IR4 goes high during an SHA instruction, the output of the gate in IC49 will be low. This will reverse the states of the gates in IC53 which feed the accumulator and will, therefore, initiate a shift left operation. During either a shift left or a shift right operation, the shift occurs only one bit position. Refer to the mode states for the 74198 accumulator in the Appendix.

The other gate in IC49 is used to detect the halt instruction and to generate a pause (PAUS) signal that will cause the computer to stop. This gate in IC49 monitors the HLT line as well as the T7, EXEC, and clock lines. If a halt instruction occurs, the $\overline{\text{HLT}}$ input line to the A2 circuit board goes low, forcing the output to the inverter in the IC48 high and thereby enabling the gate in IC49. During the execute phase and the T7 time period when a clock pulse occurs the PAUS signal will be generated. This will cause the run/stop flip-flop on the C3 circuit board to reset and stop the clock.

THE C3 CIRCUIT BOARD

The C3 circuit board contains the main systems clock and the related control circuitry. The main clock is a complementary multivibrator made up of transistors Q_2 and Q_3. This multivibrator generates a train of rectangular pulses that are shaped and enabled and passed to the CLK and $\overline{\text{CLK}}$ outputs on C3.

The frequency of the clock operations depends upon the values of R_1, R_2, R_3 and capacitors C_1 and C_2. Resistors R_2 and R_3 form a voltage divider that sets the voltage at the base of Q_3 to +2.5 volts. When power is applied, C_1 initially acts as a short circuit before it begins to charge. With the emitter of Q_3 near ground and its base at +2.5 volts, its emitter base junction is reverse biased and the transistor is cut off. Q_2 is also cut off at this time, therefore, base drive is supplied to Q_1 through R_2. Q_1 will be saturated and its output will be low.

C_1 begins to charge and as its voltage reaches +2.5 volts added to the emitter-base voltage drop of Q_3, the emitter base junction of Q_3 becomes forward biased. Q_3 conducts and causes current to flow in Q_2. Q_2 will saturate and will, therefore, remove the base drive from Q_1. The output of Q_1 will go high. When Q_2 conducts, capacitor C_1 will discharge quickly through it and Q_3. This will cause the emitter base junction of Q_3 to again be reverse biased, thereby cutting off Q_2 and turning on Q_1. The output pulse generated at the collector of Q_1 is determined by the discharge time of C_1. C_1 again begins to charge and the cycle repeats over and over again. The normal clock frequency produced by this circuit is approximately 250 kHz. If a lower frequency is desired as in this computer, the clock slow/fast switch is thrown, thereby connecting the 100 mfd capacitor C_2 in parallel with C_1. This causes the charge time to be much longer, thereby generating a lower frequency clock. The clock pulses in this with the larger capacitor in the circuit is approximately two pulses per second.

The clock pulses at the output of Q_1 are applied through the lower gate in IC62 at pin 13. If this gate is enabled by the output from the latch in IC60 the clock pulses will pass through the gates in IC62 and the buffer gates in IC63 to the CLK and $\overline{\text{CLK}}$ output terminals. IC63 is a high power driver that has a large fanout and provides buffering for the CLK and $\overline{\text{CLK}}$ signals that are applied to all points of the computer.

The gates in IC60 are connected to form two latches that are used to buffer the contact bounce from the Step push button and the Auto/Step selection switch. With the Auto/Step switch in the Auto position, the gate to which clock pulses are applied is enabled, thereby permitting the computer to operate automatically from the clock. If the Step position is selected, the output of the latch applied to pin 2 of IC62 will be high, thereby enabling the latch connected to the Step push button. This will cause a logic level transition to occur each time the Step push button is depressed, thereby generating a clock signal.

Two of the gates in IC61 are connected as a latch to buffer the contact bounce of the load accumulator push button. The output of this latch is an MLA signal which causes the accumulator to be loaded in parallel from the front panel switch register. The other two gates in IC61 are not used but are available at terminals on the board for modifications if desired.

The operation of the clock is determined by the state of transistor Q_4. If this transistor is cut off, then the clock will run and generate clock pulses. However, if this transistor conducts, it saturates and shorts out capacitors C_1 and C_2, thereby preventing the clock from running. The state of transistor Q_4 is determined by the run/stop flip-flop which is one of two JK flip-flops in IC59.

