PROJECTS–HI-FI–COMPUTERS

$1.25 ■ JAN. 1979

Radio-Electronics

THE MAGAZINE FOR NEW IDEAS IN ELECTRONICS

600-MHz COUNTER

A battery-powered 600-MHz frequency counter that's small enough to fit in the palm of your hand. Build it for less than 17 cents per MHz. Turn to page 39.

VIDEO MOTORCYCLE GAME

Rev your engine, hang a wheelie, accelerate up the ramp and see how many obstacles you can jump. Construction starts on page 44.

DIGITAL CIRCUIT DESIGN

Part 2. How to design digital circuits from scratch. The walk through sequential and combinational circuits and circuit reduction techniques starts on page 47.

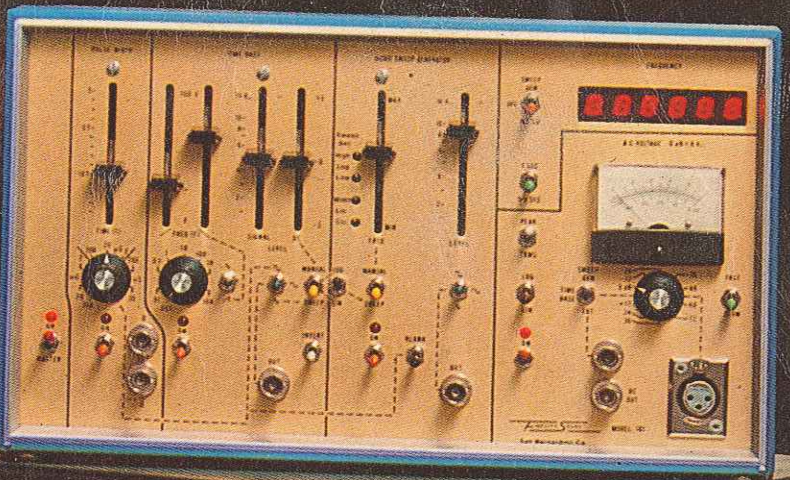
4-CHANNEL FM

A look at the different broadcast systems under consideration by the FCC and what it will mean to you. For the complete story, turn to page 51.

PLUS:

- ★ Build a NOM card for an 1802-based computer
- ★ Hobby Corner
- ★ Computer Corner
- ★ Jack Darr's Service Clinic
- ★ 2 Hi-Fi Lab Test Reports

A GERNSBACK PUBLICATION



COVER STORY
AUDIO TEST STATION

Professional-quality test instrument you can build combines several important instruments into a single cabinet. Construction starts on page 35.

BUILD THIS

NOM Card For The 1802

Part 2—Add-on math board for an 1802-based microcomputer. Based on a number-crunching IC, this board speeds execution time, reduces software overhead and saves memory

L. STEVEN CHEAIRS

LAST MONTH WE LOOKED AT HOW THE NOM card reduces computer memory requirements and increases processing speed by eliminating number-crunching software routines. This month, we present final construction details.

Construction

The components used in this project are all readily available; assembly is straightforward; and the circuit can be wire-wrapped or built on a PC card.

Use a double-sided glass epoxy circuit board with 2-ounce copper foil (available from Questar). A heavy plate layer covers all runs, and the holes are plated-through. The card has gold-plated fingers and a solder mask. For those who wish to etch the circuit board themselves, the foil patterns are shown in Figs. 8 and 9.

In assembling the board, pay special attention to component orientation. Figure 10 shows the correct placement and orientation of all the components and Fig. 11 shows the PC board pinout and switch placement. First, install and solder all resistors, capacitors and diodes. Connect the +5-volt and −15-volt leads, (the −4 volts is derived from the −15-volt source) and methodically test all power-supply pads for the proper voltages. If the power levels are OK, disconnect the power and install the IC's; if not, check for possible shorts or faulty components. No calibration is required.

Check-out and operation

Check-out and operation is theoretically very simple. First, enter the first number into the X-register. Follow this with the next number; all numbers enter the X-register. Execute a math operation, such as an ADD. Enter and execute an OUT instruction. If the correct answer is obtained, then 90% of the test is complete, and the only remaining functions to test are error and branch. If you did not receive the correct answer, check to see if all the DIP switches were closed; if so, then recheck component placement and orientation. If no mistake was made (and you have programmed the 1802 correctly), then a component failure has occurred. Use normal digital troubleshooting techniques to isolate and solve the problem.

