Radio-Electronics

SUPERCONDUCTIVITY: THE LATEST NEWS

$2.25 FEB. 1988
IN CANADA $2.75

TECHNOLOGY - VIDEO - STEREO - COMPUTERS - SERVICE

BUILD AN AUDIO/VIDEO SWITCHER

PUT YOUR A/V SIGNALS ON THE RIGHT TRACK

MAKE YOUR OWN PC BOARDS

NEW STEP BY STEP GUIDE EXPLAINS HOW IT'S DONE

THE FUTURE OF SHORTWAVE RADIO

NEWS FROM THE 1987 HIGH-FREQUENCY BROADCAST CONFERENCE

PROGRAMMABLE LOGIC DEVICES

AN ATTRACTIVE ALTERNATIVE TO STANDARD TTL PARTS

COMPUTERDIGEST

DESKTOP PUBLISHING SYSTEMS FROM SIMPLE TO SUPER

BUILD THE ULTIMATE CONTROL SYSTEM

THE REACTS 7000 ADVANCED MODULAR CONTROL SYSTEM

A GERNSBACK PUBLICATION

Radio-Electronics®

ADVANCED CONTROL SYSTEM

A complete, sophisticated control computer that is capable of operating almost every appliance or system in your home, and more!

H. EDWARD ROBERTS, M.D.

THIS IS THE FIRST IN A SERIES OF ARTICLES IN WHICH WE WILL introduce REACTS, the *R*adio-*E*lectronics *A*dvanced *C*on*T*rol *S*ystem. Over the next few months, we will build a control/robotics computer called the REACTS 7000, which is based upon the DataBlocks, Inc. *Altair II* system, a complete line of modular control elements currently available for personal and industrial-control computers. We'll also build all the peripherals the computer will need in order to perform a wide variety of control functions. In addition, we will show you how to use your computer in *real* applications.

Control computers

Control computers differ from conventional computers in several important ways. One of those differences, the ability to easily interface with a huge variety of external devices, gives the control computer the potential, over the next few years, to revolutionize our lives in more ways than anything yet produced by mankind.

In a typical home, there are applications for dozens of control computers. Some simple applications include controlling appliances; adjusting the heating, air conditioning, and/or humidity; minimizing power consumption; and ultimately running a robot lawnmower or vacuum cleaner so you won't have to.

REACTS 7000 is designed to be completely modular, with each module containing the circuitry to perform one or more complete functions. It also uses conventional programming languages (BASIC, C , assembly, etc.). Each subunit or module in the computer plugs into every other module. For example, the first module we will build is the central processor/computer module. That module is a complete stand alone system that includes its own memory, serial port, disk, vectored interrupt, real-time clock, system clock, memory-expansion hardware and all necessary buffers. In other words, it is a complete personal computer. Following that, we will build modules that contain semiconductor disk systems along with integral PROM programmers, complete CRT terminals, A/D converter modules, stepping-motor modules, etc.

The goal of this series of articles is to make *you* the system designer. You do the designing by simply selecting what modules you need for your application. Those modules are then stacked together in any order to create a custom control computer for any application. You need only to pick a language and write the program. Surprisingly, most control applications require simple programs.

Central to REACTS is the soft-hardware concept. Soft-hardware is simply hardware that can be changed or altered as easily as software. The author first developed the concept at MITS while designing the Altair computer, but only recently has it been possible to exploit it in a practical way. Over the next few months you will see how we achieve soft-hardware using straightforward, well-established principles. Indeed, the main requirement for soft-hardware is to make sure that each module's operation doesn't interfere with the operation of any other module. Additionally, each module must be self-sufficient mechanically and electrically, Further, each module should provide shielding to meet FCC requirements, and each module must provide its own motherboard.

Sources

The following items are available from DataBlocks, Inc., 579 Snowhill Road, Glenwood, GA 30428. Or call (800) 652-1336; in Georgia call (912) 568-7101: DP-CPU—design package of schematics and instructions, $10.00; PC-CPU—PC board for CPU module (includes DP-CPU design package,$37.00; PC-CLK—PC Board for clock, $18.00; SYS-PROM—the REACTS operating system (enhanced SB-80) installed on a 32K UV-erasable PROM (includes operating system documentation), $44.00; REC-CPU—complete kit of parts, PC boards, IC's, connectors, for CPU module (does not include clock or system PROM), $147.00; REC-CLK—complete clock subsystem including all parts, PC boards, NiCd battery, and connectors, $43.00; and REC-SYS—All of the above, $218.00. An Elpac power supply is also available for $49.00. Please add $10.00 postage and handling per order. GA residents must add appropriate sales tax.

CPU module

The REACTS CPU module consists of a microprocessor, 64K static RAM, 32K EPROM disk system, serial I/O port, vectored interrupt system, completely buffered bus drivers, crystal-controlled clock, 1-megabyte memory-expansion subsystem, and internal sense switches. If you aren't familiar with any of those terms, don't give up! Stick with us as we are going to discuss each item individually and will explore the engineering philosophy used in each decision.

Cost and reliability are important considerations with any computer system, but to a large extent they determine where and when it is practical to use a control computer. For instance, it is not practical to use a $20,000 dedicated computer to provide security and environmental control for your $100,000 home. On the other hand, if the same job could be done by a computer system that sold for $5, nobody could afford to be without one. Similarly, a system that would automatically drive your car to work but was only 99.5% reliable would not be very interesting.

Our goal is to design a machine that is affordably priced and as close as possible to being 100% reliable. That is accomplished by eliminating mechanical subsystems and using special connectors. Cost is controlled, to a large extent, by the soft-hardware concept; you only include what is required for the particular application.

There are some problems along the way with developing an "ultimate" control computer. First, most of us don't have a $250,000 microprocessor-development lab. We have to be able to develop the hardware and software we need on the target system (a target system is the final product of a development project). But, that requires the development of both hardware and software that may only be needed for development and then scrapped when finished.

Fortunately, the soft-hardware approach allows us to use our target system as the development system. We do that by including the development modules (CRT controllers, PROM burners, etc.) during the development stage and then removing the unneeded development modules when finally we install the target system. That approach is especially economical for those who will build multiple systems.

Most board-level computers are designed to work with custom software because that is the easiest to implement. The approach we will take is to use a disk operating system even on the most minimal system. That allows us to use a disk-operating system with all its inherent power. There are a number of disk operating systems from which to choose, but a CP/M-like system is familiar to both MS-DOS users as well as CP/M users. For that reason, we are using the DataBlocks disk-operating system which is a superset of CP/M; indeed, all standard CP/M software will be compatible with our system.

Included in the operating system are all the utilities needed for development, such as drivers for the PROM programmer, drivers for printers, debug routines, etc. Those special utilities can be left out of the final target system if desired, to minimize system size and cost.

