


10

BUILD AN INTELLIGENT SERIAL EPROM PROGRAMMER

*Steve's new and improved device includes
on-board CPU and intelligent firmware*



I don't like admitting that I made a mistake, but apparently I did. Well, not actually. You see, I was dragged into . . . Let me start from the beginning.

My February 1985 Circuit Cellar article was a project on how to build a serial EPROM programmer, about which I said: "The latest Circuit Cellar EPROM programmer is a serial-port programmer that has the speed of a turtle, the intelligence of the mightiest computer (that is, it has absolutely no smarts of its own), and is as functional as a doorstop between uses. On the positive side, it's fully documented, universally applicable, and easily expandable to accommodate future EPROM types."

What a mess after it was published! Everybody must have built this programmer. BYTEnet almost shut down the Peterborough phone company as people downloaded the BASIC listings, and my staff developed "postage tongue" replying to the correspondence. Needless to say, the project was well received.

didn't see it as having any greater performance than low-cost bus-compatible programmer boards. I didn't arrange to have it made into a printed circuit board as are most of my projects. I'm embarrassed to say that even after all these years I underestimated the number of experimenters who wanted to build a serial EPROM programmer.

It's too late to go back now, but I have to make up for past indiscretion and find some way to save face. I know that there are warmed soldering irons all across the country waiting for me to apologize appropriately. I trust you'll accept this improved rendition on an old theme as proper recompense.

As the title indicates, this programmer is still intended for serial-port operation. Thus, it retains computer and bus independence. The primary difference between then and now, however, is the addition of a microprocessor that greatly enhances its functions. The new Circuit Cellar intelligent serial EPROM programmer (CCSP for short) programs more types of EPROMs faster and

In truth, it was an experimenter's project intended to satisfy a certain core of supporters yet enlighten the larger audience of readers about EPROM programming in general. Because I could not gauge its potential reception, and also because I

more reliably. It also functions as a stand-alone programmer for copying or verifying EPROMs. (See photo 1.) The following is a list of CCSP features:

- RS-232-compatible (no handshaking necessary)
- internal V_{pp} power generation
- menu-selectable EPROM types (no programming configuration jumpers)
- default power-up selectable data rates
- automatic power-down of EPROM for installation/removal
- stand-alone or computer system/terminal-connected operation
- menu-driven operation
- single-byte or full-buffer write modes
- 32K-byte on-board memory buffer
- read, copy, or verify EPROM
- Intel hexadecimal file upload/download
- verify after write
- verify EPROM erasure
- screen dump by page or byte
- BASIC driver that can be modified by the user
- program EPROMs in standard 50-

millisecond and 1- μ s fast algorithm modes

- support V_{pp} settings of 25, 21, and 12.5 volts
- program all 27xxx 5-V single-supply EPROMs, including 2716, 2732, 2732A, 2764, 2764A, 27C64, 27128, 27128A, 27C128, 27256, 27512, and any functional equivalents

Obviously, a list this impressive would take a great deal of effort to put together as a single chapter's project. The potential software development nightmares of assembly language serial drivers, menu displays, and table manipulations hardly made it worth adding a microprocessor to my original BASIC-language-manipulated unit. Besides, how could it be done in one month?

BASIC allowed a significant level of interactive menus and help displays while requiring little software overhead. Unfortunately, using a high-level-language interpreter to simplify software development is of little value when the primary goal of producing a better programmer requires fast data manipulations that are best ac-

complished in assembly language.

Rather than be thwarted by this apparent dilemma, I decided to design a hybrid system that used both BASIC and assembly language. The obvious choice was the BASIC-52 computer controller I presented in the August 1985 project. With the help of software guru and friend Bill Curlew, the CCSP was designed, built, and tested in two weeks flat.

The CCSP uses an Intel 8052AH-BASIC microprocessor that contains an 8K-byte ROM-based BASIC interpreter. Besides manipulating strings tables, and menus, the BASIC contains serial communication drivers and easily links to assembly language routines. It seemed the perfect engine for a quickly designed user-modifiable project.

A HYBRID APPROACH

The CCSP is a stand-alone microcomputer with an application-specific I/O configuration. It supports 40K bytes of operating system and buffer RAM and 16K bytes of program ROM. It uses six parallel I/O ports to drive the programming-pin level-shifter voltage-

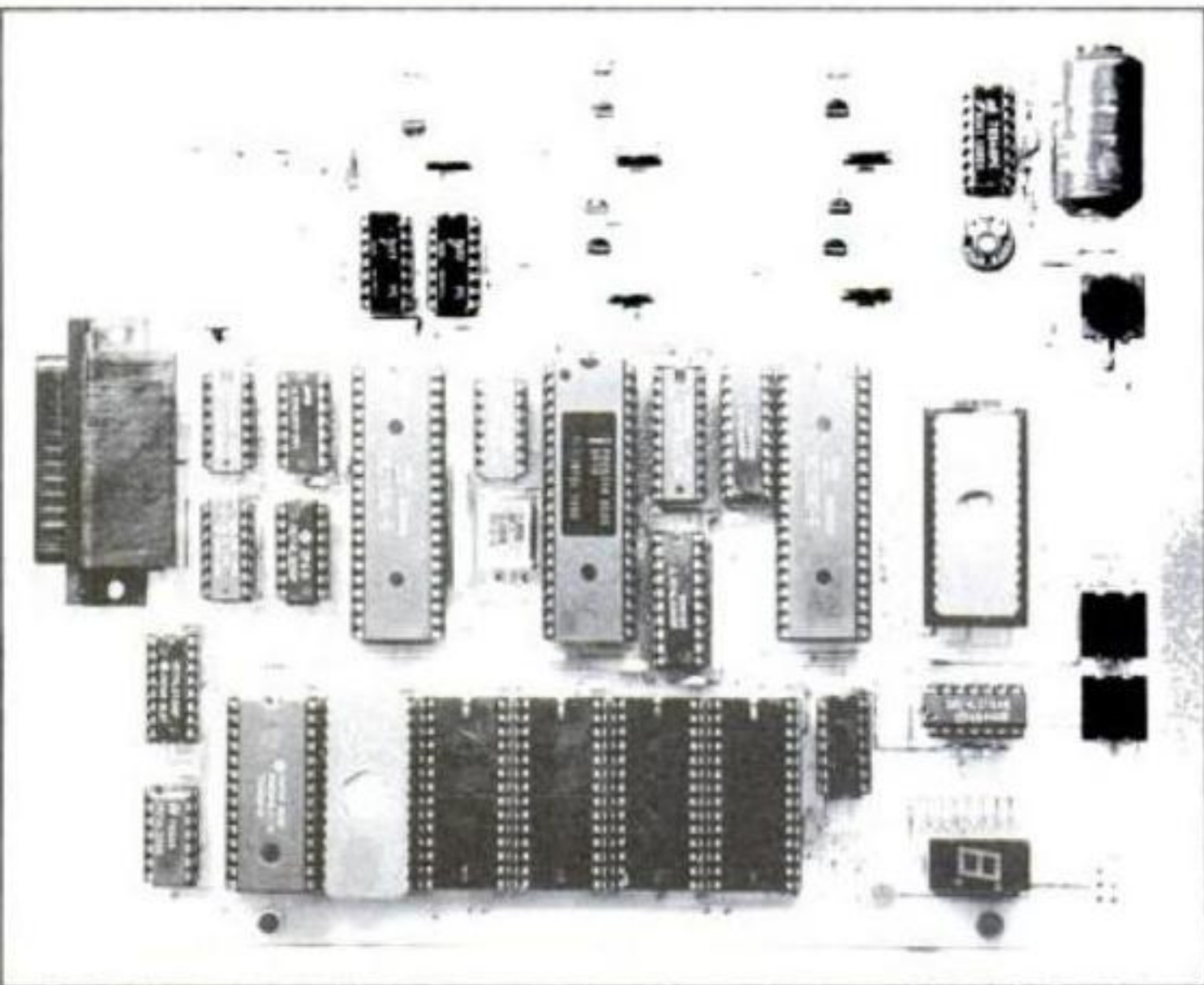


Photo 1: Finished printed circuit prototype of the serial EPROM programmer. The digital section and memory buffer are at the bottom center. The analog-voltage-level switching section is at the top center.

control circuitry, EPROM address and data lines, and user-interactive buttons and display. The CCSP can be used by itself to copy EPROMs or, when connected to a terminal or computer, as a full-function programmer/verifier. It requires no programming jumpers or personality modules and is completely automatic. It programs/examines/verifies all 5-V EPROMs from 2716s through 27512s in both standard and fast modes (on applicable devices).

In the sections that follow, I'll describe the configuration of the microcomputer and its unique I/O structure. Once you have the hardware in hand, I'll describe the system software and how the different modes operate. First, I'll go over some EPROM basics.

A REVIEW

A personal computer, even in its minimum configuration, always contains some user-programmable memory, or RAM, usually in the form of semiconductor-memory integrated circuits. This memory can contain

both programs and data. Any machine-word-level storage element within the memory can be individually read or modified (written) as needed.

Any of several kinds of electronic components can function as bit-storage elements in this kind of memory. TTL-type 7474 flip-flops, bistable relays, or tiny ferrite toroids (memory cores) are suitable, but they all cost too much, are hard to use, and have other disadvantages.

In personal computer and other microprocessor-based applications, the most cost-effective memory is made from MOS integrated circuits. Unfortunately, data stored in these semiconductor RAMs is volatile. When the power is turned off, the data is lost. Many ways of dealing with this problem have been devised, with essential programs and data usually stored in some nonvolatile medium.

In most computer systems, some data or programs are stored in ROM. A semiconductor ROM can be randomly accessed for reading in the same manner as the volatile memory, but the data in the ROM is perma-

wavelength of 2537 angstroms. Conveniently, most EPROM chips are packaged in an enclosure with a transparent quartz window.

HOW AN EPROM WORKS

EPROMs from several manufacturers store data bits in cells formed from stored-charge FAMOS (floating-gate avalanche-injection metal-oxide semiconductor) transistors. Such transistors are similar to positive-channel silicon-gate field-effect transistors, but with two gates. The lower or floating gate is completely surrounded by an insulating layer of silicon dioxide, and the upper control or select gate is connected to external circuitry.

The amount of electric charge stored on the floating gate determines whether the bit cell contains a 1 or a 0. Charged cells are read as 0s; uncharged cells are read as 1s. When the EPROM chip comes from the factory, all bit locations are cleared of charge and are read as logic 1s; each byte contains hexadecimal FF.

When a given bit cell is to be burned from a 1 to a 0, a current is passed through the transistor's channel from the source to the gate. (The

*When data is
to be erased
from the chip,
it is exposed to
ultraviolet light.*

The 27xxx EPROMs contain bit-storage cells configured as individually addressable bytes. This organization is often called "2K by 8" for a 2716 or "8K by 8" for a 2764. The completely static operation of the device requires no clock signals. The primary operating modes include read, standby, and program (program-inhibit and program-verify modes are important primarily in high-volume applications).

Control inputs are used to select the chip and configure it for one of these operating modes. In the program mode, particular bit cells are induced to contain 0 values. Both 1s and 0s are in the data word presented on the

nent. The data in a mask-programmed ROM is determined during the manufacturing process. Whenever power is supplied to the ROM, this permanent data (or program) is available. In small computer systems, ROM is chiefly used to contain operating systems and/or BASIC interpreters—programs that don't need to be changed.

Another type of ROM is the PROM, which is delivered from the factory containing no data. The user decides what data to put in it and permanently programs it with a special device. Once programmed, PROMs exhibit the characteristics of mask-programmed ROMs. You might label such PROMs "write-once" memories.

The ultraviolet-light erasable EPROM is a compromise between the "write-once" kind of PROM and volatile memory. You can think of the EPROM as a "read-mostly" memory, used in read-only mode most of the time but occasionally erased and reprogrammed as necessary. The EPROM is erased by exposing the silicon chip to ultraviolet light at a

electrons, of course, move the opposite way.) At the same time, a relatively high-voltage potential is placed on the transistor's upper select gate, creating a strong electric field within the layers of semiconductor material. (This is the function of the +12.5-V, +21-V, or +25-V V_{pp} charging potential applied to the EPROM.) In the presence of this strong electric field, some of the electrons passing through the source-drain channel gain enough energy to tunnel through the insulating layer that normally isolates the floating gate. As the tunneling electrons accumulate on the floating gate, it takes on a negative charge, which makes the cell contain a 0.

When data is to be erased from the chip, it is exposed to ultraviolet light, which contains photons of relatively high energy. The incident photons excite the electrons on the floating gate to sufficiently high-energy states that they can tunnel back through the insulating layer, removing the charge from the gate and returning the cell to a state of 1.

data lines, but only a 0 causes action to take place. For example, the 27128 is in the programming mode when V_{pp} input is at 21 V and CE and PGM are both at TTL low. The data to be programmed is applied 8 bits in parallel to the data output pins. For regular programming, CE should be kept TTL low at all times while V_{pp} is kept at 21 V. When the address and data are stable, a 50-ms (55 ms maximum) active-low TTL program pulse is applied to the $\overline{\text{PGM}}$ input. A program pulse must be applied at each address location to be programmed.