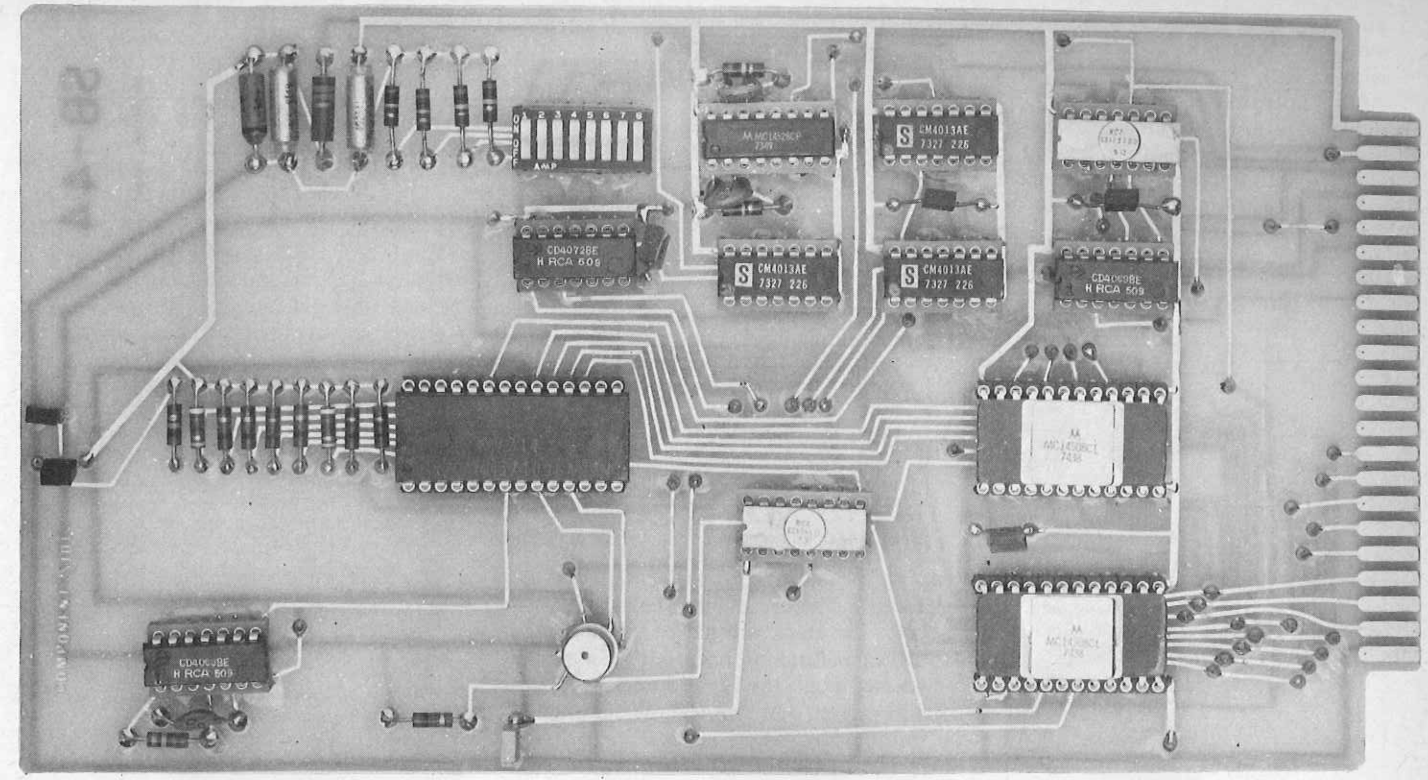
Now, enter a branch instruction to see if the branch outline interrupts the 1802. If this works, proceed to the final test. Enter a zero into the X-register, then execute the 1/X instruction. If an error occurs, you have completed the NOM interface; if not, check the error flip-flop.