Design principles

The following discussion will give you an idea of the design concepts used in engineering REACTS. It will also give you a feel for the tradeoffs made in the design of the system.

The microprocessor used in the CPU module is a version of the Z-80. That is probably the most popular and widely used microprocessor ever made. It has become the standard in the control industry, and more personal computers have been built using it than any other microprocessor. It isn't the fastest of the microprocessors—indeed some of the newer high-performance microprocessor systems will out-perform it by a factor of 20 to 1. But of the modern microprocessors it is the easiest to understand, and it does not have any unpredictable or "funny" interfacing quirks. That is especially important to the non-expert designer. As we proceed, you will see how we get around the processor-speed problem by using the principle of distributed processing; that is the use of multiple processors in a single system. The multiprocessor concept is based on the assumption that computers are free. It turns out that the principle is reasonably valid in practice, since the CMOS Z-80 is less expensive than a number of other IC's in REACTS. We will also use a number of other techniques that will greatly expand the power and capability of the system.

REACTS CPU

To get an idea of how fast the REACTS CPU module is in terms of control functions, let's look at a simple example. Assume we needed to turn on or off 2000 switches in a predefined manner. The high-speed version of REACTS would be capable of doing that at a rate of 500 times a second, or approximately 1-million switch operations per second.

The microprocessor used is CMOS. Indeed, all components in the module are CMOS. That increases cost, but it makes battery power, either emergency or continuous, easy. In addition, the noise immunity is improved, and less heat is generated so no fans are needed. Further, less-expensive power supplies can be used and the system can be used in relatively confined quarters without the problem of overheating.

The minimum system memory is 64K of static CMOS RAM, but the system will address a total of 1 megabyte. Static RAM adds to the cost of the system, but it does allow for easy battery powering and makes the system more predictable in a multiprocessor and/or control environment. The addressing scheme is straightforward with the exception of the expanded memory. The expanded memory is based on a paging scheme that allows the computer to switch pages of memory 32K bytes at a time. We will discuss some specific uses of that memory in future articles. It is interesting to note that the page-switching system allows the system to operate at effective direct-memory-access speeds of 10 gigabytes per second. As a comparison, if we could read a 40 megabyte disk that fast, it would only take 4 milliseconds to read the whole disk!

A conventional RS-232 port is included in the basic CPU module. That port can be used to connect to a terminal, modem, or any other standard RS-232 device. The baud rate and signal characteristics are under software control and can be modified from within the program.

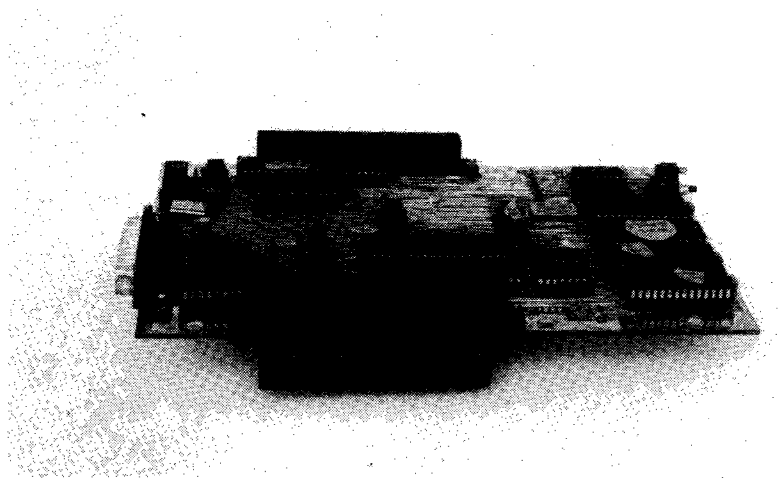


FIG. 1—EACH MODULE'S MOTHERBOARD interfaces to the REACTS system bus via feed-through Molex connectors.

The CPU module supports 9-levels of interrupts: one *N*on-*M*askable *I*nterrupt, or NMI, and 8 vectored levels. An interrupt-driven system is especially useful in process control.

There is somewhat of an aura of intimidation associated with interrupts. But actually, they make programming simpler and much faster in many applications. To use an interrupt, you simply pull the line low. That halts the program that is currently running and causes a jump to a special subprogram in memory; its much like a GOSUB command in BASIC. The computer executes the interrupt program and then returns to the original program. Interrupts provide two advantages to the system designer. The first is that the interrupting device can be serviced at random; that is, the program doesn't have to keep checking to see if a service is needed. Second, the interrupt can be serviced instantly; it doesn't have to wait for the main program. Vectoring simply means that the interrupts have levels of priority and that a high-priority interrupt can interrupt a lower priority one. Indeed, it is possible to have a number of interrupts waiting for service in a busy system. In later articles, we will see some detailed examples of practical uses of the interrupt system.

The basic CPU module includes sense switches. Those are simply switches that can be set and read by the program. They are actually a "poor man's" keyboard. An example of where they are useful is in the development vs. target system. In those systems, the computer checks to see the setting of the sense switch on power up, and from that determines whether it should look for a terminal or start executing a program. In a real target system, the sense switches would be used to select which program is loaded at auto start-up.

REACTS disk

In order to meet our design criteria, even the most minimum system will contain a disk. That is achieved in our system by using a UV-erasable PROM disk. That disk is seen as a disk by both the external hardware and software. It has all the attributes of a write-protected magnetic disk, save one: it operates at blinding speeds. In later articles, we will build larger semiconductor disk systems as well as a PROM programmer that allows you to burn your own PROM disk system for the computer.

REACTS makes extensive use of RAM and PROM disks. Semiconductor disks are significantly more reliable than conventional magnetic disk drives since there are no moving parts. They also consume much less power and are smaller. If large amounts of data need to be stored, then a magnetic disk becomes more attractive. In later articles, we will build a miniature floppy disk that's appropriate for mass storage and that will be compatible with our system. The overwhelming advantage of the semiconductor disk is speed.