STANDARD VS. FAST

In the old days, all we had to contend with were 50-ms timing pulses (neglecting obsolete 1702 and 2708 EPROMs). Today, the newest EPROMs can use a fast closed-loop programming algorithm that lessens programming time (realize that a 27512 takes about 1 hour to program in 50-ms increments). The CCSP supports fast programming.

The fast algorithm uses closed-loop margin checking. To ensure reliable program margin, the fast algorithm utilizes two different pulse types: initial and overprogram. The duration of the initial PGM pulse(s) is 1 ms, which will then be followed by a longer overprogram pulse of length $4x$ ms; some

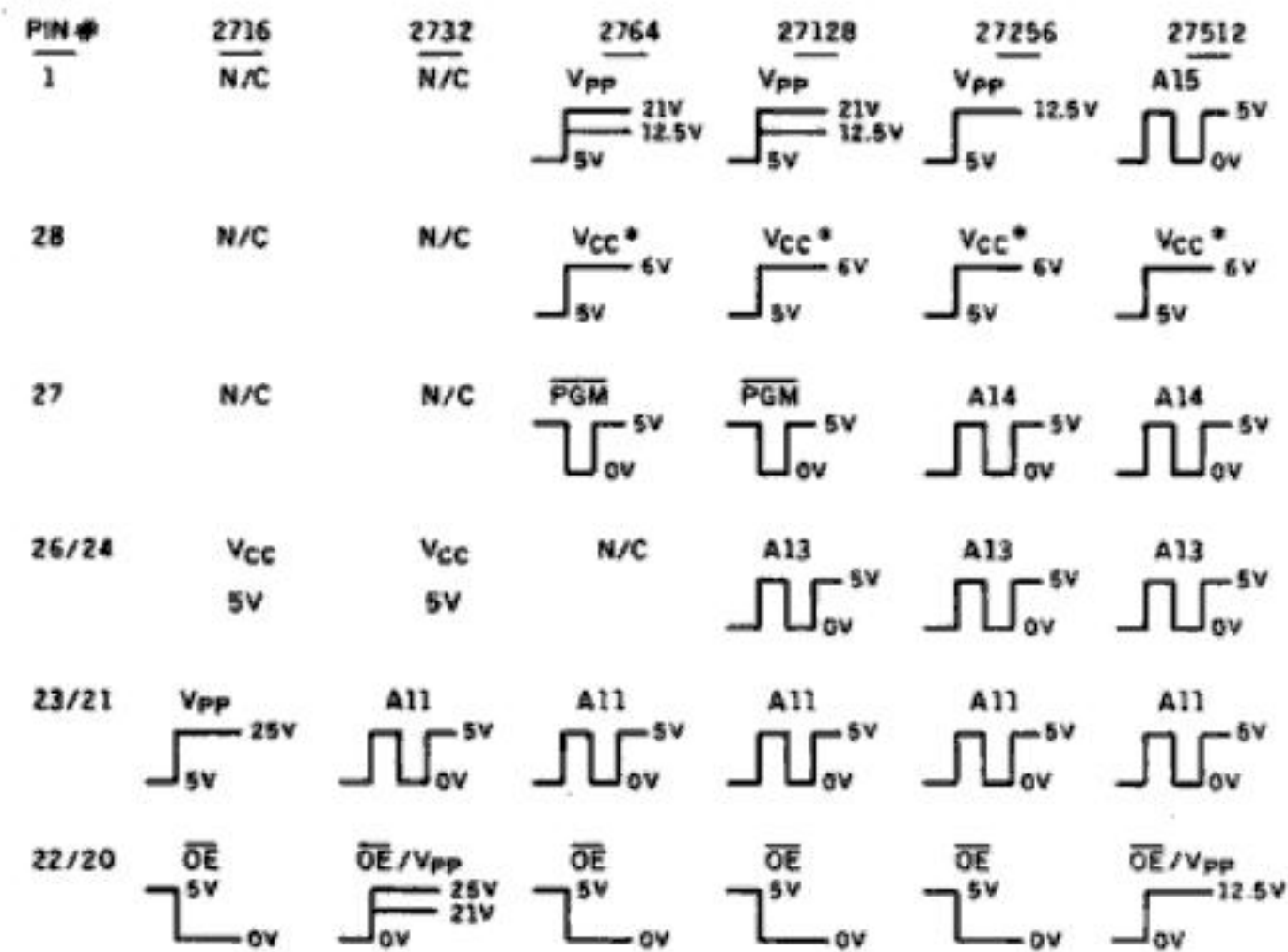
chip types use $3x$ (x is the number of initial 1-ms pulses applied to a particular location before a correct verify occurs). Once it is verified, four times that number of pulses are applied to the same location to permanently burn the data. If 15 (some chip types require 25 pulses) 1-ms pulses are ap-

plied to any single-byte location without reaching the margin, the overprogram pulse is applied automatically.

The entire sequence of program pulses and byte verifications is performed at $V_{cc} = 6.0$ V and $V_{pp} = 21.0$ V (V_{pp} may be 12.5 V on some EPROMs). When the fast programming cycle has been completed, all bytes should be compared to the original data with $V_{cc} = V_{pp} = 5.0$ V.

The fast algorithm may be the preferred programming method since it allows certain EPROMs to be programmed in significantly less time than the standard 50-ms-per-byte programming routine. Typical programming times for 27128s, for example, are on the order of 2 minutes, a six-fold reduction in programming time from the standard method.

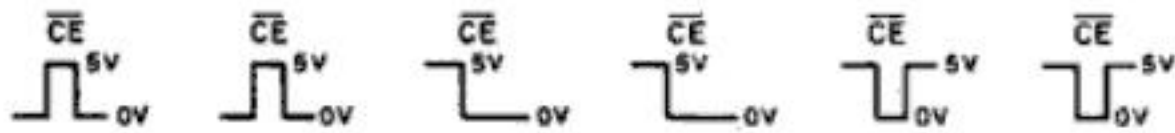
(a)



CONFIGURATION MAZE

The first problem encountered in any EPROM programmer design is to compare the pins of the various EPROMs (see figure 1b). Among the 28 defined pins (four are unused on 24-pin devices), 21 are used for the same functions (address and data). Evidently, semiconductor manufac-

20/18



* 6.0 VOLTS ONLY ON EPROMS THAT ALLOW FAST PROGRAMMING

Evidently, semiconductor manufacturers never thought very far ahead or talked to each other, because the remaining seven pins are a complicated switching maze. Among the different EPROMs, the same pin location can

(b)

27512	27256	27128	2764	2732A	2716	27XXX						2716	2732A	2764	27128	27256	27512
A15	V _{PP}	V _{PP}	V _{PP}			1	28				V _{CC}		V _{CC}	V _{CC}	V _{CC}	V _{CC}	
A12	A12	A12	A12			2	27				$\overline{\text{PGM}}$		$\overline{\text{PGM}}$	A14	A14	A14	
A7	A7	A7	A7	A7	A7	3	26			V _{CC}	V _{CC}	N.C.	A13	A13	A13	A13	
A6	A6	A6	A6	A6	A6	4	25			A8	A8	A8	A8	A8	A8	A8	
A5	A5	A5	A5	A5	A5	5	24			A9	A9	A9	A9	A9	A9	A9	
A4	A4	A4	A4	A4	A4	6	23			V _{PP}	A11	A11	A11	A11	A11	A11	
A3	A3	A3	A3	A3	A3	7	22			$\overline{\text{OE}}$	$\overline{\text{OE}}/\text{V}_{\text{PP}}$	$\overline{\text{OE}}$	$\overline{\text{OE}}$	$\overline{\text{OE}}$	$\overline{\text{OE}}$	$\overline{\text{OE}}/\text{V}_{\text{PP}}$	
A2	A2	A2	A2	A2	A2	8	21			A10	A10	A10	A10	A10	A10	A10	
A1	A1	A1	A1	A1	A1	9	20			$\overline{\text{CE}}$	$\overline{\text{CE}}$	$\overline{\text{CE}}$	$\overline{\text{CE}}$	$\overline{\text{CE}}$	$\overline{\text{CE}}$	$\overline{\text{CE}}$	
A0	A0	A0	A0	A0	A0	10	19			07	07	07	07	07	07	07	
O0	O0	O0	O0	O0	O0	11	18			06	06	06	06	06	06	06	
O1	O1	O1	O1	O1	O1	12	17			05	05	05	05	05	05	05	
O2	O2	O2	O2	O2	O2	13	16			04	04	04	04	04	04	04	
GND	GND	GND	GND	GND	GND	14	15			03	03	03	03	03	03	03	

Figure 1: (a) EPROM programming-pin functions by EPROM type. (b) The great EPROM pin-out maze, illustrating the configuration of those EPROMs the CCSP is designed to handle.

supply power, address, or programming pulses. Figure 1a illustrates the differences in detail.

In inexpensive programmers, configuration jumpers are frequently used to select the specific wiring configuration for different EPROM types. Wire jumpers rather than semiconductor switches are used because of the high currents involved. Take pin 26 (pin 24 on 24-pin EPROMs) with either a 2732 or 27128 installed, for example. In both cases, the voltage level is 5 V. On a 27128 it is a TTL A13 address line; on a 2732 it is a 150-milliampere V_{cc} power line. Similarly, pin 22 (all pin numbers are referenced to a 28-pin layout) has to be set at 0 V, 5 V, 12.5 V, 21 V, or 25 V at currents ranging from 400 microamperes to 50 mA, depending upon the EPROM.

Fortunately, only five of the seven configuration pins require elaborate voltage and current control. Rather than use mechanical jumpers, I designed a voltage-control circuit that could be preset to the voltage limits

ing a negative 1.2 V to the adjustment pin. Because there is no way to know how many of the control circuits will be set to 0 V at any one time or if the 7407 drivers are enabled concurrently, the -1.4-V bias source is itself a regulated supply.

The CCSP level-shifter circuit can simulate a variety of programmable conditions. For example, by setting the 7407 driver that limits the output to 5 V and pulsing the 0-V enable line, we have a TTL-level \overline{PGM} , \overline{OE} , or \overline{CE} control line. (In the tests I conducted, the circuit easily responded to control input changes of 20 kilohertz with little overshoot on the output. At those speeds, however, the output filter capacitor should be small.) Since the circuit is also capable of supplying 500 mA at 5 V, it is also appropriate to use this same circuit to supply and control V_{cc} .

The heart of the CCSP is found in the analog switching system and the management of the seven control lines in figure 1a. While I haven't ex-

I designed a circuit that could be preset to the voltage limits of the desired EPROM.

plained yet how these level shifters are individually controlled, it still seems appropriate to show how they are ultimately configured. Figure 3a demonstrates how they are connected to the ZIF socket (zero insertion force programming socket), and figure 3b outlines their power source connections.

8-BIT MICROCOMPUTER INTELLIGENCE

As I mentioned earlier, the CCSP's intelligence is provided by an Intel 8052AH microcomputer. BASIC-52 is particularly suited for this application.

of the desired EPROM type and easily pass high current when required. Figure 2 illustrates this basic circuit that is duplicated for each of the five pins (pins 28, 26, 1, 22, and 23).

The level shifter uses an LM317 voltage regulator as a programmable voltage controller. The basic LM317 output voltage is set by two resistors: R1 between the adjustment pin and ground and R2 between the adjustment pin and the output. As the formula shows, with R1=665 ohms and R2=221 ohms, the output is 5.0 V.

In this configuration, various R1 resistors can be connected from the adjustment pin to ground through open-collector 7407 drivers. These were used since they operate at up to 30 V (don't substitute a 7417). The four drivers from top to bottom set 5 V, 12.5 V, 21 V, and 25 V, respectively (not all sections are required for each EPROM pin). Their inputs are fed by a parallel output port.