The basic operation is outlined in the flowchart shown in Fig. 12. The user program first places the numbers in a FIFO table along with the required mathematical operations. Enter the first number into the X-register, then exchange the X- and Y-registers. Now, enter the next number into the X-register, then perform the desired operations. Enter the next number (if any) and perform the desired operations. Continue until all numbers and operations are completed. Execute an OUT instruction, store the digits into the user's FIFO table and return to the user program. The above description implies that the 1802 is tied up 100% of the time with the NOM, but actually a very small percentage of the 1802's time is spent with the NOM during these operations. The 1802 only moves data/instruction into (and out) of the NOM; most of the time it is used to perform mathematical calculations or to manipulate data inside the NOM. During this time the 1802 is free to perform other tasks.

Programming the NOM

The NOM has 70 instructions that can be classified into seven groups: Digit entry, move, math, clear, branch, input/output, and mode control. Table 3 lists the mnemonics for these instructions and the associated binary code. (A detailed description of what each instruction does; including the mnemonic, octal op-code and execution time; can be obtained by sending a SASE with 28¢ postage to NOM Instructions, **Radio-Electronics,** 200 Park Ave. South, New York, NY 10003.—Editor)

The first class of instruction—digit entry—has 17 members. The stack is



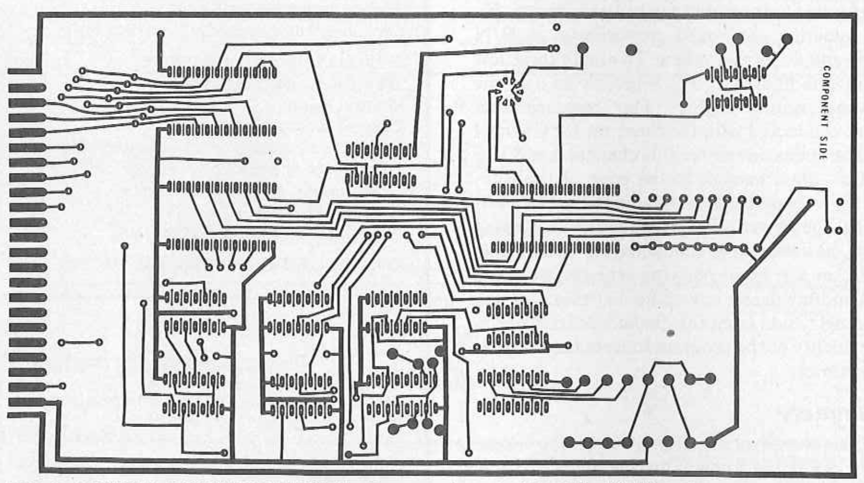


FIG. 8—PRINTED CIRCUIT PATTERN for the component-side of the NOM card.

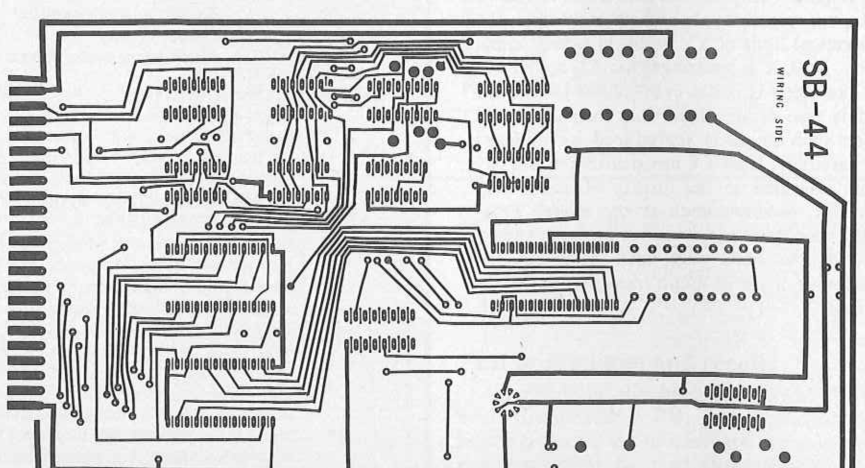


FIG. 9—PRINTED CIRCUIT PATTERN for the foil-side of the NOM card.