The random-access time of a modern high-speed hard-disk system is approximately 25 milliseconds. The random-access time of our disk is approximately 10 microseconds, or approximately 2500 times faster. That is one of the reasons why our system is capable of outperforming some of the bigger and more expensive pro-

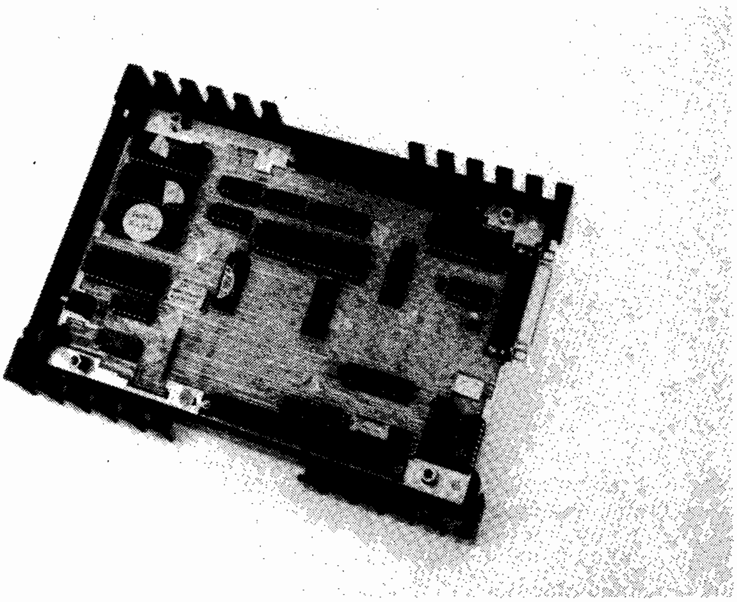
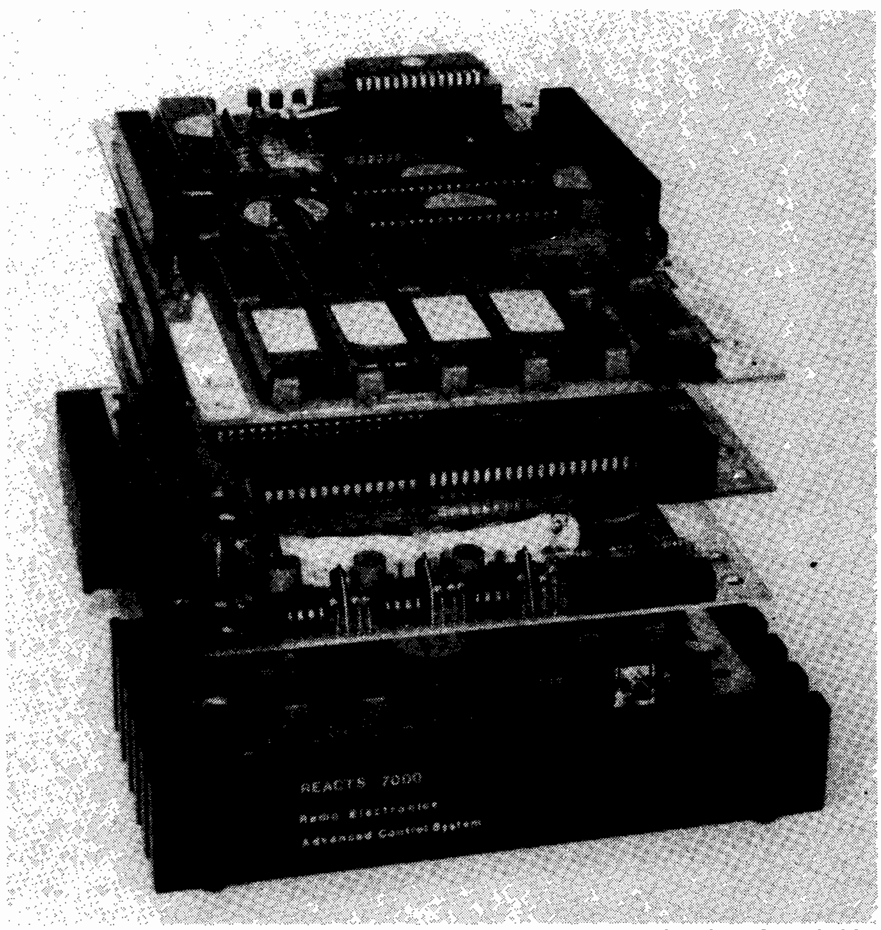


FIG. 2—THE EXTRUDED ALUMINUM CASE is designed to comply with FCC shielding requirements.



THE REACTS 7000 control/robotics computer system consists of series of stackable modules. Here, a four-module system consisting of the CPU, a power supply, a CRT/ printer interface and a PROM programmer is shown. The modules are shown out of their shielded cases.

cessors. Finally, semiconductor disk systems are much easier to understand and much easier to use when designing custom software.

In order to maintain the soft-hardware concept, the REACTS bus is driven by CMOS drivers. No more than 1 or 2 CMOS loads are ever placed on the bus by any one module. That buffer system also allows for disconnecting the processor from the bus. That is a necessary condition in multiprocessor schemes. Buffers are not the most exciting topic, but their proper use is critical in a multi-processor, soft-hardware system so they will be addressed as we procede.

The standard module card is 8 inches by 5.3 inches and connects to the 120-line system bus using special feed-through Molex connectors. See Fig. 1. Those two connectors allow the modules to be stacked together in any order. The bus that we are using is the Altair-II bus developed by DataBlocks. Our system is designed to be fully compatible with all the existing DataBlocks modules and software. At the present time, there are literally dozens of different modules available that use the DataBlocks Altair-II bus.

REACTS case

Each module can be provided with its own shielded case that meets FCC standards. If you desire, you can mount the finished, unenclosed assembly inside a conductive case, which also meets FCC requirements. Figure 2 shows a module in its extruded aluminum case; note that the rear panel has been removed for clarity. While that case is unique to REACTS, it is compatible with the standard Altair-II system or equivalents.

A word about FCC standards is appropriate at this time. Each builder is responsible to make sure he meets the FCC requirements. All modules in this series of articles are tested to the most stringent FCC requirements using self-contained cases. Nevertheless, it is your responsibility to verify that your system doesn't interfere with any other service.

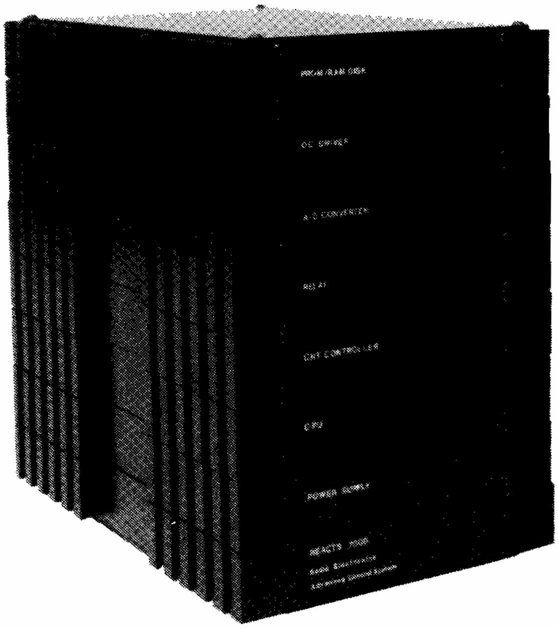
The real-time clock is plugged into the CPU-module board. That is done to keep cost at a minimum; the clock need only be installed if an application requires it. An on-board NiCd battery is used to provide backup power for the clock. The clock provides time, date, month, year, and can be used to generate interrupts to the main system. We will make extensive use of the clock in some of our future articles.