Normally, the regulated output of an LM317 is 1.2 V to 32 V. An additional two-transistor control circuit is added to allow the output to go to 0 V on command. Rather than providing a resistance path to ground, however, this is accomplished by apply-

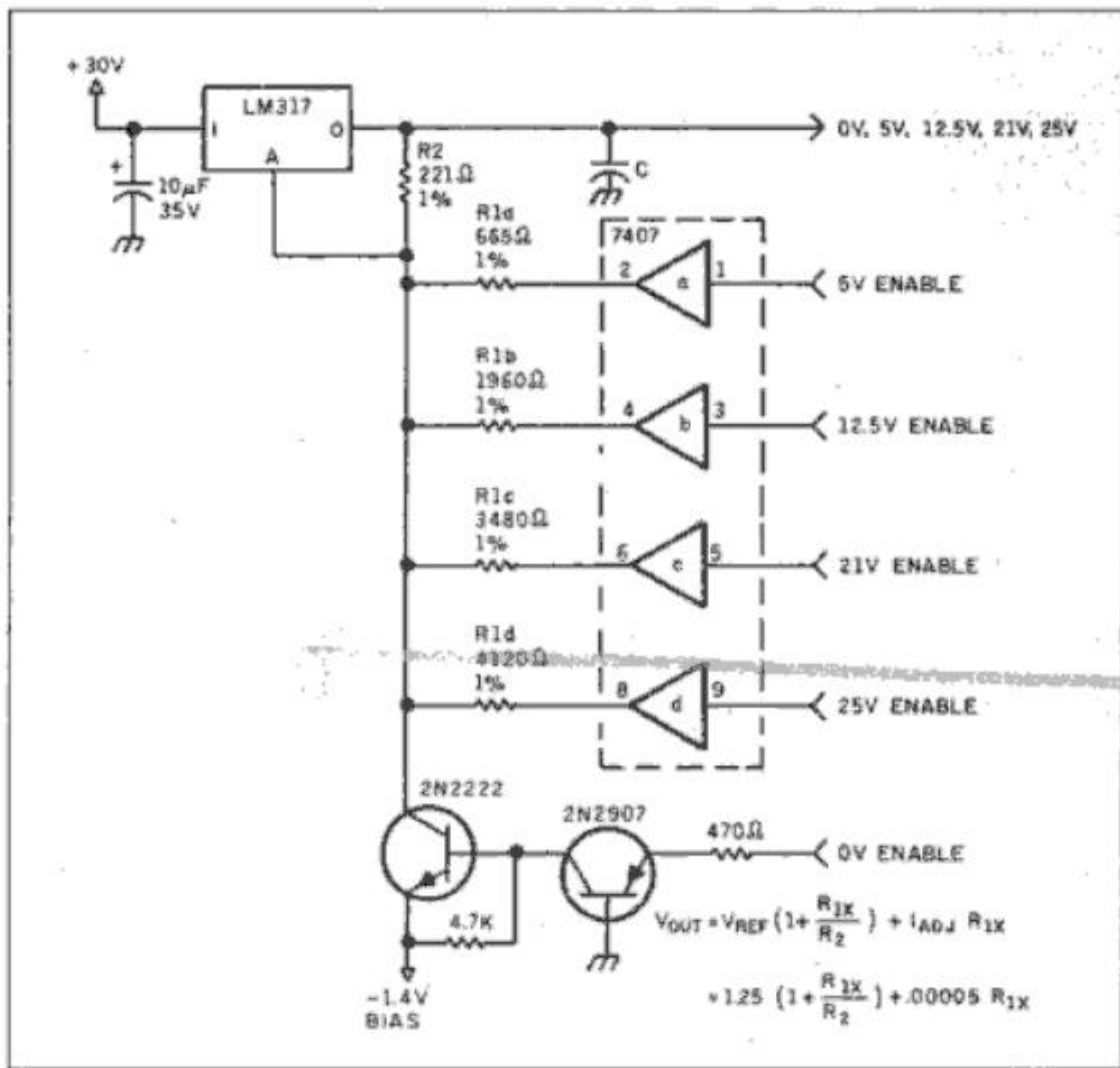


Figure 2: Typical programmable voltage-control circuit for EPROM pins 28, 26, 1, 22, and 23.

*Three control lines—
 \overline{RD} , \overline{WR} , and
 \overline{PSEN} —are gated
to allow 64K bytes
of combined program
and data memory.*

providing IF...THEN, FOR...NEXT, DO...WHILE/UNTIL, ON TIME, and CALL statements as well as a broad repertoire of 8051 assembly language instructions. Calculations can be handled in integer or floating-point math,

The 8052AH contains an 8K-byte BASIC interpreter in ROM, 256 bytes of RAM, three 16-bit counter/timers, six interrupts, and 32 I/O lines that are redefined as a 16-bit address and an 8-bit data bus. A minimum of 1K byte of RAM is required for BASIC-52 to function, and any RAM must be located starting at 0000 hexadecimal. (I won't go into great detail on this computer since it closely resembles the BCC-52 presented in August 1985.) The microcomputer section of the CCSP is outlined in figure 4.

Three control lines— \overline{RD} (pin 17), \overline{WR} (pin 16), and \overline{PSEN} (pin 29)—are gated to allow 64K bytes of combined program and data memory. The three most significant address lines (A13–A15) are connected to a 74LS138 decoder chip, IC4, which separates

the addressable range into eight 8K-byte memory segments, each with its own chip select (Y0–Y7). The four most significant chip selects are connected to 8K-byte 6264 static RAMs, ICs 7–10. This area is the RAM buffer for reading or writing EPROMs. IC6, addressed at 0000 hexadecimal, must be another 6264 RAM for BASIC-52 to function. IC11 (2000–3FFF hexadecimal) contains the programmer software and is intended for either a 2764 or 27128.

All together, 56K bytes of memory are defined on the CCSP if you use five 6264 RAMs (as ICs 6–10) and a 27128 EPROM in IC11. To use the programmer, you need only the one RAM chip installed in IC6 (such a limited buffer area will require many passes to write or copy any large

(a)

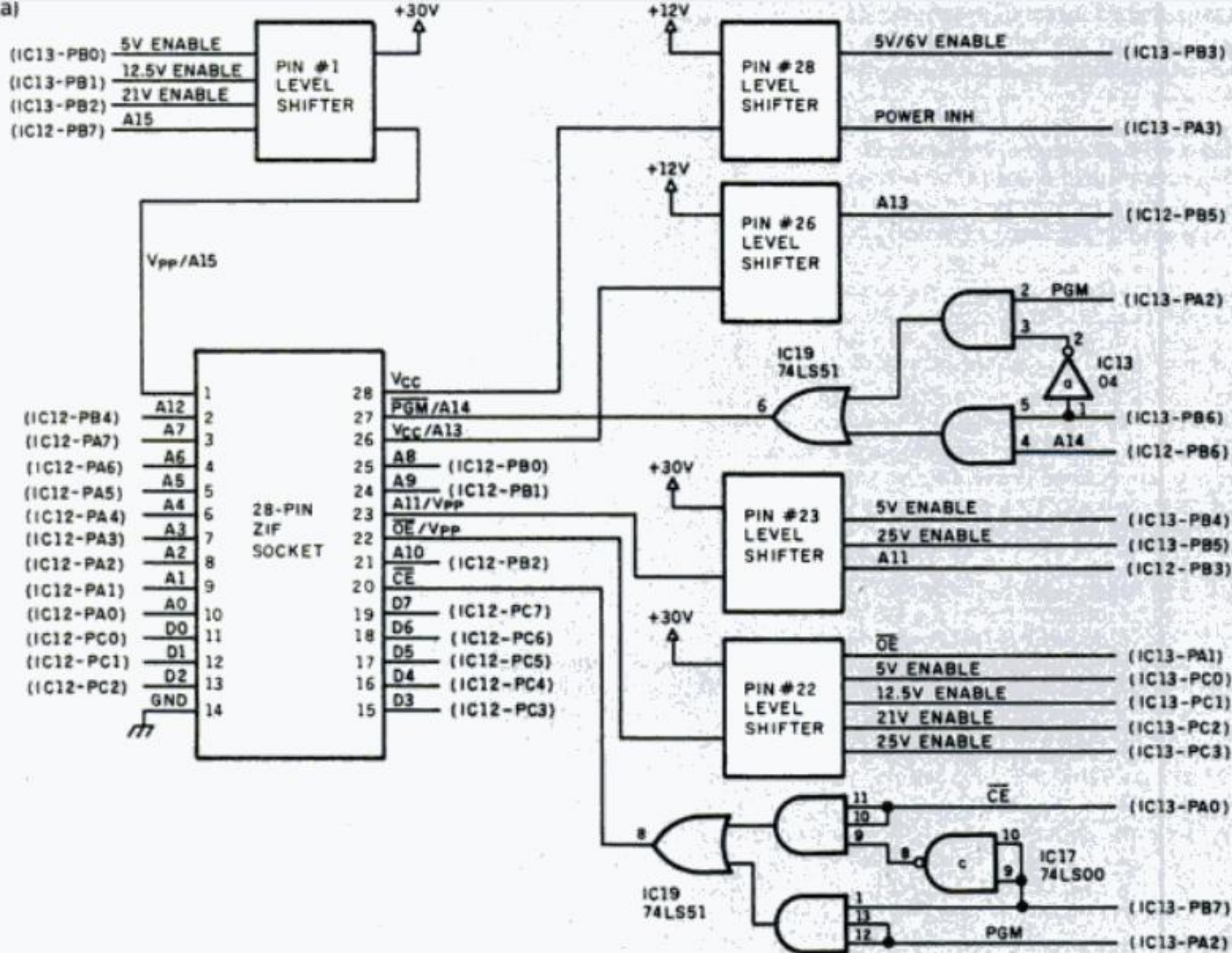


Figure 3: (a) Block diagram showing the connections to the ZIF socket. Note the level-shifter circuitry connections for those pins that require programming voltages or that differ across EPROM types.

EPR0M). The memory cannot be expanded since the rest of the address space is decoded as I/O.

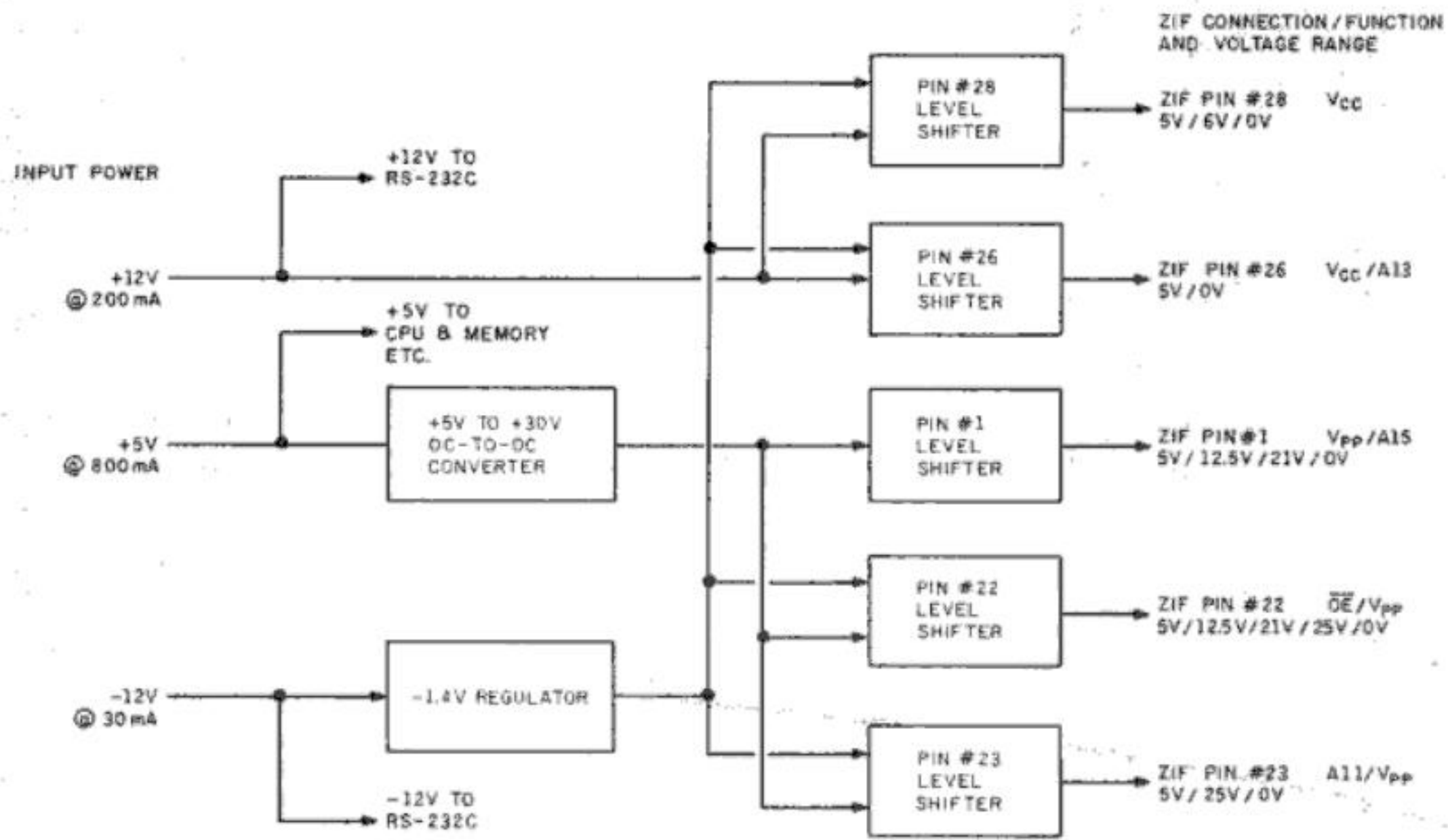
The address range of 6000-7FFF hexadecimal is divided into two I/O strobes at 6000 and 7000 through IC17. [Editor's note: For the remainder of the chapter, all addresses will be in hexadecimal.] Two 8255A-5 peripheral interface adapters providing three 8-bit I/O parallel ports each are controlled by a strobe line. The three I/O ports—labeled A, B, and C—and a write-only mode-configuration port on each 8255 occupy four consecutive addresses at 6000-6003 (IC12) and 7000-7003 (IC13), respectively. The ZIF socket and level-shifting circuitry outlined in figure 2 are connected to 41 of these parallel I/O bits. The lines attached to IC12 (the control PLA) are

used primarily for presetting the level shifters and providing the programming pulses. IC13 (the address and data PLA) supplies the address and data bus lines to the EPROM. Figure 5 details the configuration and connection of the level shifters and power distribution.