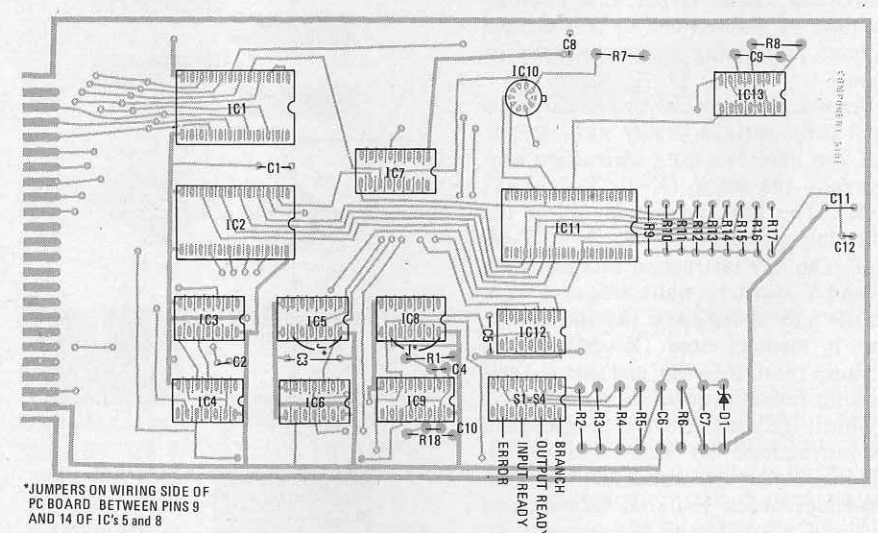


FIG. 10—COMPONENT PLACEMENT DIAGRAM shows where the parts go on the circuit board.

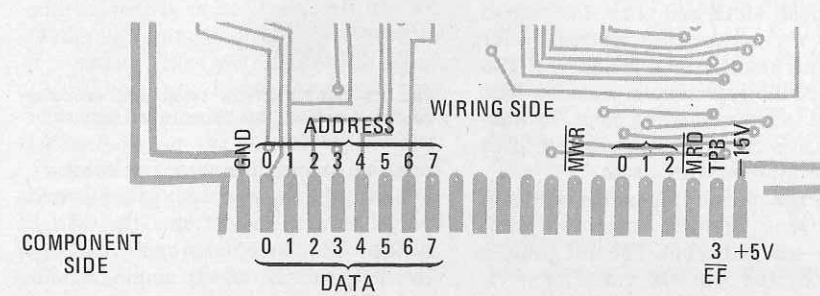


FIG. 11—PINOUT DIAGRAM shows where connections on the circuit board go.

pushed and the X-register is cleared when a digit, a decimal point, or a π is entered with an AIN, 0-9, DP, or PI instructions. After "initiation of number entry," the digit and future digits are entered into the X-mantissa. Any digits following the eighth mantissa digit are ignored, and any subsequent entry of digits or DP, EE, or CS

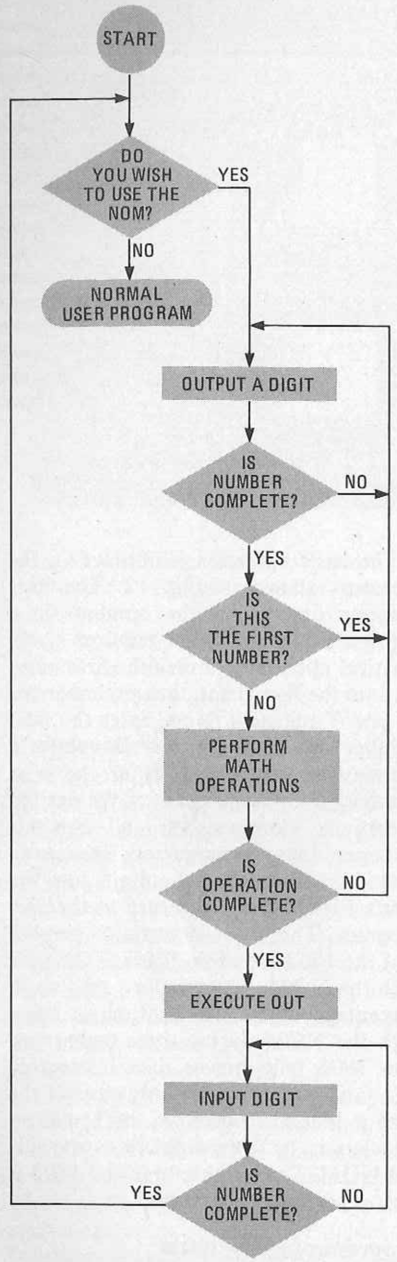


FIG. 12—TYPICAL OPERATION PROCEDURE for the 1802's interface program.