That's all we have room for this time. When we next meet we continue our look at REACTS and show you how to build the first of our modules, the CPU. **R-E**

Build REACTS:

THE RADIO-ELECTRONICS ADVANCED CONTROL SYSTEM

This month we take an in-depth look at the CPU board's circuitry.



H. EDWARD ROBERTS M.D.

Part 2 AS WE SAID last time, the REACTS 7000 uses an operating system that is compatible with the CP/M operating system. The version we use is an enhanced SB-80 from Lifeboat Associates. Special utilities have been added to the basic DOS (*D*isk *O*perating *S*ystem) in order to make robotics/control applications easy to implement.

A number of the systems commands are known as *intrinsic*—i.e. they are part of the basic system. Examples of intrinsic commands are: DIR, which lists the directory of a disk drive; DEL, which deletes files; SAVE, which saves files; and BATCH, which can be used to chain together a number of different commands. Of course, there are many more, but that gives you an idea of some of the commands if you aren't familiar with CP/M or MS-DOS.

Additionally, the operating system includes a number of utilities such as CONFIG, RBURN, and FORMAT, etc. Utilities are programs that have been added to the basic operating system. The following is an explanation of some of the utilities included with the REACTS DOS:

- CONFIG: Used to change different parameters of the DOS such as, disk-drive assignments, boot drive, serial-port parameters, I/O assignments, etc. If the changes are to be made permanent, a new boot PROM can be burned using RBURN described below.
- RBURN: is a utility that allows you to create a custom PROM disk on a 32K ROM. Using that utility, you can create a PROM disk as easily as you create a floppy disk.
- FORMAT: That allows the formatting of the REACTS 3.5-inch floppy disk if a disk controller and disk are included in the system.

That gives you some idea of what can be done with the CP/M operating system. There are a number of excellent books on CP/M for those who want to get more information. In addition, a detailed manual is included with the operating system sold by the source mentioned elsewhere in this article.

It is important to note that you don't have to have an operating sys-

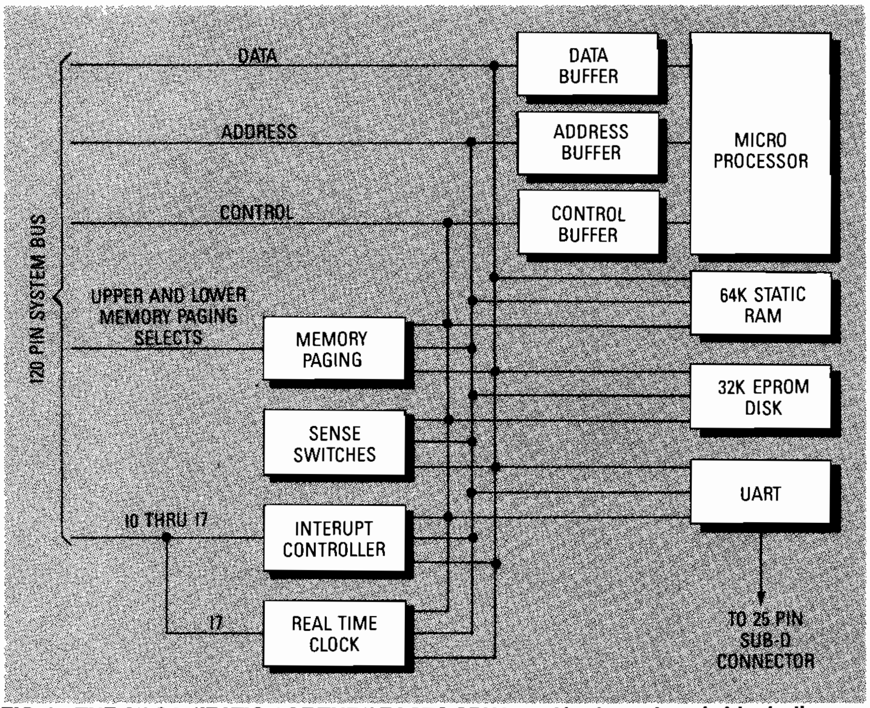


FIG. 1—THE ORGANIZATION OF THE REACTS CPU board is shown here in block-diagram form.