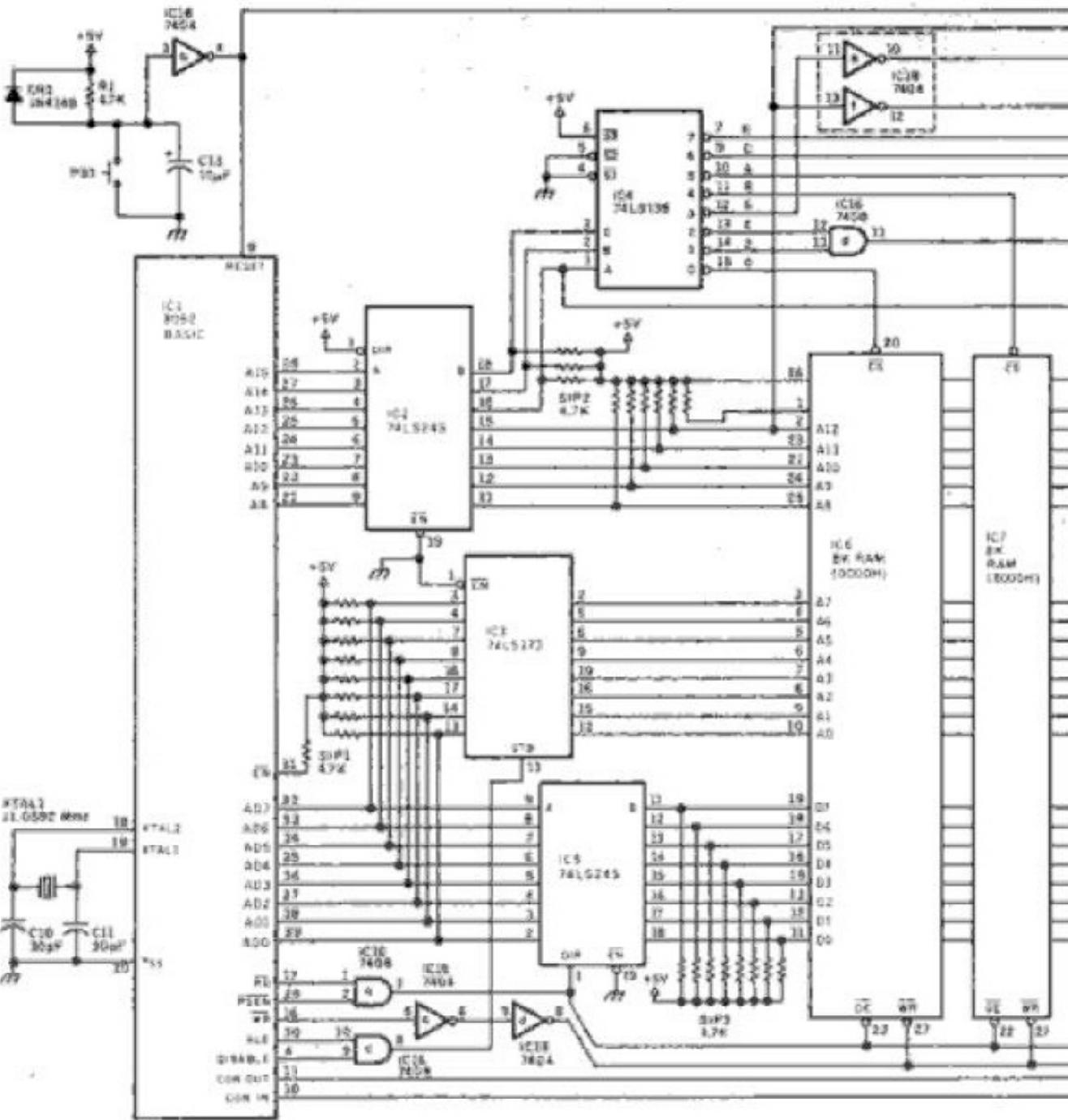
The CCSP communicates with a terminal or host computer through an on-board serial port. The port's data rate is hard-coded in the program ROM and is preset at 1200 bits per second, but you can reprogram it to any standard value between 300 and 19,200 bps. (The 8052AH chip has the capability for automatic data-rate selection on the console port. Because the CCSP has both a local and a remote operating capability triggered by the GET command, the

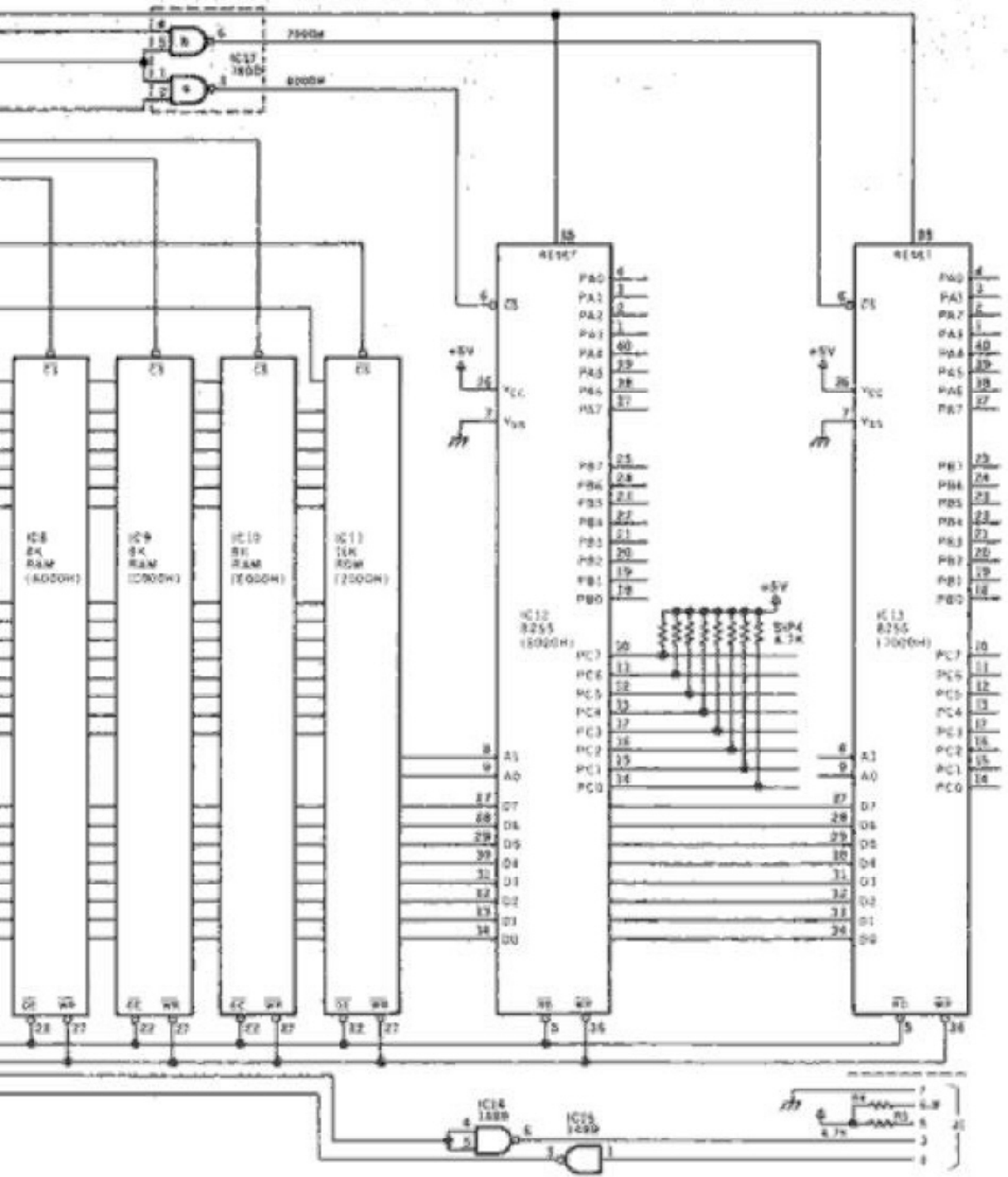
Power supplies with the required output are readily available, so I excluded an on-board supply to keep costs lower.

automatic data rate cannot be used.) MC1488 and MC1489 drivers/receivers (ICs 14 and 15) convert the 8052's serial I/O line TTL logic levels to RS-232.



(b) CCSP power-distribution block diagram.





Power for the CCSP is provided by an external supply that must deliver +12 V at 200 mA, +5 V at 800 mA, and -5 V to -12 V at 30 mA. Power supplies with these outputs are readily available on the surplus market, so I excluded an on-board supply to keep costs lower. In fact, a perfect unit is the Coleco computer power supply available from Radio Shack for \$4.95 (part #277-1022).

Three V_{pp} voltages must be contended with: 12.5 V, 21 V, and 25 V. All are derived from the +30-V output of the DC-to-DC converter circuit shown in figure 5. IC24 is a 78S40 switching regulator configured as a voltage multiplier. This circuit is capable of producing 30 V at 50 mA from a 5-V input. (For more information on this regulator and this specific

circuit, see my November 1981 article, "Switching Power Supplies: An Introduction.")

The user entry/display interface is shown in figure 6. It consists of a two-button entry panel through which you operate the programmer in local mode, a local/remote LED indicator, EPROM power-on indicator, and a seven-segment display through which the computer displays EPROM type and errors. To save I/O bits, I used a somewhat unorthodox display driver rather than the usual parallel port and seven-segment decoder configuration. The seven-segment LED is attached to an 8-bit shift register that has each output connected to drive an individual segment and the decimal point. To display a character, the seven-segment information is ex-

tracted from a memory-resident table and quickly shifted into the shift register. Ordinarily, I wouldn't use such a software-intensive approach, but I didn't have to write the software.

PROGRAMMER SOFTWARE

The CCSP is controlled by a program that is a combination of BASIC and 8051 assembly language. The BASIC-52 program provides all initialization and control functions, including local mode support and menu processing in the remote mode. The assembly language routines are used only where speed is critical, as in reading, comparing, verifying erasure, and programming EPROMs. In addition, the Intel hexadecimal file upload and download routines are written in

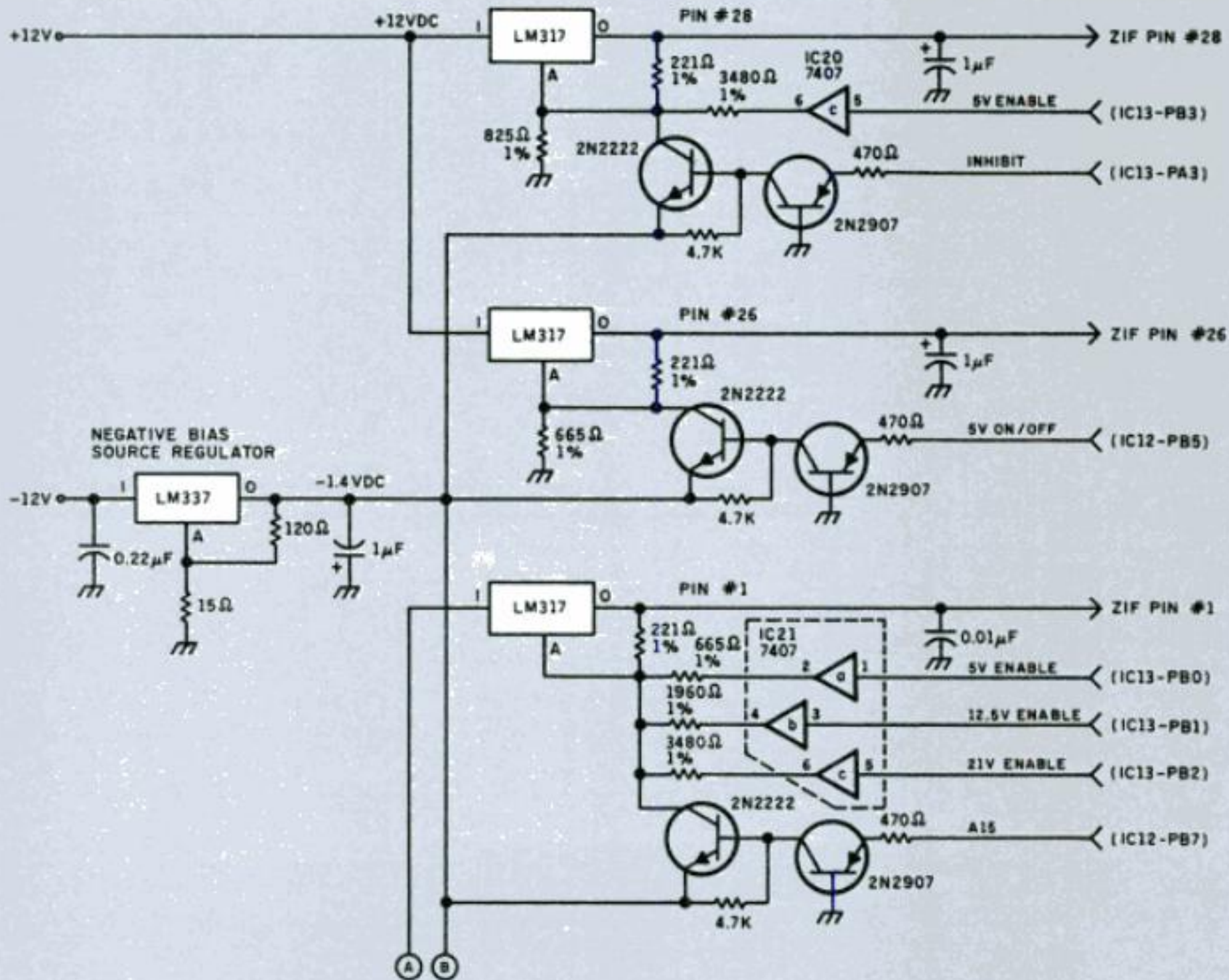


Figure 5: Detailed schematic of the programmer's level-shifting circuitry.