instructions do not cause initiation of number entry. Termination of the number-entry mode occurs when any instruction is executed (except 0–9, DP, EE, CS, PI, AIN, or HALT). When termination occurs, two things happen. First, the number is normalized by adjusting the exponent and decimal point position. The decimal point is placed to the right of the first digit. Second, the next digit, π, or decimal point entered causes initiation of number entry. As you might expect, there is an

TABLE 3—Instruction Summary Table

I_{4-1} I_6I_5 →	00	01	10	11
0000	0	TJC	INV	XEY
0001	1	TX=0	EN	EX
0010	2	TXLTO	TOGM	10X
0011	3	TXF	ROLL	SQ
0100	4	TERR	SIN(SIN^{-1})	SQRT
0101	5	JMP	COS(COS^{-1})	LN
0110	6	OUT	TAN(TAN^{-1})	LOG
0111	7	IN	SF1	1/X
1000	8	SMDC	PF1	YX
1001	9	IBNZ	SF2	+(M+)
1010	DP	DBNZ	PF2	−(M−)
1011	EE	XEM	ECLR	x(Mx)
1100	CS	MS	RTD	/(M/)
1101	PI	MR	DTR	PRW1
1110	AIN	LSH	POP	PRW2
1111	HALT	RSH	MCLR	NOP

(Note: All 00 instructions do not terminate number entry)

exception to the number-entry initiation rule. The stack will not be pushed if the ENTER instruction occurred prior to the entered digit; the X-register, however, is still cleared and the new digit is entered into the X-register.

The IN instruction is used to enter all digits of a number; it does not cause number-entry initiation. However, it does terminate this mode if the NOM is in that mode prior to the IN instruction being executed. Thus, 0–9, AIN and IN instructions can be mixed without performing an ENTER instruction before the IN instruction. The IN instruction always pushes the stack except if the previous instruction was an ENTER, thus allowing multiple IN instructions to be executed without performing an ENTER between them.

Second, the move instruction group has eight parts: ROLL, POP, XEY, XEM, MS, MR, LSH, and RSH. The ROLL instruction simply rolls the stack (X→T, T→Z, Z→Y, Y→X). The POP instruction causes the following sequence: Y→X, Z→Y, T→Z, O→T. The XEY instruction exchanges the X- and Y-registers; while XEM exchanges the X- and M-registers; the MS instruction is memory-store (X↔M); MR is memory-recall (M↔X); and LSH and RSH are shift mantissa instructions.

Third, the math group is composed of 24 instructions. As mentioned earlier, this instruction set is available from **Radio-Electronics** Editorial Offices upon receipt of a SASE with 28¢ postage.

Fourth, the clear group has only two instructions: MCLR and ECLR. Instruction MCLR is the master clear instruction for all internal registers and memory. It also initializes the I/O control signals—MDC = 8 and MODE = floating point. Instruction ECLR is the error flag clear; it loads the error flip-flop with a zero.

Fifth, the branch instruction has eight instructions divided into two subgroups—test and count. The test group is formed by the JMP, TJC, TERR, TX = 0, TXF, and TXLTO. The count subgroup