This flip-flop is initially reset when the Reset push button is depressed. Depressing the Reset push button causes the state of the contact bounce latch in IC57 to change. The output of this latch generates the CLR and $\overline{\text{CLR}}$ signals which are used to reset the various registers in the computer. The output of the latch is also used to reset the run/stop flip-flop in IC59 through the negative OR gate made up of gates in IC57. With the stop output in the high state, the flip-flop is reset and thereby causes Q_4 to conduct and the clock to be inhibited.

When the Start push button is depressed, its contact bounce is buffered by a second JK flip-flop in IC59 connected as a latch. The output of this latch causes the run/stop flip-flop to be toggled and as a result it becomes set. The Run output line will go high. This will cause the run lamp on the left-hand display to turn on. The Stop output from the flip-flop will go low and, therefore, Q_4 will be cut off. This permits the clock to run.

The run/stop flip-flop is also controlled by the $\overline{\text{PAUS}}$ signal. This is a signal generated when a halt instruction is decoded. However, when the $\overline{\text{PAUS}}$ signal goes low, the run/stop flip-flop is reset through the negative OR gate in IC57 and the clock stops.

The other circuitry on the C_3 circuit board consists of the eight accumulator switch register switches. Each of these is connected in series with a 1k-ohm resistor between +5 volts and ground. When the switches are open, the +5 volts is applied to the AI∅ through AI7 terminals through the 1k-ohm resistors. If the switches are closed, the AI∅ through AI7 terminals are low, representing a binary 0.

The lamp display select switch is used to generate the LS signal that controls the display multiplexer on the C_2 circuit board. In one position a binary 1 is generated and in the other a binary 0 is generated to display either the instruction register or the program counter.

CIRCUIT MODIFICATIONS

The Model 832 was designed to use the full memory complement of 16 programmable read-only locations and the 16 read/write locations in the optional semiconductor memory. In the basic machine, the 16 words of semiconductor memory are not available. However, read/write operations from memory are demonstrated with a single memory location made up of an 8-bit shift register, IC11, on the M2 board. In order to permit this single memory location to operate properly, some additional logic circuitry is used. This circuitry is implemented with the various free gates and inverters available on the circuit boards. When the optional semiconductor memory is installed, most of this circuitry is removed.

Refer to Fig. 9. The JMZS signal to the A2 shown in Fig. 9 controls the clock to the accumulator. Normally, the JMZS signal from the C1 board will connect directly to the JMZS terminal on the A2 board. The logic circuit shown here permits the STA instruction to control the single memory location clocking along with that of the accumulator. The SKC instruction also modifies the signal to the JMZS input to the A2 board.

In Fig. 9 a free gate on the C3 board controls the clock pulses to the single shift register memory location. It permits clock pulses to step the register only in the EXEC phase.

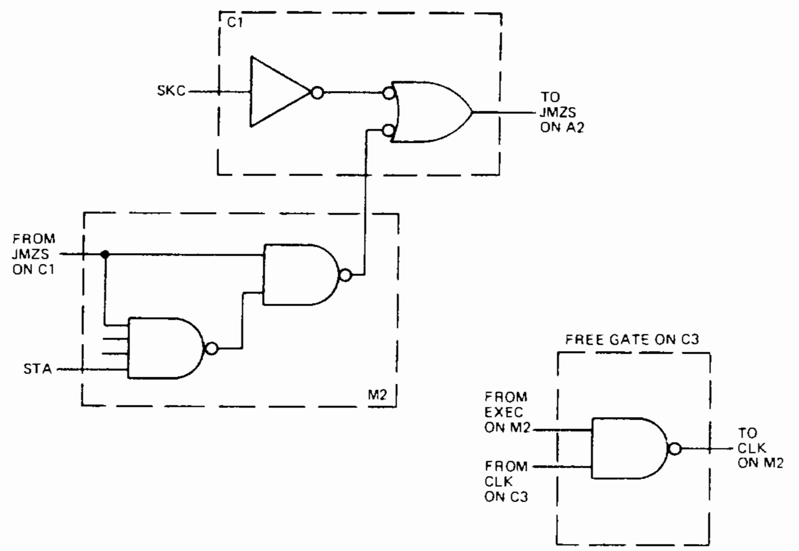
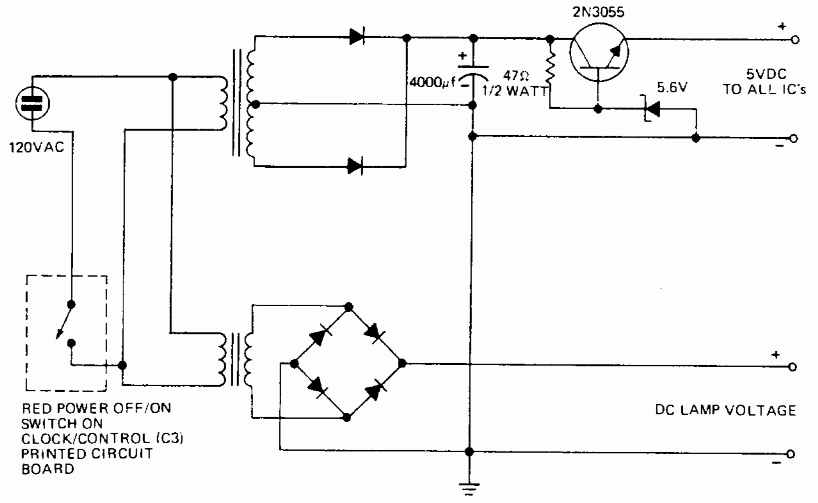