When working in local mode, you can copy any 27xxx EPROM by swapping the original and copy EPROMs multiple times.

assembly language to keep up with the attached ASCII terminal device. Figure 7 outlines the CCSP software logic flow.

The software that drives the CCSP is capable of running in two basic modes: local, where the CCSP acts as an EPROM copier controlled by but-

tons, and remote, where the CCSP acts as a full-featured programming workstation serially connected to the user's terminal. When working in local mode, you can copy any 27xxx EPROM regardless of its size by swapping the original and copy EPROMs multiple times. The larger the RAM buffer is, the fewer times you will have to change the EPROMs.

POWER-UP AND RESET

When the CCSP is first powered up or reset, its software configures itself for a 2716 EPROM, the default type. After setting up the hardware, the software outputs a 0 in the seven-segment LED display to indicate the EPROM type, turns on the local mode LED, and sizes the RAM buffer.

If no memory is located at 8000 (the buffer area), the CCSP allocates 4K bytes of system RAM in IC6 as the

buffer area. If it is unable to accomplish this, it will stop and display an alternating error code, E and 0, in the seven-segment LED display. Pressing a button or sending a character to the serial port will force the CCSP to reattempt sizing memory (memory sizing is destructive). If you have RAM chips plugged into locations IC7 through IC10, this will provide additional buffer memory. After memory is sized, the CCSP enters a loop to determine what mode you want the programmer to operate in.

During the mode-setting loop, the CCSP will decide if it is going to run in local or remote mode. The mode selected is determined by which event occurs first: If one of the buttons is pressed first, the CCSP establishes local mode; if a character is detected at the serial port first (via the BASIC-52 GET command), the pro-

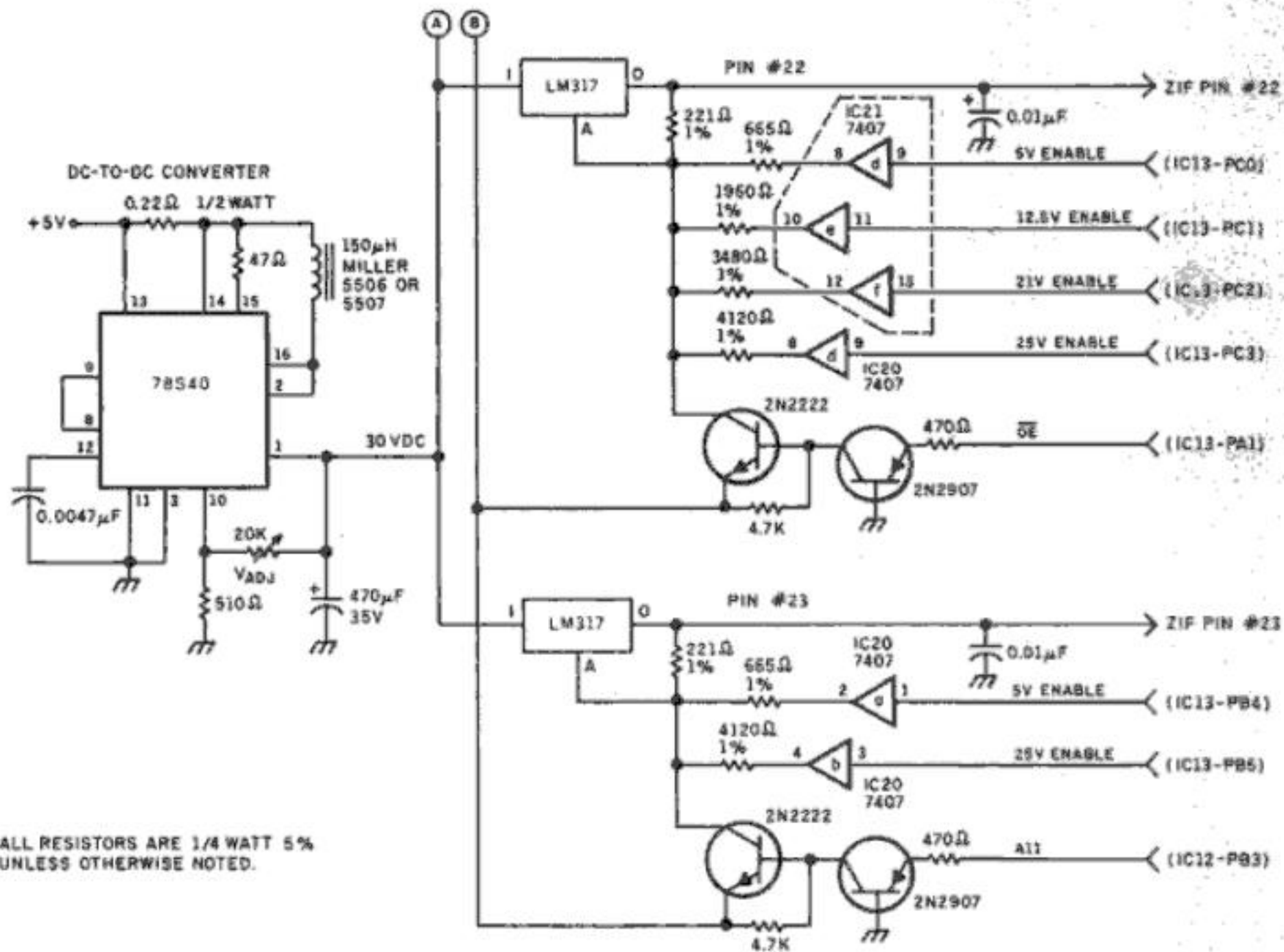


Figure 5 continued.

grammer enters remote mode. Once a mode has been selected, the CCSP must be reset or powered off/on to change modes.

STAND-ALONE LOCAL MODE

In local mode, the CCSP is controlled by two buttons called Type and Start/Next. Displays to the user are made via the seven-segment LED display. When local mode is initially entered, or at any point between completed programming cycles, you can change the designated EPROM type by pressing Type. Each press of the button steps the CCSP to the next EPROM type, and the seven-segment LED display is updated with the number that indicates the currently selected type. The designations are shown in table 1.

target EPROM or a 2 for a failed comparison. If an error does occur, you will be returned to the "between copies" state at the next press of the button.

Assuming all went well, the CCSP checks to see if the entire EPROM has been copied. If it has, the CCSP returns to the "between copies" state and displays the current EPROM type selected on the seven-segment display.

If the entire EPROM has not yet been copied, the effective starting address of the RAM buffer will be incremented by the size of the RAM buffer, and the CCSP will prompt you to insert the original EPROM again. This time, the programmer reads the EPROM starting at the new address.

The amount of data read will be either the RAM buffer size or the remaining bytes to be copied from the EPROM, whichever is less. After reading the original, the CCSP calls for the copy EPROM, and programming continues as described above.

These steps will continue until the entire contents of the original EPROM have been transferred into the copy EPROM. Using this approach allows any size EPROM to be copied, regardless of the amount of memory in the RAM buffer.

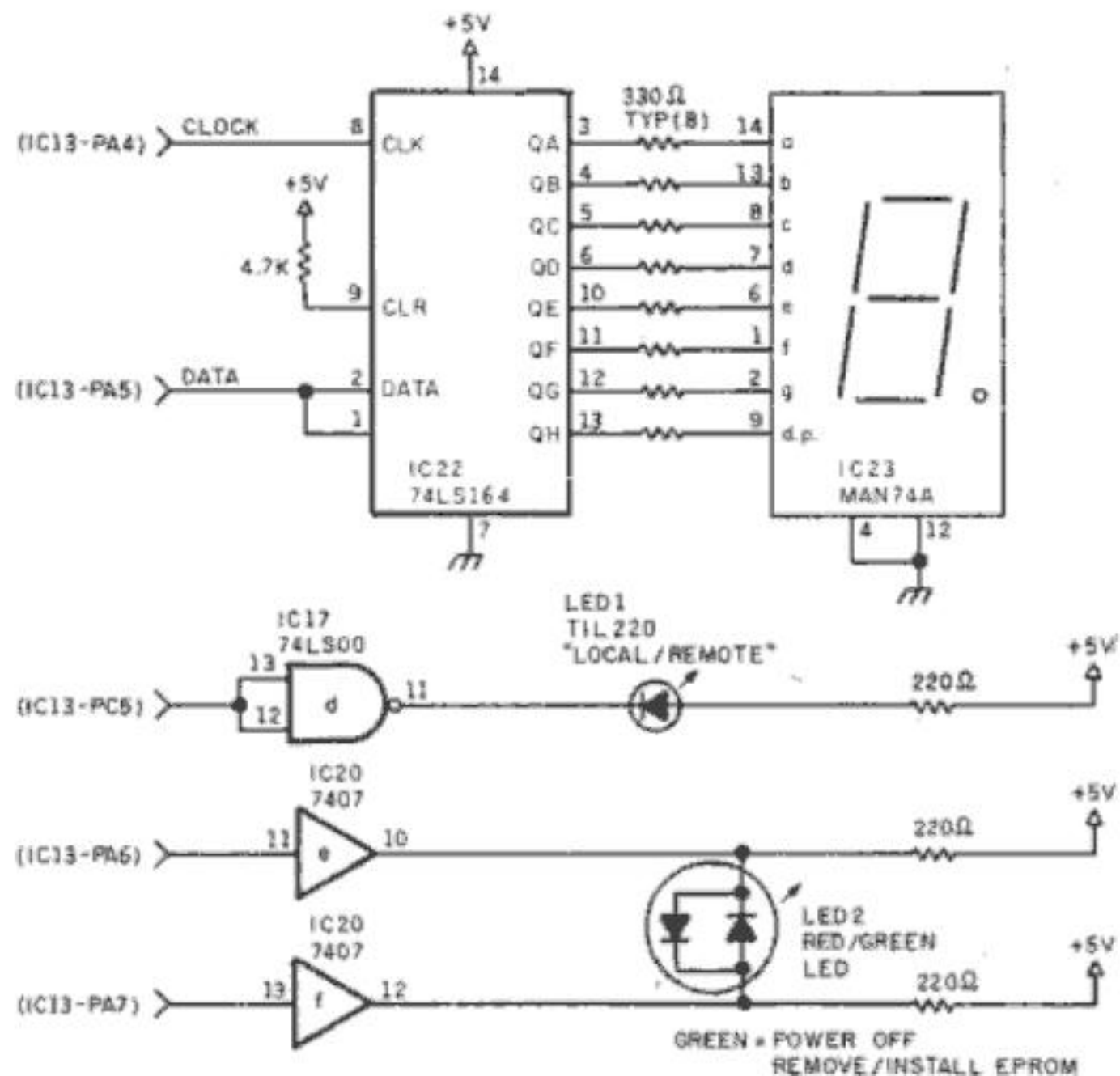
REMOTE MODE OPERATION

When used in remote mode, the CCSP turns into a menu-driven programming workstation, controlled by an

After setting the type of EPROM to work with, you begin the copy cycle by pressing Start/Next. At this point the seven-segment LED will display an alternating L and O, indicating that you should insert the original EPROM into the ZIF socket. You then load the original EPROM and press Start/Next again to begin the next step: reading the EPROM.

When the CCSP has read as much of the EPROM data as the memory buffer will allow, it signals you to remove the original EPROM and insert the copy EPROM by displaying an alternating L and C on the seven-segment display. After doing this, you again press Start/Next.