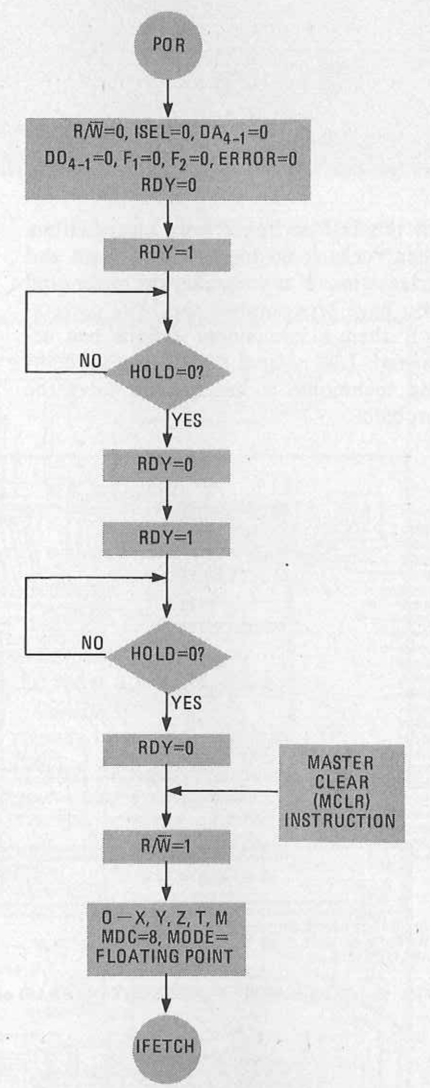


FIG. 13—INSTRUCTION fetch and execution flowchart defining the initialization routine.

contains instructions IBNZ and DBNZ.

Sixth, the input/output group is composed of nine instructions that can be divided into three subgroups: multidigit instructions (IN, OUT), single-digit instructions (AIN), and flags (SF1, PF1, SF2,

PRW1, PRW2). In the design described here the event flags were not used: therefore, the SF1, PF1, SF2 and PF2 instructions are of no use with this interface.

The last group of instructions is mode control: It contains TOGM, SMDC and INV.

On power-up the mantissa digit count is set at 8, and the mode is set to floating point. Figure 13 shows the initialization routine. After initialization (POR) or the completion of an instruction, the input-ready signal goes to a logic high, which tells the external hardware to supply a new instruction.

Sixteen of the 70 instructions are two words long (See Fig. 14.) These instructions are: INV+, INV−, INVX, INV/, INV SIN, INV COS, INV TAN, JMP, TJC, TERR, TX = 0, TXF, TXLTO, IN, OUT, and SMDC. The first word of a two-word instruction is the same format as the one-word instruction. The second word contains the branch address that is to be loaded into the PC (*Program Counter*) during the branch; or it contains the MDC (*Mantissa Digit Count*) for SMDC instructions; or it may have the high-order address bits for external RAM on the IN/OUT instructions. The low-order address is placed on the DA lines. This interface does not use an external RAM. Generally, the second word is ignored except for the SMDC instruction. The $I_{1\text{-}4}$ lines must contain digit data during the AIN and IN instructions. Each two-word instruction generates an input ready twice—one for each word. The first type are the inverse instructions. These require the INV instruction to be executed first (followed by the desired instruction). The second type is the SMDC instruction. The second word of this instruction is the mantissa digit count—a BCD number from 1 to 8. The other types of two-word instructions have adequately been discussed previously.

Since there are software differences between the many 1802 systems involved (primarily in the I/O and memory addresses), the software I used in my system is not included here. As an alternative, the flowcharts in Figs. 12 and 13 can serve as a guide in developing your own software. **R-E**

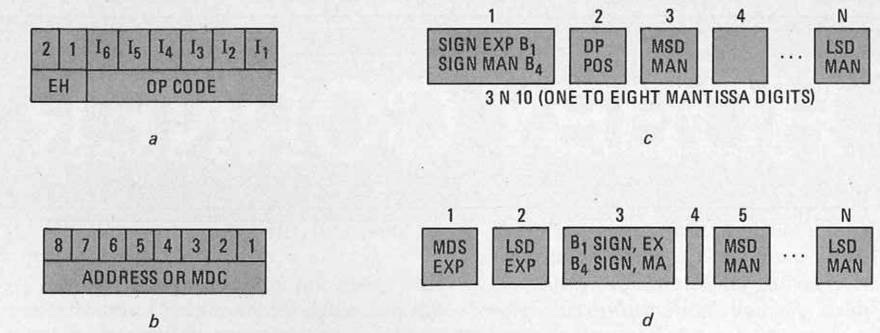


FIG. 14—FIRST WORD of 2-word instruction (*a*), second word (*b*). Floating-point notation formats and scientific-notation formats are illustrated at *c* and *d*, respectively.