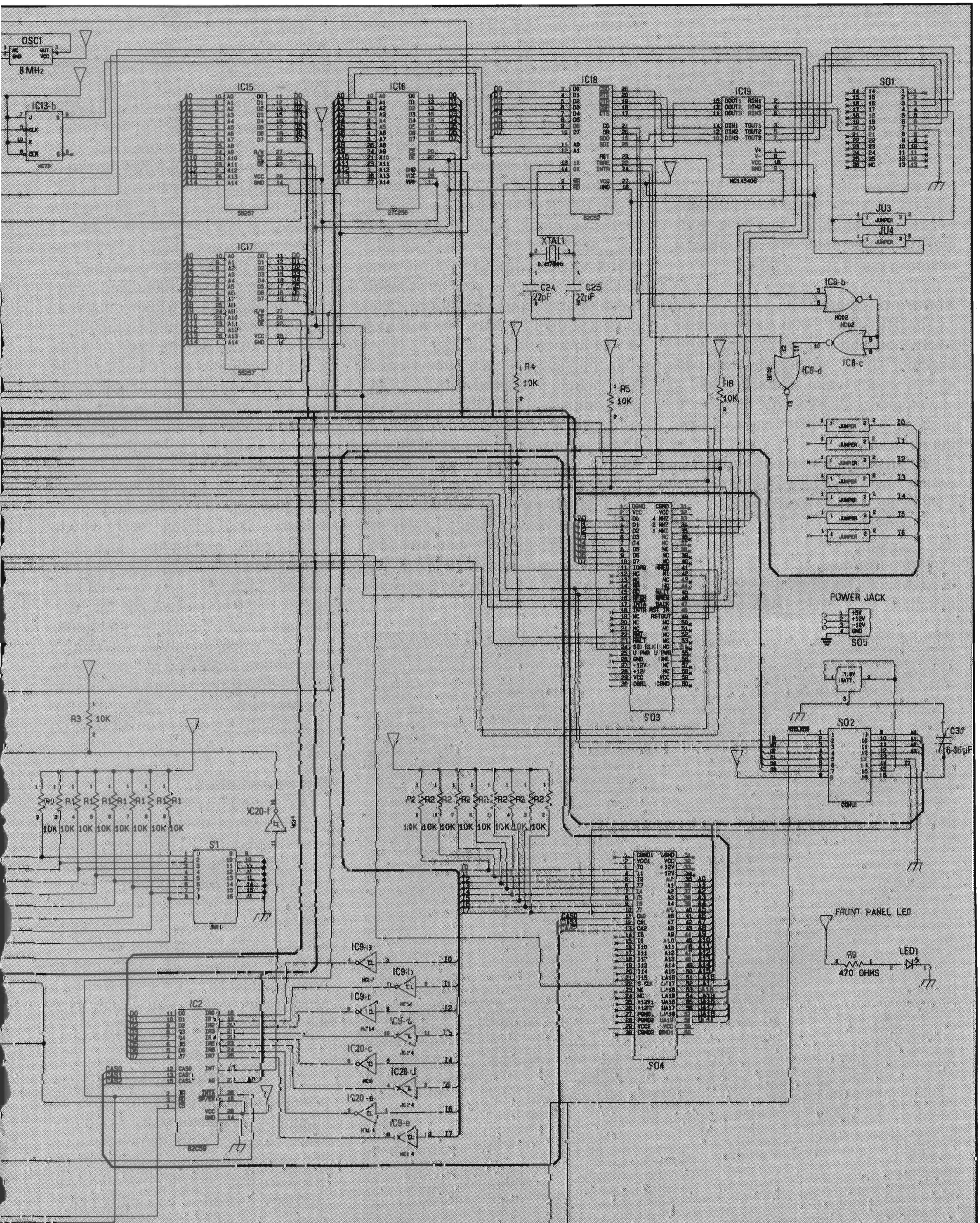

OSC1
8 MHz
IC13-b
IC15
58257
IC16
27C256
IC17
58257
IC18
82C52
IC19
MC145406
SO1
JU3
JU4
XTAL1
C24
22pF
C25
22pF
IC8-b
IC8-c
IC8-d
R4
10K
R5
10K
R8
10K
POWER JACK
SO5
SO3
SO2
C30
6-35pF
R3
10K
IC20-f
S1
SO4
FRONT PANEL LED
R9
470 OHMS
LED1
IC2
82C59

TABLE 1—I/O PORTS	
Subsystem	**I/O Port(s) Used**
Sense Switches	187
Real Time Clock	192 thru 207
Memory Paging	210
Interrupt Controller	216 thru 217
UART	212 thru 215

tem to make the system function. If you are familiar with assembly-language programming and have your own cross assembler, you can create a custom system from scratch.

Theory of operation

The REACTS 7000 CPU is relatively conventional in organization. Figure 1 shows an overall block diagram of the system. The schematic is shown in Fig. 2. Note that because of its large size, the print had to be reduced in scale.

As already discussed, the processor used in the computer is a CMOS version of the Z80 (IC1). An on-board crystal oscillator provides the necessary clock signals for the system. Jumpers JU1 and JU2 are used to select between an 8-MHz (JU1 installed) or 4-MHz (JU2 installed) clock frequency. Other major subsystems on the central-processor module includes a set of eight sense switches, an interrupt controller, a *U*niversal *A*synchronous *R*eceiver/*T*ransmitter (UART), two 32K static RAM chips, a 32K boot PROM-disk, bus buffers, and the memory-paging capability that allows the addressing of up to one megabyte of system RAM. Also, an optional real-time clock is plugged in as a daughter board on the CPU module. All of those components must communicate with the Z80 processor; most of that communications takes place via the I/O ports. We will look at that in more detail later.

In our system each subsystem is hard-wired to a predetermined I/O port address; the addresses for the subsystems are shown in Table 1. Those addresses are known to the operating system, and communication occurs automatically in most cases. If required, an application program can also bypass the operating system and communicate directly with any subsystem. We will talk more about how to bypass the operating system in future articles.

TABLE 2—BUS DEFINITIONS

Left Bus				Right Bus			
Pin No.	**Function**	**Pin No.**	**Function**	**Pin No.**	**Function**	**Pin No.**	**Function**
1	CASE GND	31	DGND	1	DGND	31	CASE GND
2	VCC	32	VCC	2	VCC	32	VCC
3	INT 0	33	+12V	3	DO	33	4 MHZ
4	INT 1	34	−12V	4	D1	34	2 MHZ
5	INT 2	35	A0	5	D2	35	1 MHZ
6	INT 3	36	A1	6	D3	36	RESERVED
7	INT 4	37	A2	7	D4	37	RESERVED
8	INT 5	38	A3	8	D5	38	RESERVED
9	INT 6	39	A4	9	D6	39	RESERVED
10	INT 7	40	A5	10	D7	40	RESERVED
11	CA 0	41	A6	11	IO/M	41	RESERVED
12	CA 1	42	A7	12	RESERVED	42	RESERVED
13	CA 2	43	A8	13	RESERVED	43	RESERVED
14	INT 8	44	A9	14	$\overline{\text{WR}}$	44	RESERVED
15	INT 9	45	A10	15	$\overline{\text{RD}}$	45	$\overline{\text{WAIT}}$
16	INT 10	46	A11	16	$\overline{\text{RFSH}}$	46	BREQ
17	INT 11	47	A12	17	$\overline{\text{INTA}}$	47	BACK
18	INT 12	48	A13	18	$\overline{\text{INTR}}$	48	$\overline{\text{RESET IN}}$
19	INT 13	49	A14	19	RESERVED	49	RESET OUT
20	INT 14	50	A15	20	RESERVED	50	RESERVED
21	INT 15	51	LA16	21	RESERVED	51	RESERVED
22	RESERVED	52	LA17	22	$\overline{\text{NMI}}$	52	RESERVED
23	RESERVED	53	LA18	23	$\overline{\text{HALT}}$	53	RESERVED
24	RESERVED	54	LA19	24	SIO (CLK)	54	RESERVED
25	+12V (PWR)	55	UA16	25	UNREG PWR	55	RESERVED
26	+12V (PWR)	56	UA17	26	GND (±12)	56	GND (±12)
27	GND (PWR)	57	UA19	27	−12V	57	RESERVED
28	GND (PWR)	58	UA19	28	+12V	58	RESERVED
29	VCC	59	VCC	29	VCC	59	VCC
30	CASE GND	60	DGND	30	DGND	60	CASE GND

The microprocessor

The Z80 processor is inherently capable of addressing up to 64K bytes of memory. Additionally, the Z80 can address up to 256 eight-bit I/O ports. The following is a discussion of how I/O operations take place in the system.

Whenever the computer executes an I/O command, it notifies the rest of the system by raising the system bus' $\overline{\text{MREQ}}$ line high. (Pin definitions for the system bus are given in Table 2). When that occurs, the lower 8 address lines contain the address of the I/O device. The $\overline{\text{RD}}$ (read) and $\overline{\text{WR}}$ (write) lines tell the system whether that is an input or output; i.e. if the read line is low the system is requesting an input, if the write line is low, the data on the bus is an output. Incidentally, the memory-read or -write system operates in exactly the same manner, except the $\overline{\text{MREQ}}$ line is low for memory operations.