Fig. 9. Circuit modifications to permit proper operation of the single read/write memory location in the basic version of the Model 832.



2N3055
+
5VDC
TO ALL IC's
4000μf
47Ω
1/2 WATT
5.6V
120VAC
RED POWER OFF/ON SWITCH ON CLOCK/CONTROL (C3) PRINTED CIRCUIT BOARD
DC LAMP VOLTAGE

Fig. 10. Power supplies in the Model 832 digital computer.

POWER SUPPLIES

The Model 832 contains two power supplies, one for the integrated circuits and the other for the display lamps. Both are shown in Fig. 10.

The IC Supply. The power supply for the integrated circuits furnishes a stabilized 5 volts to all ten printed circuit boards in the computer. It consists of a power transformer with a center-tapped secondary used with two diodes in a standard full wave rectifier circuit. The pulsating dc generated by this supply is filtered by a 4000 mfd filter capacitor. The unstabilized output voltage is applied to an emitter follower stabilizer. It consists of a 2N3055 series pass transistor whose base voltage is set by the 5.6 volt Zener diode biased by the 47-ohm resistor. The output voltage is 5 volts. The computer circuitry draws approximately 1 ampere of current from this supply.

The Lamp Supply. A separate power supply is used to furnish current to the display lamps. The lamps require nearly 1 ampere of current when they are all on. The voltage to operate them does not have to be regulated.

The supply consists of a power transformer and a full wave diode bridge rectifier. The supply is not filtered. The pulsating dc output of the supply is applied directly to the lamps.

Appendix

Decimal/Hexadecimal Conversion Chart

The chart on the next page will help you to readily convert hexadecimal values into decimal values and vice versa. The table is valid for eight bits of data where numbers between 0 and 255 can be represented. The numbers in this range can be represented by a two-digit hexadecimal number between the values 00 and FF.

To Convert a Two-Digit Hex Number Into Its Decimal Equivalent. The most significant hex digit is given in the far left-hand column of the table. Next to it is a blank representing the least significant digit. The sixteen possible least significant digits appear in a row across the top of the table. As an example, let's convert the hex number 7C into its decimal equivalent. First, locate 7 in the vertical most significant digit column. Then locate the C in the horizontal row. Follow the 7 row to the right and the C column downward until they intersect. The number at their intersection is the decimal equivalent, in this case the number 124.