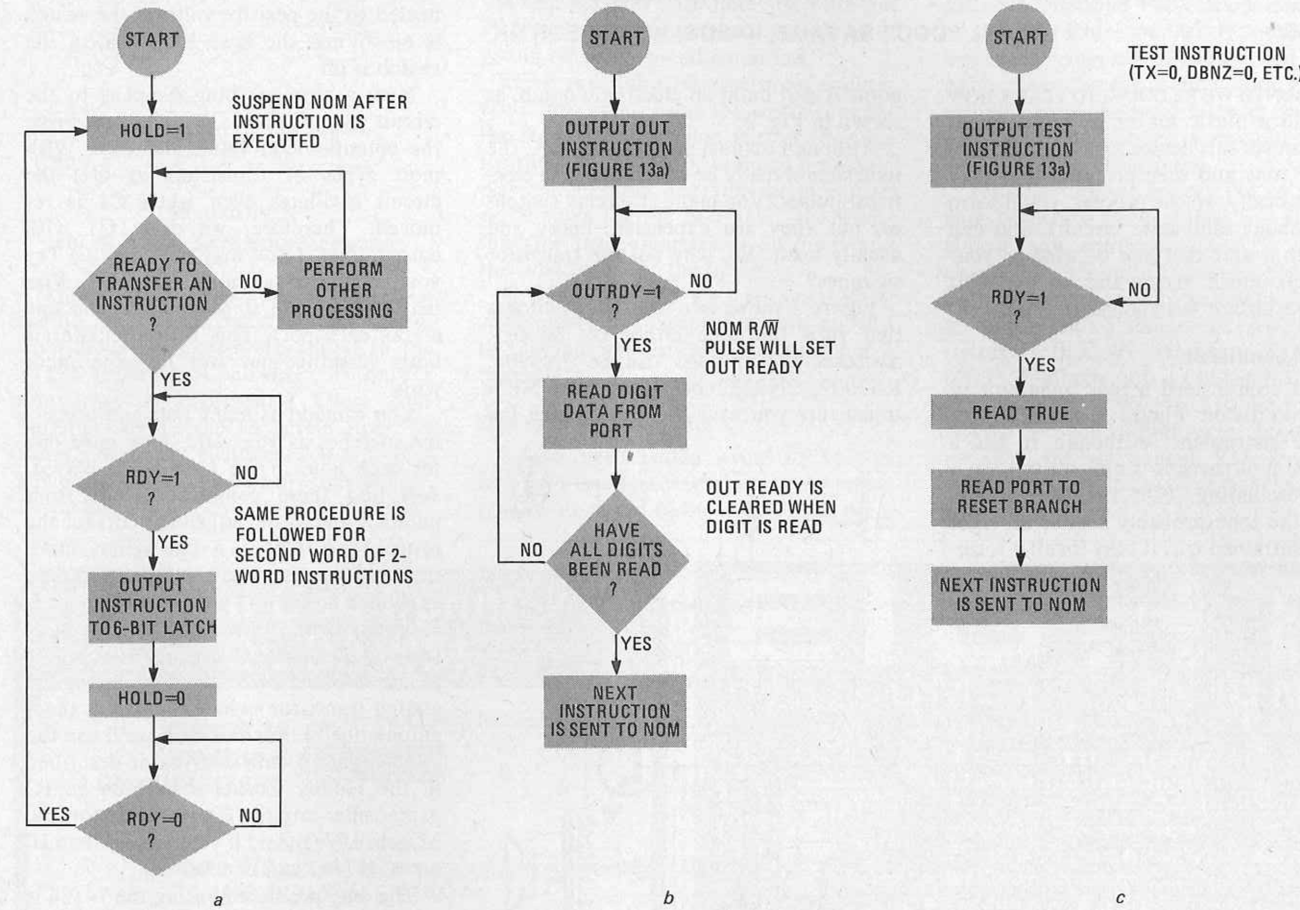


FIG. 15—SOFTWARE FLOWCHARTS. a) TRANSFER of data to NOM; b) TRANSFER of data from NOM; c) USE OF branch-type instructions.

"While I was waiting, I adjusted your roof antenna, aligned your set and cleaned the tuner."