In summary, in order for a device that uses I/O-port addressing to be selected, the $\overline{\text{MREQ}}$ line must be high. Additionally, it must have an address that matches the address on the lower address lines (A0–A7), and the IORQ signal must be correct for that particular channel. If all those conditions are met when inputting, the correct data will be placed on the data bus by the inputting device and then sent to the processor. For an output, the processor will place the correct data on the bus.

Sense switches

As stated in Part 1, the sense switches can be thought of as a "poorman's keyboard." They allow you to input data to the system without connecting any other device. That's useful in a minimal system where size, space, and/or cost prevent using a more sophisticated input device.

Those switches are hard-wired to input channel 187. If you are writing a program in BASIC you simply type the following line:

```
120 S = INPUT 187
```

That would be read by the interpreter to mean input the data on channel 187 and store it as variable S. Since there are eight switches, up to 256 unique instructions could be supplied to the computer with those switches. Indeed, if you really like to work, you could input all of the standard ASCII codes using only the sense switches.

In addition, the four rightmost switches are used by the REACTS' disk-operating system at power-up. The setting of those switches select the console device, the serial port's baud rate, and which disk drive the system should boot from. A common use of the sense switch would be to automatically load and execute a program at power up.

Interrupt controller

The Z80 processor has two interrupt inputs—a nonmaskable and a maskable interrupt. Indeed, one of the reasons that the Z80 has become so popular as a controller is the power of its interrupt system.

A CMOS version of the industry standard 8259 interrupt controller (IC2) was added to the REACTS CPU. That provides us with eight additional vectored (i.e. prioritized) interrupt inputs. Those interrupts are brought out to the system bus and can be used by any of the other modules. Complete control of the interrupt system can be made under software control. Communication with the interrupt subsystem is made through I/O ports 216 and 217. Future articles will show you how to exploit interrupts. For now, let's just say that the interrupt system adds real power to the REACTS computer. In addition, many programming requirements can be simplified by using interrupts.

The UART

A standard RS-232 serial port is included on the REACTS CPU board. The serial port's communication parameters, baud rate, number of stop bits, word length, parity, etc., are configured by the Z80 through an 8252 UART (IC18). The 8252 also converts the parallel data from the system into a serial data train for use with serial communications systems. The CTS (*C*lear *T*o *S*end) line is configured—always high (for send-only or full-duplex applications) or floating—via jumpers JU3 and JU4. The serial port can be used to interface the computer module to a terminal, modem, or any other device that requires serial data input/output. The operating system communicates with the 8252 UART through I/O ports 212, 213, 214, and 215. The CRT controller we will build later on does not use that port, which means that it is normally available for any other purpose. But if we use an external terminal or computer as the controlling device, that port is used as the communication port.

Sources

The following items are available from DataBlocks, Inc., 579 Snowhill Road, Glenwood, GA 30428. Or call (800) 652-1336; in Georgia call (912) 568-7101: DP-CPU—design package of schematics and instructions, $10.00; PC-CPU—PC board for CPU module (includes DP-CPU design package,$37.00; PC-CLK—PC Board for clock, $18.00; SYS-PROM—the REACTS operating system (enhanced SB-80) installed on a 32K UV-erasable PROM (includes operating system documentation), $44.00; REC-CPU—complete kit of parts, PC boards, IC's, connectors, for CPU module (does not include clock or system PROM), $147.00; REC-CLK—complete clock subsystem including all parts, PC boards, NiCd battery, and connectors, $43.00; and REC-SYS—All of the above, $218.00. An Elpac power supply is also available for $49.00. Please add $10.00 postage and handling per order. GA residents must add appropriate sales tax.

Memory paging

System memory paging is performed via I/O port 210. The four least-significant data bits select the lower 32K page while the four most-significant data bits select the upper 32K page. That allows us to literally change the contents of the system memory in about 10 microseconds. That is equivalent to a DMA (*D*irect *M*emory *A*cess) transfer rate of approximately 10 gigabytes-per-second, which is fast by any standard, or of being able to move the entire contents of a 40-megabyte hard disk in 4 milliseconds. The basic REACTS system will support more than 1 megabyte of additional memory.

The memory-paging system is implemented using three custom PAL (*P*rogrammable *A*ray *L*ogic) IC's (IC10–IC12). Those can be purchased from the supplier mentioned in the Sources box.

Static RAM

Two 32K static CMOS RAM IC's (IC15 and IC17) provide 64K of system memory for the computer, and that is just the basic memory on the CPU module. Static memory IC's are somewhat expensive, but their use makes battery powering easy. That's because static RAM doesn't have any of the traps associated with multiple processor systems—we don't have to make sure that the memory is in any special configuration to power down or to perform DMA's.

System disk

As we discussed earlier, even the most minimal REACTS system includes a disk operating system. That is because the CP/M P drive is a 32K UV-erasable PROM that is included on the CPU module (IC16). That PROM is set up to allow for the BIOS (*B*asic *I*nput/*O*utput *S*ystem software) as well as drive P to be installed. There is enough storage available on that prom disk to store the BASIC interpreter and a decent sized user program. That means that for some applications, no other system hardware, such as a disk, needs to be added to the basic CPU module; only your interface hardware is required.

In a future article we will provide the construction plans for a PROM programmer as well as for a large RAM/PROM disk system.

Real-time clock

In many applications a real-time clock will be desirable and perhaps even mandatory. The real-time clock used by REACTS can be programmed to operate in a twelve or twenty-four hour time-keeping format. It can provide single or repeated interrupts to the processor with a resolution of 0.1 second. It includes an automatic calendar function (that accounts for leap years), and it includes a NiCd battery for back up. I/O ports 192 through 207 are used to configure and set the clock. The interrupt output of the real-time clock is connected to system interrupt 7.

The clock interfaces with the main CPU board via a 16 pin connector. (Note that the rest of the clock circuit will be shown in a future installment of this series.) The use of an interface circuit makes the clock optional and keeps the sytem cost to a minimum in systems where the clock is not needed.

Next time

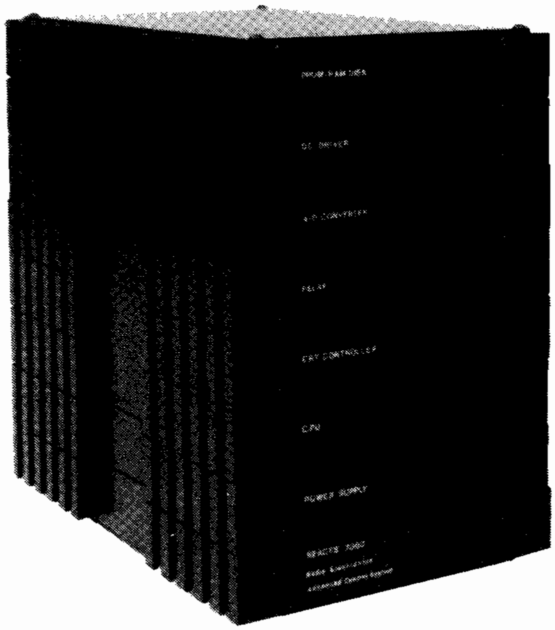
That's all we have room for this time. Next time we will get our hands dirty and show you how to build the CPU module. **R-E**

Build REACTS:

THE RADIO-ELECTRONICS ADVANCED CONTROL SYSTEM

This month we get to work and build the CPU module.