To Convert a Decimal Number Into Its Hex Equivalent. Find the desired number between 0 and 255 in the chart. Suppose this is 212. Now, trace directly to the left and note the most significant digit in the left hand column, which is D. Then trace vertically from 212 to the top row to find the least significant digit 4. The decimal number 212 in hex is D4.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0__	0000	0001	0002	0003	0004	0005	0006	0007	0008	0009	0010	0011	0012	0013	0014	0015
1__	0016	0017	0018	0019	0020	0021	0022	0023	0024	0025	0026	0027	0028	0029	0030	0031
2__	0032	0033	0034	0035	0036	0037	0038	0039	0040	0041	0042	0043	0044	0045	0046	0047
3__	0048	0049	0050	0051	0052	0053	0054	0055	0056	0057	0058	0059	0060	0061	0062	0063
4__	0064	0065	0066	0067	0068	0069	0070	0071	0072	0073	0074	0075	0076	0077	0078	0079
5__	0080	0081	0082	0083	0084	0085	0086	0087	0088	0089	0090	0091	0092	0093	0094	0095
6__	0096	0097	0098	0099	0100	0101	0102	0103	0104	0105	0106	0107	0108	0109	0110	0111
7__	0112	0113	0114	0115	0116	0117	0118	0119	0120	0121	0122	0123	0124	0125	0126	0127
8__	0128	0129	0130	0131	0132	0133	0134	0135	0136	0137	0138	0139	0140	0141	0142	0143
9__	0144	0145	0146	0147	0148	0149	0150	0151	0152	0153	0154	0155	0156	0157	0158	0159
A__	0160	0161	0162	0163	0164	0165	0166	0167	0168	0169	0170	0171	0172	0173	0174	0175
B__	0176	0177	0178	0179	0180	0181	0182	0183	0184	0185	0186	0187	0188	0189	0190	0191
C__	0192	0193	0194	0195	0196	0197	0198	0199	0200	0201	0202	0203	0204	0205	0206	0207
D__	0208	0209	0210	0211	0212	0213	0214	0215	0216	0217	0218	0219	0220	0221	0222	0223
E__	0224	0225	0226	0227	0228	0229	0230	0231	0232	0233	0234	0235	0236	0237	0238	0239
F__	0240	0241	0242	0243	0244	0245	0246	0247	0248	0249	0250	0251	0252	0253	0254	0255

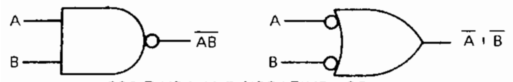
Decimal-Hexadecimal-Binary Table

Decimal	Hexadecimal	Binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

Powers of Two Table

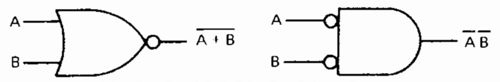
2^n	n	2^{-n}
1	0	1.0
2	1	0.5
4	2	0.25
8	3	0.125
16	4	0.062 5
32	5	0.031 25
64	6	0.015 625
128	7	0.007 812 5
256	8	0.003 906 25
512	9	0.001 953 125
1 024	10	0.000 976 562 5
2 048	11	0.000 488 281 25
4 096	12	0.000 244 140 625
8 192	13	0.000 122 070 312 5
16 384	14	0.000 061 035 156 25
32 768	15	0.000 030 517 578 125

Logic Symbols



POSITIVE NAND/NEGATIVE NOR

$\overline{AB} = \overline{A} + \overline{B}$



POSITIVE NOR/NEGATIVE NAND

$\overline{A + B} = \overline{A}\,\overline{B}$

Subroutines

MULTIPLICATION

The following subroutine multiplies the number in location N + 11 by the number in location N + 12. The product will appear in the accumulator. The multiplication is by successive additions. The maximum permissible product size is 255 as limited by the size of the accumulator. The sizes of the multiplier and multiplicand should be limited accordingly.

Location	Instruction/Address
N	LDA N + 11
N + 1	RAB
N + 2	LDA N + 12
N + 3	RAB
N + 4	DCA
N + 5	JOZ N + 9
N + 6	RAB
N + 7	ADD N + 12
N + 8	JMP N + 3
N + 9	RAB
N + 10	HLT
N + 11	Multiplier
N + 12	Multiplicand

DIVISION

The following subroutine divides the number in location N + 14 by the contents of location N + 15. The quotient will appear in the accumulator, and the remainder will appear in location N + 16. Both dividend and divisor must be positive and less than 255. The division process is by successive subtractions.