The CCSP will now attempt to program the contents of the RAM buffer into the copy EPROM. After verifying that the target area of the copy EPROM is erased, the letter "P" is displayed on the seven-segment display to indicate that programming is in progress (LED2 will be red, indicating that power is on to the EPROM and it should not be removed). When programming is complete, the contents



programming is complete, the contents of the EPROM are compared to the memory buffer contents. During this time, the letter "C" is displayed on the seven-segment display (LED2 will be green, indicating power off).

If the target EPROM is not erased or the programming was not successful (bad compare), the seven-segment LED will display an alternating E and a numeral, either a 1 for an unerased

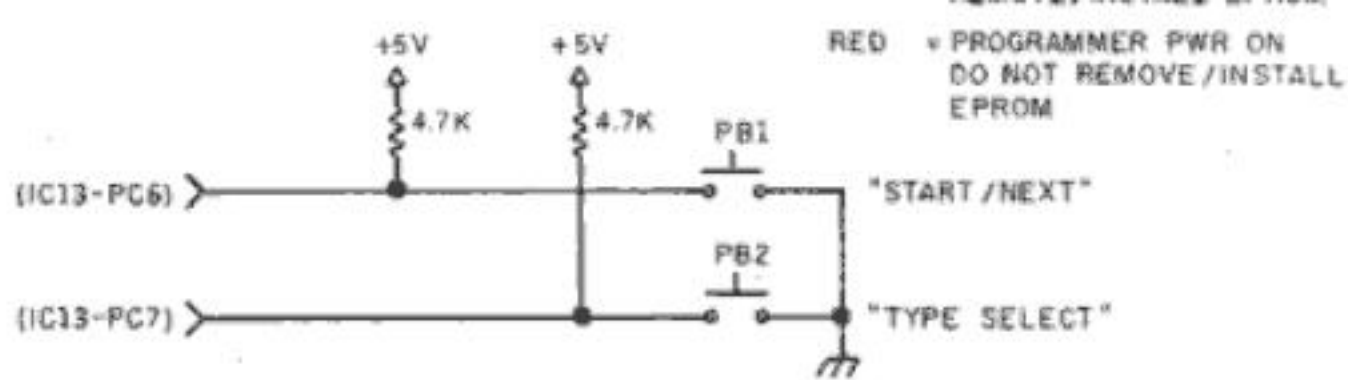


Figure 6: The CCSP's entry/display circuit.

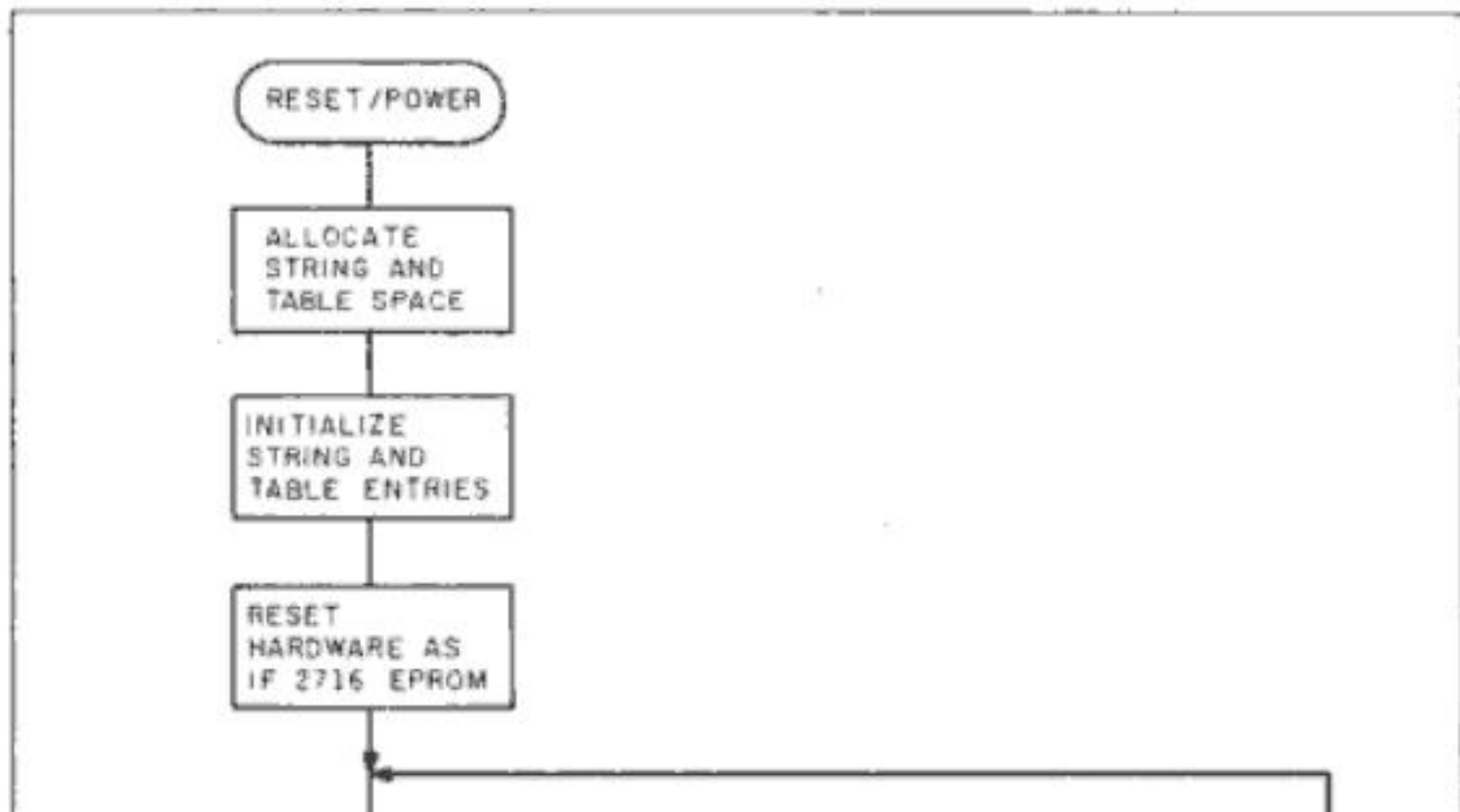
All menus displayed
on the terminal are
generated by the CCSP.
A terminal-emulation
program is the only
software necessary to
use this programmer.

ASCII terminal. (See photo 2.) The data rate of the terminal must be hard-coded because the 8052 cannot automatically start the BASIC-52 program unless the data rate is defined. Using the automatic data-rate feature causes the 8052 to wait for a "space" character from the serial port before executing any program stored in it; this would eliminate the local mode of the CCSP. The data rate is set at 1200 bps, but you can change it to

EPROM all depend on the RAM buffer, and they usually use the effective starting address and the length of the RAM buffer to determine the area of the EPROM that is being worked with. Think of the RAM buffer as a window into the contents of the EPROM. If the RAM buffer is not large enough to show you the whole EPROM, you can move it around by changing its effective starting address.

Let's use an example. The EPROM

type is a 27512, which is 64K bytes, and the RAM buffer is 16K bytes. It should be pretty obvious that you can't get the whole 27512 into the RAM buffer at the same time. In this case, you would set the starting address of the RAM buffer to 0000 to work with the first quarter of the EPROM and then set it to 4000 to work with the second quarter, 8000 for the third, and C000 for the last. The READ, COMPARE, and PRO-



any standard value by reprogramming the system ROM with the default data-rate byte changed (details on this procedure are included with the software).

All the menus displayed on the screen of the terminal or computer are generated by the CCSP. No software other than a terminal-emulation program (if connected to a computer rather than a real terminal) is necessary to use this programmer. The remote mode menu provides the following options:

- read, compare, program, and verify EPROM
- display and change RAM buffer contents
- download and upload Intel hexadecimal files
- set EPROM type
- set effective starting address of the RAM buffer

The menu screens contain enough information to guide you through the use of most of these functions. Other pertinent information on the various options is given below.

Read, compare, program, and verify

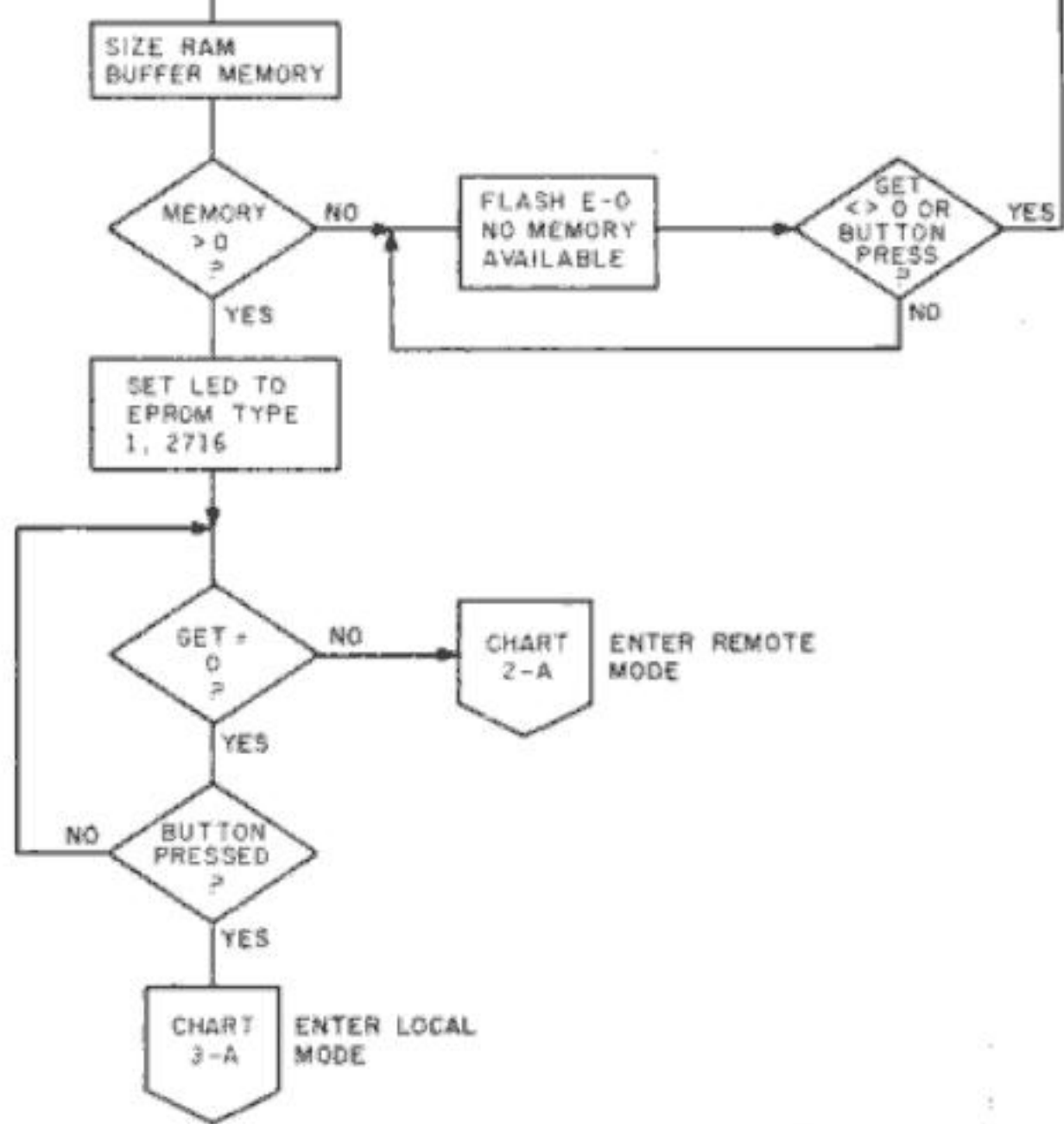


Figure 7a: Flowchart 1 of the CCSP's overall logic flow, showing the power-up and reset routines.

GRAM commands would use the starting address of the RAM buffer to see where to read data from or write data to the EPROM. The greatest length of the transferred data would be the size of the RAM buffer or the remaining number of bytes in the EPROM, whichever was smaller.