H. EDWARD ROBERTS, M.D.



Part 3 LAST TIME we looked at the theory behind the REACTS CPU module. Now, its time to put our theory to practice and build the CPU module.

Construction

Construction is straightforward. The PC board, which is shown in PC Service, is double-sided; a plated-through board with silk screen and solder mask applied is available from the supplier listed in the Sources box.

Components are mounted on the board following the parts-placement diagram shown in Fig. 1. Figure 2 is a photograph of the author's prototype. Although there are some minor differences, your finished board will closely resemble that one.

Note that, as shown in Fig. 2, we used sockets for all IC's; despite the fact that only IC16, the EPROM disk, strictly requires the use of a socket, we strongly recommend that you too use them for all IC's.

Care should be taken in handling the CMOS parts since they are very sensitive to static-discharge damage. Grounding the workbench and soldering iron is highly desirable, especially if you don't use sockets. Placing the IC's, PC board, and tools on a single sheet of grounded aluminum foil will prevent static damage to any components.

You should be careful when installing the Molex connectors (SO3 and SO4). The connector comes in two pieces, one with pins and one without pins. To install the connector, the piece with the pins is mounted flush with the top (component) side of the board and the pins are soldered on the bottom of the board. When soldering the connector to the board, use a minimum amount of solder and make sure no solder bridges occur. Furthermore, don't "drag" solder up the pins. That's because those pins must slide down over the other half of the connector, and any excess solder could damage the friction fit. The connectors are designed to be assembled once only; it is a bad idea to push the two halves together and then to pull them apart. But if assembled following the preceding recommendations, they are extremely reliable and durable, and may be plugged and unplugged into other connectors literally hundreds of times.

Note, however, that when separating modules **it is important that the connectors be carefully pulled apart by pure vertical force. Prying the boards apart may cause serious damage to the connectors.** The connectors are the most expensive single item on the board, and they are time consuming to remove and replace if damaged. That doesn't mean that they are fragile, but with careless handling they can be broken.

Other than the connectors, installation of all other parts is straightforward, although care should be taken to be sure that all polarized components (IC's, diodes, electrolytic capacitors, etc.) are installed correctly. After all components are installed, a careful inspection for proper compo-

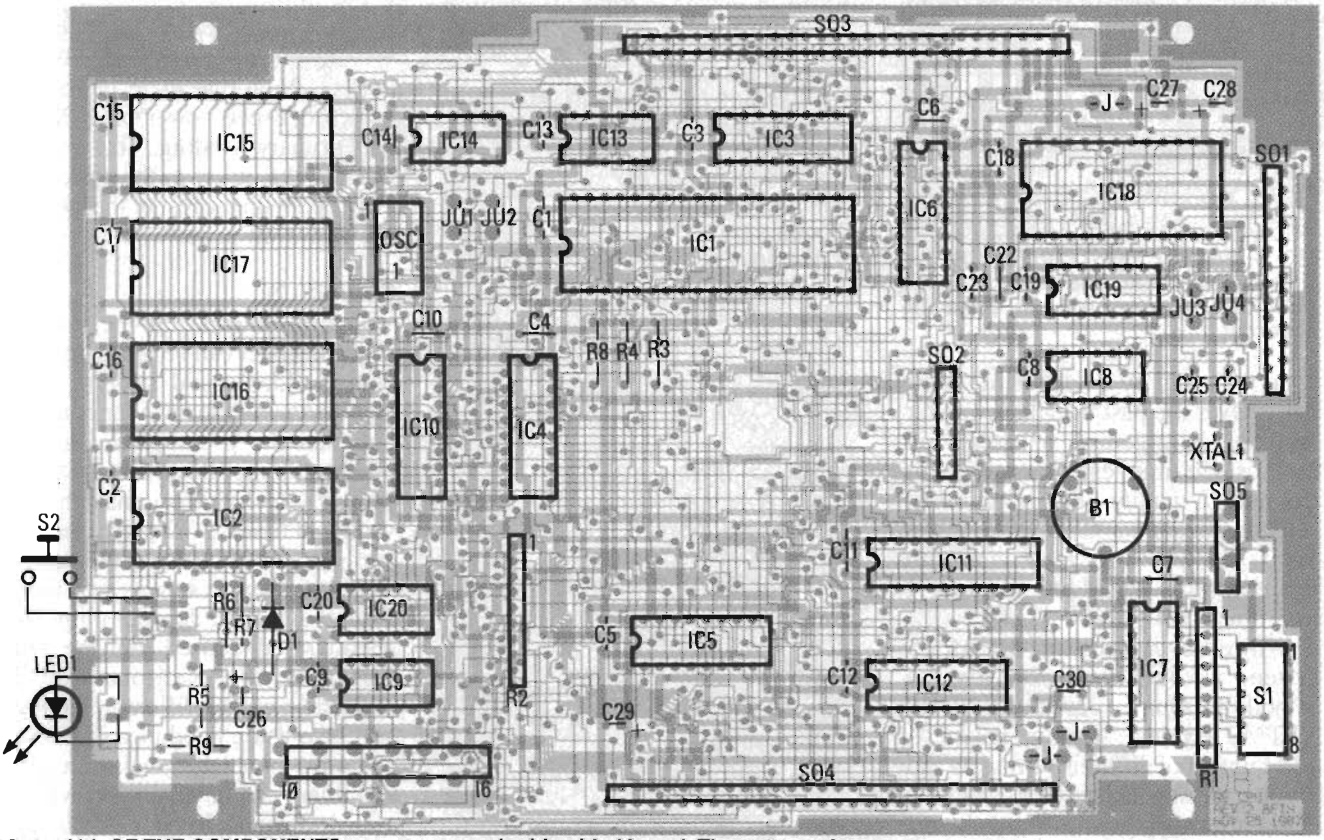


FIG. 1—ALL OF THE COMPONENTS mount on one double-sided board. The patterns for the board can be found in PC Service.