Location	Instruction/Address
N	RAB
N + 1	LDA N + 14
N + 2	STA N + 16
N + 3	LDA N + 16
N + 4	SUB N + 15
N + 5	JOM N + 11
N + 6	STA N + 16
N + 7	RAB
N + 8	ADD N + 13
N + 9	RAB
N + 10	JMP N + 3
N + 11	RAB
N + 12	HLT
N + 13	1
N + 14	Dividend
N + 15	Divisor
N + 16	Remainder

Signal Mnemonics, Definitions, and Boolean Equations

SIGNAL	DEFINITION
A, $\overline{\text{A}}$	Output LSB (2°) bit flip-flop of timing-distributor counter.
AØ – A7	Eight outputs of accumulator register flip-flops. AØ = LSB, A7 = MSB or sign.
AØN	Serial input to E register from accumulator.
ACZ	Accumulator equal to zero. ACZ = $\overline{\text{AØ}}\cdot\overline{\text{A1}}\cdot\overline{\text{A2}}\cdot\overline{\text{A3}}\cdot\overline{\text{A4}}\cdot\overline{\text{A5}}\cdot\overline{\text{A6}}\cdot\overline{\text{A7}}$
ADD	Add to accumulator instruction decoded. ADD = $\overline{\text{IR7}}\cdot\text{IR6}\cdot\overline{\text{IR5}}$
ADRØ – ADR4	Five output bits of address multiplexer and inputs to memory address register.
AIØ – AI7	Eight parallel inputs to accumulator register.
AND	Operate function code decoder output indicating that a logical AND operation is to be performed. AND = $\overline{\text{IR2}}\cdot\text{IR1}\cdot\overline{\text{IRØ}}$
ASD	Add or subtract or decrement instruction decoded. Used to initially reset the carry flip-flop prior to an arithmetic operation. ASD = ADD + SUB + DCA.
B, $\overline{\text{B}}$	Output of 2^1 bit flip-flop of the timing distributor counter.
BØ – B7	Eight outputs of auxiliary (B) register flip-flops. BØ = LSB; B7 = MSB.
BN	Serial data input line to B register.
BEØ	Serial data output of either the B or E registers. BEØ = BØ·RAB + EØ·RAE
BRH	Branch. Jump instruction decoded. BRH = JMP + JOM·A7 + JOZ·ACZ
C, $\overline{\text{C}}$	Outputs of 2^2 bit flip-flop of timing distributor counter.
CDX	Skip instruction condition signal used to determine if skip occurs.
C_i	Carry input to adder.
CLK, $\overline{\text{CLK}}$	Clock signal from either clock oscillator or front panel Step push button.
CLR, $\overline{\text{CLR}}$	Signal generated by depressing front panel Reset push button.
CMA	Complement accumulator operate function decoded. CMA = $\text{IR2}\cdot\overline{\text{IR1}}\cdot\overline{\text{IRØ}}$
C_o	Carry output of adder
D, $\overline{\text{D}}$	Outputs of MSB (2^3) flip-flop of timing distributor counter. Fetch ($\overline{\text{D}}$) – Execute (D) flip-flop.
DCA	Decrement accumulator operate function decoded. DCA = $\overline{\text{IR2}}\cdot\text{IR1}\cdot\text{IRØ}$
DDM	Enable signal for timing distributor decoder.
EØ – E7	Eight outputs of the E register flip-flops. EØ = LSB, E7 = MSB
ECLR	Reset signal for E register. Binary 1 to reset.

ESE	Enable serial input to E register.
EXEC	Execute output of fetch/execute flip-flop
FETCH (FTCH)	Fetch output of fetch/execute flip-flop.
G1, G2, G3, ...Gn	Ground reference.
HLT	Halt operate function decoded. HLT = IR2·IR1·IRØ
IRØ – IR7	Outputs of eight instruction register flip-flops.
IVA	Invert accumulator operate function decoded. Same as CMA.
JA, JB, JC, JD	J input to A, B, C, and D JK flip-flops in timing distributor register.
JMP	Unconditional jump instruction decoded. $\text{JMP} = \text{IR7} \cdot \overline{\text{IR6}} \cdot \overline{\text{IR5}}$
JOM	Jump on accumulator minus instruction decoded. $\text{JOM} = \text{IR7} \cdot \overline{\text{IR6}} \cdot \text{IR7}$
JOZ	Jump when accumulator zero instruction decoded. $\text{JOZ} = \text{IR7} \cdot \text{IR6} \cdot \overline{\text{IR7}}$
JMZS	Any jump or STA instruction decoded. $\text{JMZS} = \overline{\text{JMP}} + \overline{\text{JOZ}} + \overline{\text{JOM}} + \overline{\text{STA}}$
JMZSH	Any jump, STA, or HLT instruction decoded. $\text{JMZSH} = \overline{\text{JMZS}} + \overline{\text{HLT}}$
KA, KB, KC, KD	K inputs to A, B, C and D JK flip-flops in timing distributor register.
LØ – L1	Eight inputs to lamp driver boards on C2 board, eight outputs of display multiplexer.