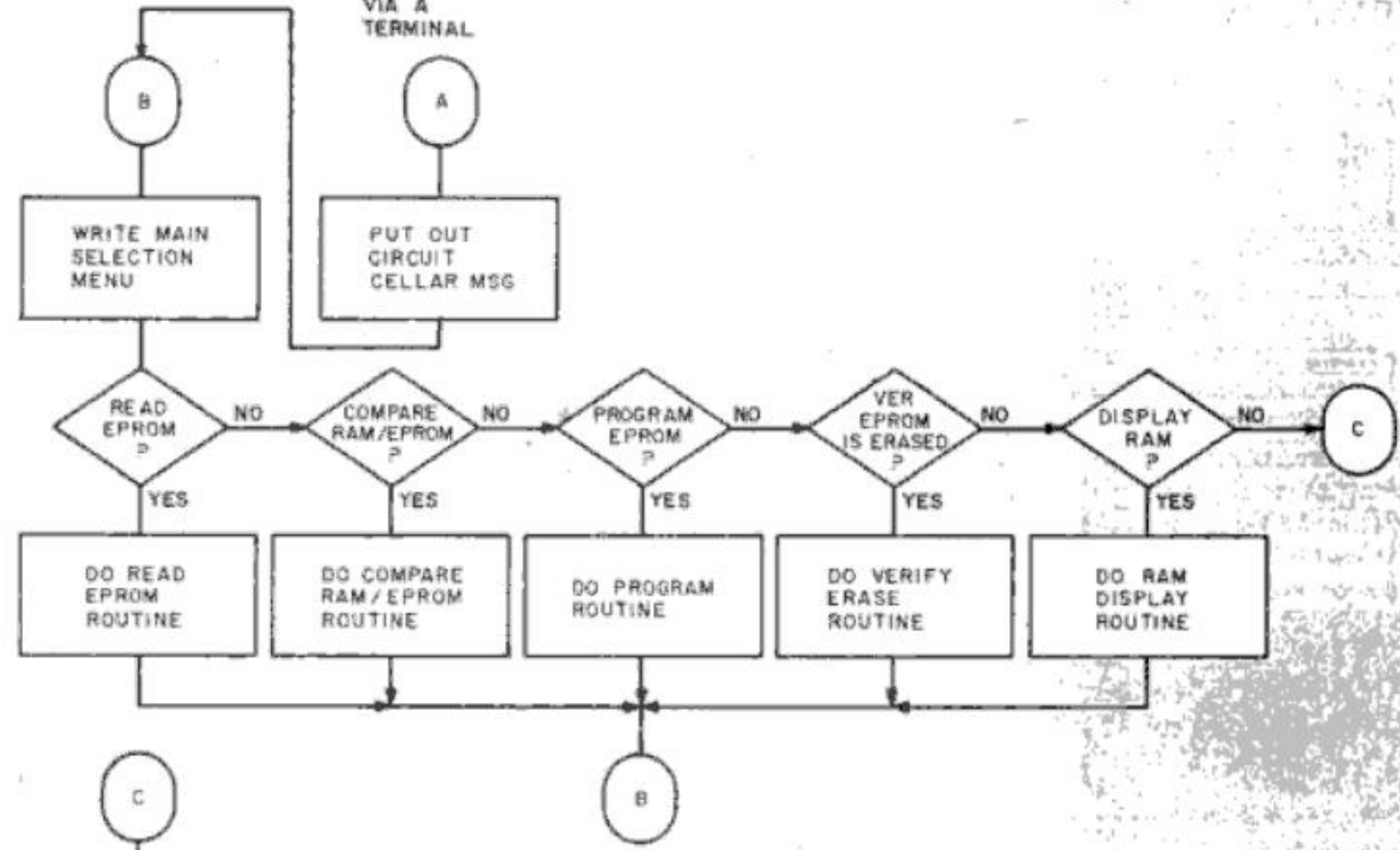
Even though the VERIFY command does not care about the size of the buffer, its default start and end addresses are controlled as described

above. This is because VERIFY generally precedes a programming cycle (you use VERIFY to confirm that the EPROM is properly erased), and the RAM buffer addressing controls programming default start and end addresses.

The following functions—display and change RAM buffer contents and download/upload Intel hexadecimal files—are also tied into the RAM buffer. Since the RAM buffer is supposed

to mirror the equivalent area of the EPROM, displays, changes, and uploads/downloads must be addressed to the RAM buffer, just as they would be in the real EPROM. This means that the software will reject addresses outside the range of the current RAM buffer area, which is especially important when doing uploads and downloads. These loads *must* be broken up to fit into the current RAM buffer area

THIS IS
REMOTE MODE.
VIA A
TERMINAL



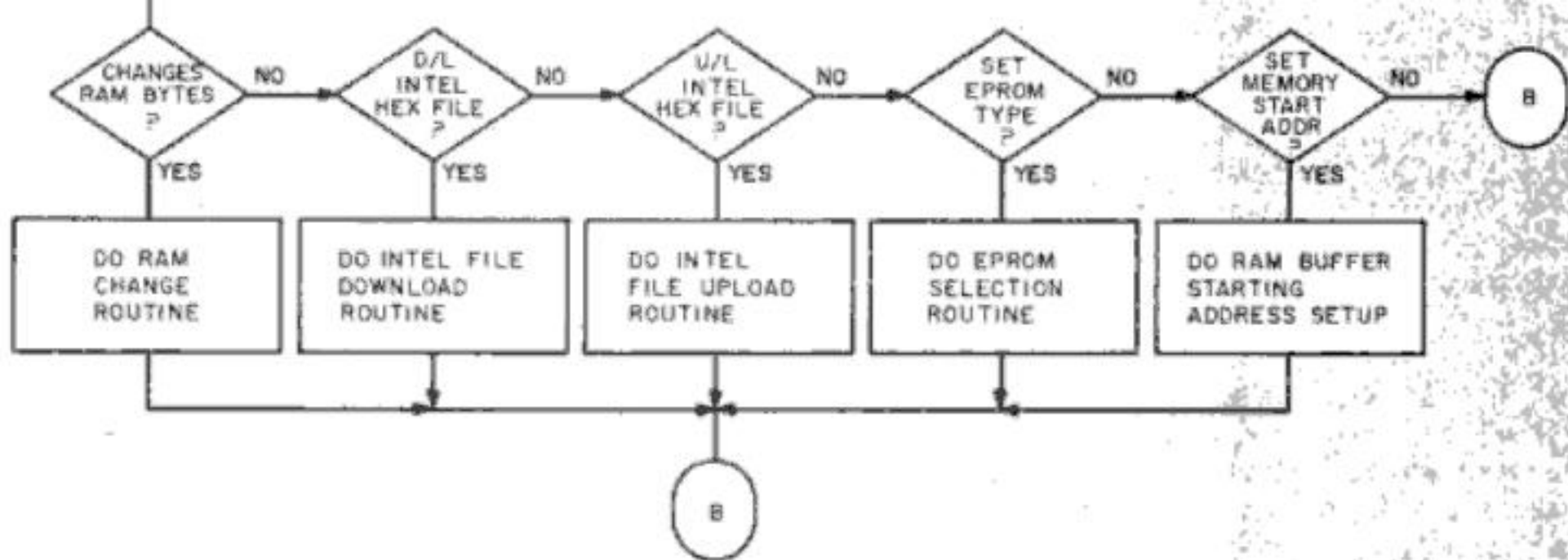


Figure 7b: Flowchart 2. logic flow for the remote mode.

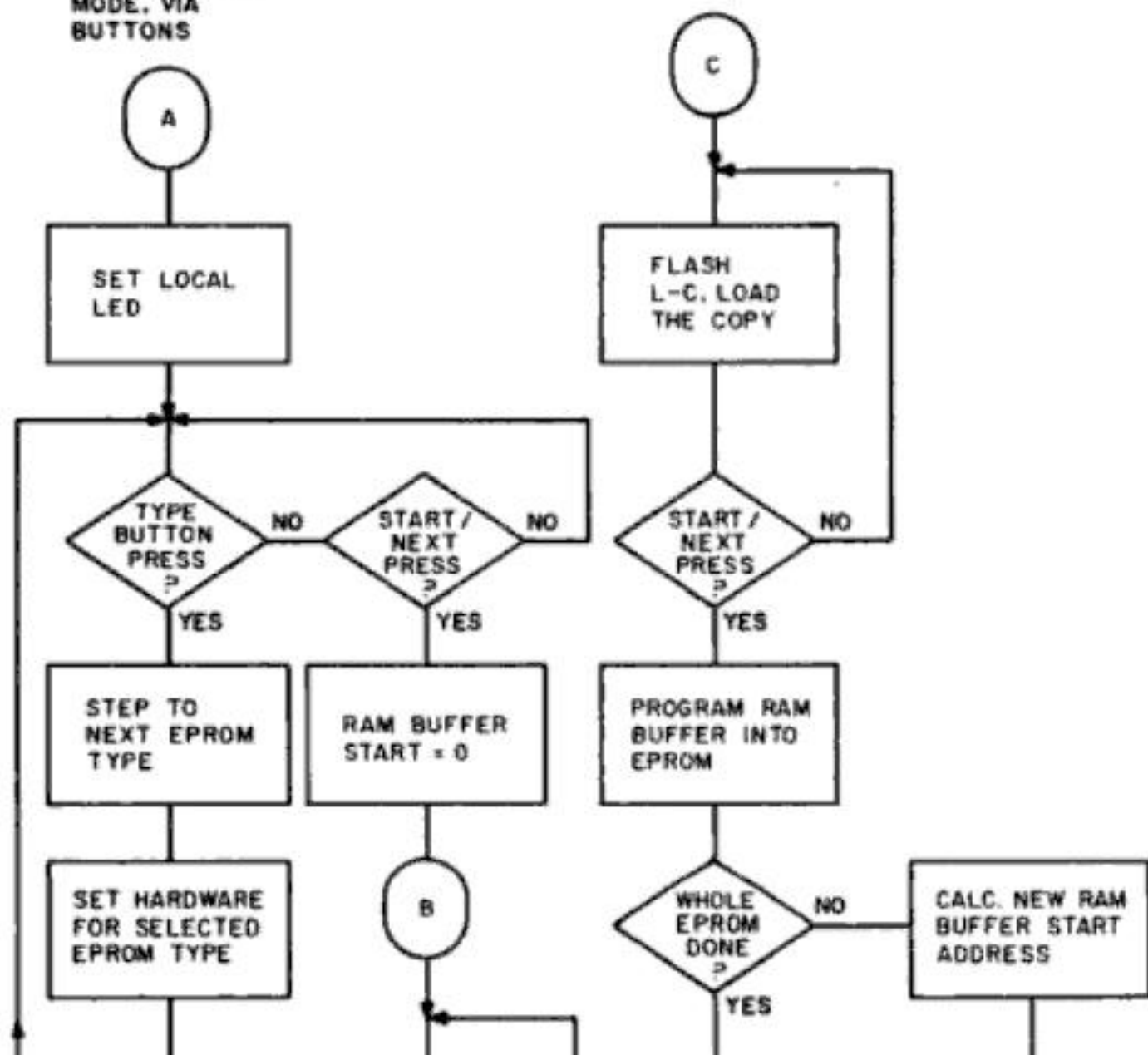
Table 1: The CCSP's EPROM selection number and corresponding EPROM types. Note that this list is of generic EPROM types, and other manufacturer designations should be cross-referenced to it. Also, since CMOS programming cycles are equivalent to those in standard EPROMs, separate 27Cxxx designations are not included.

Number	EPROM	Type	Number	EPROM	Type
0	2716	25 V	5	27128	21 V
1	2732	25 V	6	27128A	12.5 V
2	2732A	21 V	7	27256	12.5 V
3	2764	21 V	8	27512	12.5 V
4	2764A	12.5 V			

address range. Trying to go outside the range will abort the display/change/load processes.

The remaining two functions—set EPROM type and set RAM buffer starting address—let you deal with various EPROM types and manipulate the starting address of the RAM buffer. Setting the RAM buffer address lets you change the location of the window into the EPROM. This should be necessary only if the size of the EPROM exceeds the size of the RAM

THIS IS LOCAL
MODE. VIA
BUTTONS



buffer. Otherwise, there is no reason to change the starting address from its default value of 0000.

UNDER THE COVERS

In order to handle the various combinations of sizes, programming voltages, and control lines used with different EPROM types, the software incorporates control tables. Four such tables are used in the CCSP:

- system global table G(x)
- LED character table L(x)
- EPROM string table S(x)
- EPROM data table E(x)

SYSTEM GLOBAL TABLE

The system global table contains information about current values for critical system information. The table's entries are set up as is shown in table 2.

The values for these items change based on the type of EPROM you are using, how much contiguous memory is at address 8000 (or the 4K bytes stolen from system RAM), and the last bytes written to the 8255s.