LAA	Normally open terminal of Load push button switch.
LAB	Normally closed terminal of Load push button switch.
LAC	Arm (ground) of Load push button switch.
LDA	Load accumulator instruction decoded. $\text{LDA} = \overline{\text{IR7}} \cdot \overline{\text{IR6}} \cdot \overline{\text{IR5}}$
LS	Lamp select control for display multiplexer.
M2	OR input to MDA output gate from SDZ or SWM.
MAØ – MA4	Outputs of memory address register flip-flops. MAØ = LSB; MA4 = MSB.

MDA	Serial memory data output.
	MDA = SD1 + SWM or
	MDA = SD1 + SD2

MLA	Manual load of accumulator, generated when Load push button on front panel is depressed.
OPR	Operate instruction decoded. OPR = IR7·IR6·IR5
OVFC	Overflow control on overflow flip-flop.
OVFL	Overflow. Output of overflow flip-flop.
PAUS	Pause signal used to stop or halt computer by resetting the run/stop flip-flop. PAUS = HLT·EXEC·T7·CLK
PCØ – PC4	Five output bits of program counter. PCØ = LSB; PC4 = MSB.
RAB	Rotate contents of A and B registers operate function decoded. $\text{RAB} = \overline{\text{IR2}} \cdot \overline{\text{IR1}} \cdot \overline{\text{IRØ}}$

RAE	Rotate contents of A and E registers operate function decoded.
	$RAE = IR2 \cdot IR1 \cdot \overline{IR\emptyset}$
ROA	Rotate or AND operate function decoded.
	ROA = RAB + RAE + AND
RSTA	Normally open contact of Reset push button switch.
RSTB	Normally closed contact of Reset push button switch.
RSTC	Arm (ground) of Reset push button switch.
SD1	Serial data one, serial data output from programmable read-only memory multiplexer.
SD2	Serial data two, serial data output from optional semiconductor read/write memory via the multiplexer.
SHA	Shift accumulator operate instruction decoded.
	$SHA = \overline{IR2} \cdot \overline{IR1} \cdot IR\emptyset$
SHLN	Serial shift left input to accumulator.
SKC	Skip on condition operate function decoded.
	$SKC = IR2 \cdot \overline{IR1} \cdot IR\emptyset$
SME	Serial memory (E register) control. Load or recirculate.
SSA	Normally open contact of Step push button switch.
SSB	Normally closed contact of Step push button switch.
SSC	Arm (ground) of Step push button switch.

STA	Store accumulator instruction decoded.
	$STA = \overline{IR7} \cdot \overline{IR6} \cdot IR5$
STE	Store enable to optional semiconductor memory.
SUM	Serial output of one bit adder.
STOP	Complement output of run/stop flip-flop used to control the clock oscillator.
STTA	Normally open contact of Start push button switch.
STTB	Normally closed contact of Start push button switch.
STTC	Arm (ground) of Start push button switch.
SUB	Subtract instruction decoded.
	$SUB = \overline{IR7} \cdot IR6 \cdot IR5$
SWM	Serial write memory (E Register) data output.
	$SWM = E\emptyset \cdot MA4$
TØ, T1, T2, T3, T7	Timing output signals from timing distributor decoder.
WØF, W1E, W2D, W3C, W4B, W5A, W69, W78, W87, W96, WA5, WB4, WC3, WD2, WE1, WFØ	Programmable read-only memory word address enable lines from M register decoder.

THE MAN WHO COUNTS

"The man who counts is the man who is decent and makes himself felt as a force for decency, for cleanliness, for civic righteousness. First, he must be honest. In the next place, he must have courage. The timid man counts but little in the rough business of trying to do well the world's work. In addition, he must have common sense. If he does not have it, no matter what other qualities he may have, he will find himself at the mercy of those who, without possessing his desire to do right, know only too well how to make the wrong effective."–Theodore Roosevelt

This statement of Theodore Roosevelt's more than half a century old, appeals to me as being a very sound piece of practical advice. Read it carefully. It can be of real value to you--*today*.

William A. Coleman