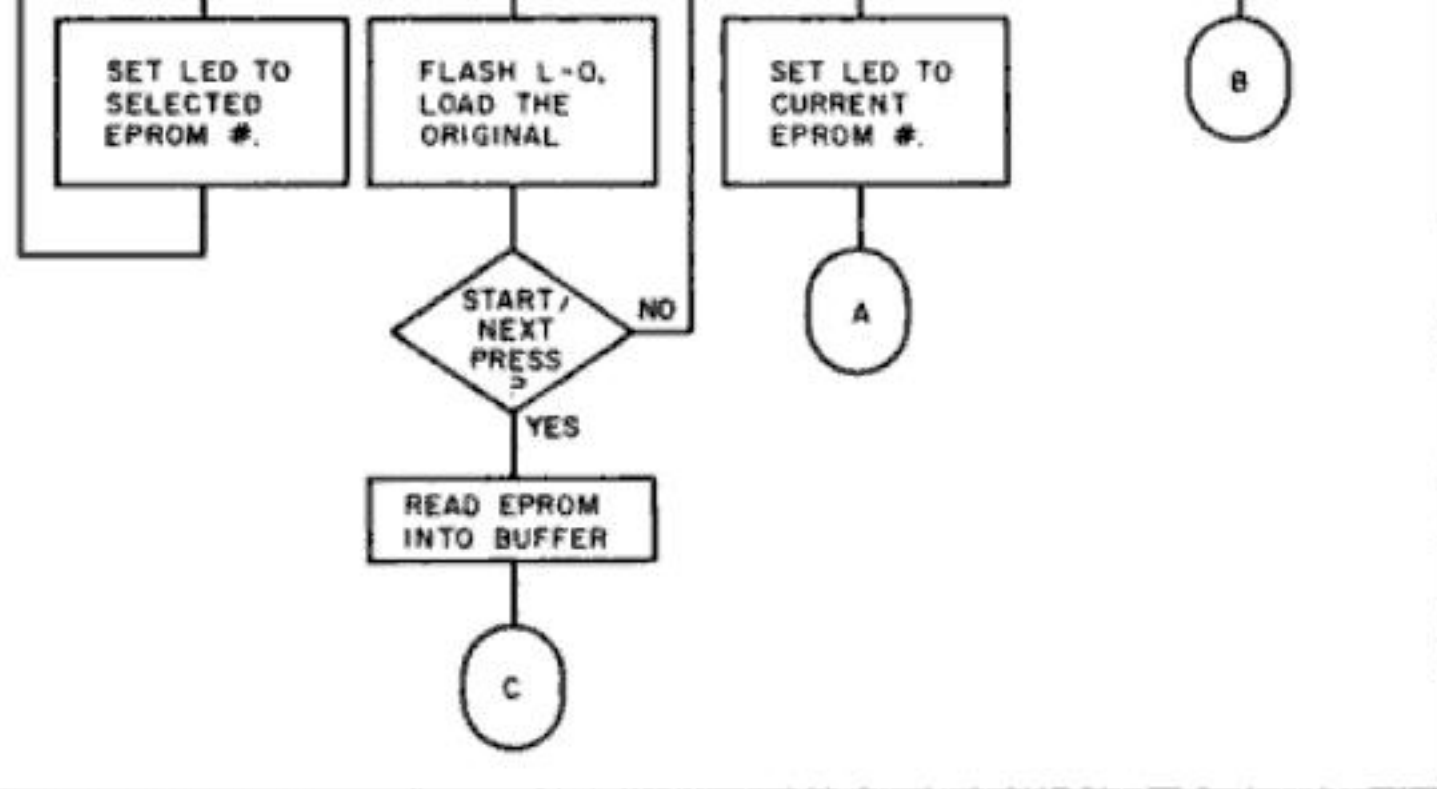


Figure 7c: Flowchart 3. logic flow for the local mode.

LED CHARACTER TABLE

The seven-segment LED display is controlled by a serial-to-parallel shift register. In order to create a character in the LED, the bits for the various segments must be shifted out in the correct order. This table contains the LED code byte needed to create the characters that can be displayed (see table 3).

EPROM STRING TABLE

BASIC-52 does not allow the mixing of text and numeric data in the same table, so the $S(x)$ string table function is used to store this information. This

table contains the EPROM designator and the programming voltage used with that type of EPROM. Actually, the programming voltage indicated in the table is only a reminder. You have to set the correct bits in the EPROM data table to ensure that the programmer uses the proper voltage.

EPROM DATA TABLE

The EPROM data table contains all the information the system requires to work with the different EPROM types. The items in each record of the EPROM data table are shown in table 4.

Listing 1 illustrates how this is handled in BASIC. This data is maintained for the use of both the BASIC and assembly language routines. BASIC passes data from the EPROM data table to the assembly language routines via the free registers of the 8052 device.

THE ASSEMBLY LANGUAGE ROUTINES

The CCSP software is a hybrid of BASIC and assembly language. Besides reading and verifying, the EPROM programming pulses are ac-

Table 2: Contents of the system global table.

Index	Use
0	Type number of the current EPROM.
1	Amount of RAM buffer available in 256-byte increments.
2	Current starting address of RAM buffer.
3	Number of items in EPROM table entry.
4	Reserved.
5,6,7	Value of the last data byte written to IC12 address/data PIA (3 bytes, one for each port of the 8255).
8,9,10	Value of the last data byte written to IC13 control PIA.
11	Number of EPROM types in the EPROM string and data tables.

Table 3: Contents of the LED character table.

Index	Use
0-9	Characters 0-9 (no decimal point)
10-15	Characters A-F (no decimal point)
16-25	Characters 0-9 (decimal point)
26-31	Characters A-F (decimal point)
32	Blanks LED
33	Character H
34	Character L
35	Character P
36	Character U

curately timed in assembly language routines. The derivation of the timing's accuracy is given in table 5.

IN CONCLUSION

At first look, the CCSP appears to be considerably more complicated than my programmer of 18 months ago. I think at this point I can change my new description to more accurately state that "this programmer is a serial-port programmer that has the speed of lightning, the intelligence of the mightiest computer (on-board), and is far too functional to be used as a doorstop between uses."

In actuality, only the explanation is more involved. With microcomputer intelligence, the CCSP achieves performance levels approaching kilobuck commercial units yet is flexible enough to be adapted to the next V_{pp} change when it happens.

I'm quite satisfied with my two-week miracle, but I still have to contend with a potential horde of builders. To make amends for my past indiscretion,

```
CIRCUIT CELLAR SERIAL EPROM PROGRAMMER  
COPYRIGHT 1986, CIRCUIT CELLAR INC.
```

```
PLEASE CHOOSE :
```

- 1 - READ EPROM DATA INTO MEMORY
- 2 - COMPARE RAM MEMORY BUFFER DATA TO EPROM
- 3 - PROGRAM RAM MEMORY BUFFER DATA INTO EPROM
- 4 - VERIFY EPROM IS ERASED
- 5 - DISPLAY (DUMP) RAM MEMORY BUFFER
- 6 - CHANGE RAM MEMORY BUFFER CONTENTS
- 7 - DOWNLOAD INTEL HEX FILE FROM TERMINAL
- 8 - UPLOAD INTEL HEX FILE TO TERMINAL
- 9 - SET EPROM TYPE
- 10 - SET RAM MEMORY BUFFER STARTING ADDRESS

```
CURRENT SETTINGS
```

```
EPROM TYPE 2716 25V SIZE 0800H BYTES  
RAM BUFFER STARTING ADDRESS 0000H , WITH 8192 BYTES AVAILABLE
```

```
ENTER YOUR CHOICE :
```

```
?  
█
```

Photo 2: Typical menu display presented when operating the serial EPROM programmer through a serial terminal or computer running in terminal-emulation mode.

Table 4: Contents of the EPROM data table.

Index	Use
0	Number of the EPROM.
1	Size of the EPROM in 256-byte increments.
2	The pin V_{ee} is applied to, referenced to a 28-pin socket.
3,4,5	The initialization values for IC13 control PIA's ports A, B, and C.
6,7	The programming mode values for IC13's ports A and B.
8	Logic true value of the programming pin (CE and PGM).
9	Normal mode programming pulse width in milliseconds.
10	Fast mode programming pulse in milliseconds (0 if no fast programming allowed).
11	Maximum number of fast programming pulses before forced overprogram pulse occurs.
12	Overprogram pulse multiplication factor.

Table 5: Derivation of the timing for the CCSP's 1-ms timing routine.

Label	Instruction	Frequency of Execution	Clock Periods Used
PPLOOP	EQU \$	A	0
	PUSH B	C	24
	CALL ONEMS	C	24
	POP B	C	24
	DJNZ B,PPLOOP	C	24
	RET	O	24
ONEMS	EQU \$	A	0
	MOV B,#MSDLAY	C	24
MSLOOP	EQU \$	A	0
	NOP	L	12
	NOP	L	12
	DJNZ B,MSLOOP	L	24
	RET	C	24
MSDLAY	EQU 227	A	0
	END	A	0

Frequency codes:

A—Assembler only, not executed

C—Executed once per 1-ms count

O—Executed only once per entry to subroutine

L—Executed in each loop of the 1-ms routine

Timing Calculations:

1MS=11059.2 clock periods (clock is 11059200 Hz)

$$11059.2 = 24 + (24+24+24+24+24+24) + N(12+12+24)$$

$$11059.2 = 168 + (N \times 48)$$

$$(11059.2 - 168)/48 = N$$

$$N = 226.9 \text{ (round to 227)}$$

$$\text{Error for 1-ms pulse is } 0.1 \times 48 \times (1/11059200) = 0.000443 \text{ ms}$$

Cumulative error for 100-ms pulse is:

$$\frac{1105920 - 24 - 14400}{4800} = 227.395 \text{ (use 227)}$$

$$0.395 \times 48 \times (1/11059200) = 0.00171 \text{ ms}$$

Listing 1: The BASIC-52 code to initialize the EPROM data table.

```
14170 REM
14180 REM INITIALIZE EPROM DATA TABLE
14190 REM
14200 FOR X=0 TO A:READ E(X):NEXT X
14210 REM TYPE SIZE PWR 7XPA 7XPB 7XPC 7PCP 7PBP PLV NPL FP FXP FACTR
14220 DATA 2716,008H,026,0BBH,061H,001H,008H,000H,001,050,000,000,000
14230 DATA 2732,010H,026,0BFH,051H,001H,008H,000H,000,050,000,000,000
14240 DATA 2732,010H,026,0BFH,051H,001H,004H,000H,000,050,000,000,000
14250 DATA 2764,020H,028,0BFH,014H,001H,000H,000H,000,050,001,025,003
14255 DATA 2764,020H,028,0BBH,012H,001H,000H,000H,000,050,001,025,003
14260 DATA 27128,040H,028,0BBH,014H,001H,000H,000H,000,050,001,015,004
14265 DATA 27128,040H,028,0BBH,012H,001H,000H,000H,000,050,001,015,004
14270 DATA 27256,080H,028,0BBH,052H,001H,000H,000H,000,050,001,025,003
14280 DATA 27512,0100H,028,0BBH,051H,002H,000H,000H,000,050,001,025,003
14281 REM SET UP EPROM NAME TABLE
14283 $(1)="2716 25V "
14284 $(2)="2732 25V "
14285 $(3)="2732A 21V "
14286 $(4)="2764 21V "
14287 $(5)="2764A 12.5V"
14288 $(6)="27128 21V "
14289 $(7)="27128A 12.5V"
14290 $(8)="27256 12.5V"
14291 $(9)="27512 12.5V"
14295 RETURN
```

there is indeed a printed circuit board and kit for this programmer. For those of you with the components at hand and a desire for wire-wrappers' cramp, the finished code for the programmer is available on a 27128 EPROM or is downloadable from the Circuit Cellar BBS and BYTENet Listings. The phone number for BYTENet Listings is (617) 861-9764. This file contains the 12K bytes of executable code that should be put into a 27128 EPROM and installed in IC11.

Special thanks to Bill Curlew for his software expertise.

There is an on-line Circuit Cellar bulletin board system that supports past and present projects. You are invited to call and exchange ideas and comments with other Circuit Cellar supporters. The 300/1200/2400-bps BBS is on-line 24 hours a day at (203) 871-1988.

Editor's Note: Steve often refers to previous Circuit Cellar articles. Most of these

past articles are available in book form from BYTE Books, McGraw-Hill Publishing Company, P.O. Box 400, Hightstown, NJ 08250, (1-800-2-MCGRAW).

Ciarci's Circuit Cellar, Volume I covers articles in BYTE from September 1977 through November 1978. *Volume II* covers December 1978 through June 1980. *Volume III* covers July 1980 through December 1981. *Volume IV* covers January 1982 through June 1983. *Volume V* covers July 1983 through December 1984.

The following items are available from

Circuit Cellar, Inc.
4 Park St., Suite 12
Vernon, CT 06066
(203) 875-2751

1. Serial EPROM programmer experimenter's kit. Includes PC board, 8052AH-BASIC chip, 11.05-MHz crystal, operating system software on preprogrammed 27128 EPROM, manual, and detailed parts list. \$89
2. Complete serial EPROM programmer kit. Includes all board-mounted components, programmed 27128 EPROM, 8052AH microprocessor, and manual. Less case and power

- supply \$199
3. Preprogrammed 27128 EPROM containing serial EPROM programmer system software \$22

The serial EPROM programmer is currently available only in kit form. It is available assembled and tested only in volume OEM quantities (telex: 643331). Price and delivery information available on request.

All payments should be made in U.S. dollars by check, money order, MasterCard, or Visa. Surface delivery (U.S. and Canada only): add \$5 for U.S., \$10 for Canada. For delivery to Europe via U.S. airmail, add \$20. Three-day air freight delivery: add \$8 for U.S. (UPS Blue), \$25 for Canada (Purolator overnight), \$45 for Europe (Federal Express), or \$60 (Federal Express) for Asia and elsewhere in the world. Shipping costs are the same for one or two units. Connecticut residents **please include 8 percent sales tax.**

Most of the individual components are available from JDR Microdevices, 1224 South Bascom Ave., San Jose, CA 95128, (800) 538-5000.