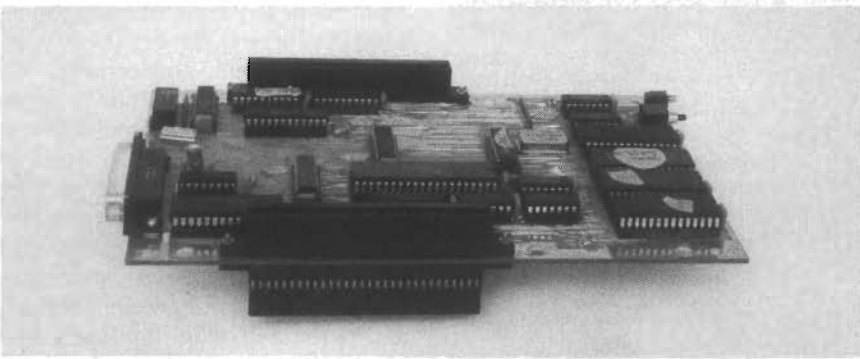


FIG. 2—THE COMPLETED AUTHOR'S PROTOTYPE. While the prototype differs slightly from the project described in the article, your board and this one will appear similar

TABLE 1

Parameter	Setting
EMULATION	QVT101C
PARITY	OFF
DATA BITS	8
STOP BIT	1
BIT8	0
FDX/HDX (Full or Half Duplex)	FDX
CHAR/LINE/BLCK	CHAR
ON LINE/LOCAL	ON LINE
BAUD RATE	9600

nent location should be made. Also, the board should be carefully searched for solder bridges or other construction defects; a magnifying glass will be useful for that.

The operating system

REACTS uses an enhanced version of Lifeboat Associates SP-80. SP-80 was designed to be an improved version of CP/M. We have added enhancements to the package in order to produce the REACTS DOS. If you are familiar with CP/M, MS-DOS, or SB-80, you will feel right at home with the operating system. REACTS DOS is available on a PROM from the vendor listed in the Sources box elsewhere in this article. Once the CPU module is assembled, that PROM is installed, a terminal is connected via the serial port, and power is applied, the CPU is ready for checkout and use.

Checkout

Power can be applied to the CPU via the power jack (SO5) using an Elpac power supply (available from the source listed in the Sources box found elsewhere in this article). Other supplies can also be used, as long as they provide the appropriate voltages and currents and are well regulated and current limited. The Elco power supply produces +5 volts DC at 860 mA, +12 volts DC at 300 mA, and −12 volts DC at 300 mA.

The checkout itself is actually quite simple. Just connect a standard terminal to the serial port and apply power to the system after setting switch S1 as shown in Fig. 3. If everything is working correctly, in less than a second (about 250 milliseconds to be more exact) you will see the opening screen. If that happens, you are ready to use your system to control the world!

Adding a terminal

Before we wrap up, let's spend a couple of moments discussing exactly

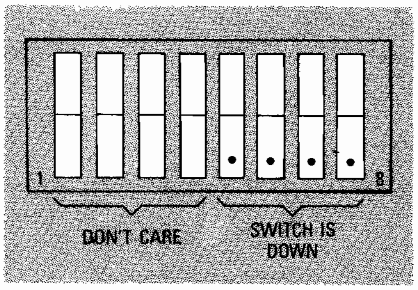


FIG. 3—SET THE EIGHT DIP SWITCHES of S1 as shown here. The settings of the don't-care switches will not affect the checkout.

what we mean by a "standard terminal." Over the past decade computer terminals have gotten "smarter" and at the same time less expensive. For example, a quality "dumb" terminal sells in single quantities for less than $400. Ten years ago, a terminal with less capability would have sold for 4 to 5 times that price.

While developing the REACTS system, we have made extensive use of both the Qume 101 and the ADDS VPT/A2 terminal. Also, we have tested a number of different terminals and all operate in a satisfactory manner.

Another alternative is to purchase a used terminal. At the present time there are a number of used terminals on the market (e.g. ADM-7, etc.) that can be purchased for next to nothing. Be sure that any terminal you consider is compatible with a standard RS-232 interface before buying. In addition, it is probably a good idea not to purchase any used terminal if its technical manual is not available.

The settings we use on the Qume 101 terminal are shown in Table 1. They will give you a starting place even if you use a different machine.

In addition, the REACTS system allows you to use two other devices for the terminal. One is to use an IBM PC or compatible as a terminal. If you have a computer and a modem, the software needed to convert your PC into a terminal is available on the RE-BBS (516-293-2283, 8 data bits, no parity, 1 stop bit). It is also available on disk from the supplier mentioned in the Sources box for $18.00 plus postage. That disk also includes a routine that allows you to use your hard or floppy disk as a storage device for the REACTS system.

Finally, you can choose to build your own terminal. The last alternative has several advantages; we will go into those in detail and provide construction plans for the REACTS terminal in a future installment. Note that the REACTS terminal also features a parallel printer port.

Next, we will show you how to build a battery back-up system. **R-E**

PARTS LIST

All resistors ¼-watt, 5%, unless otherwise noted

R1, R2—10,000 ohms, 9-into-1 resistor network, SIP package
R3–R6, R8—10,000 ohms
R7—10 ohms
R9—470 ohms

Capacitors

C1–C20, C22, C23—0.47 µF, 25 volts, ceramic disc
C21—not used
C24, C25—22 pF, 25 volts, ceramic disc
C26—10 µF, 16 volts, electrolytic
C27, C28—47 µF, 16 volts, electrolytic
C29—47 µF, 10 volts, electrolytic
C30—6–36 µF, trimmer

Semiconductors

IC1—Z80 microprocessor
IC2—82C59 programmable interrupt controller
IC3–IC7—74HC245 octal three-state transciever
IC8—74HC02 quad 2-input NOR gate
IC9, IC20—74HC14 hex inverting Schmitt trigger
IC10, IC12—16L8 custom PAL, see text
IC11—EP600 custom PAL, see text
IC13, IC14—74HC73 dual JK flip-flop with clear
IC15, IC17—55257 32K static RAM
IC16—27C256 32K UV-erasable EPROM
IC18—82C52 programmable UART
IC19—MC145406 RS-232 driver/receiver
D1—1N914 switching diode
LED1—Red LED, right-angle PC mount

Other components

OSC1—8-MHz crystal oscillator
XTAL1—2.4576-MHz crystal
B1—3.6-volt rechargeable NiCd battery (Panasonic MMB 3.6C)
S1—8-position DIP switch
S2—SPDT, momentary pushbutton switch
SO1—25-pin female connector, DB-25 type
SO2—16-pin double-row header, PC mount
SO3, SO4—60-pin male/female feedthrough bus connector assembly (Molex 70090/70088), see text
SO5—4-pin jack

Miscellaneous: PC board, extruded aluminum case (see Part 1), IC sockets, hardware, wire, solder, etc.

SURROUND SOUND

continued from page 49

correct. Note that there is no power switch between power plug PL1 and T1's primary winding. That was done because the power to the prototype is switched with the rest of the system to avoid turn-on pops.

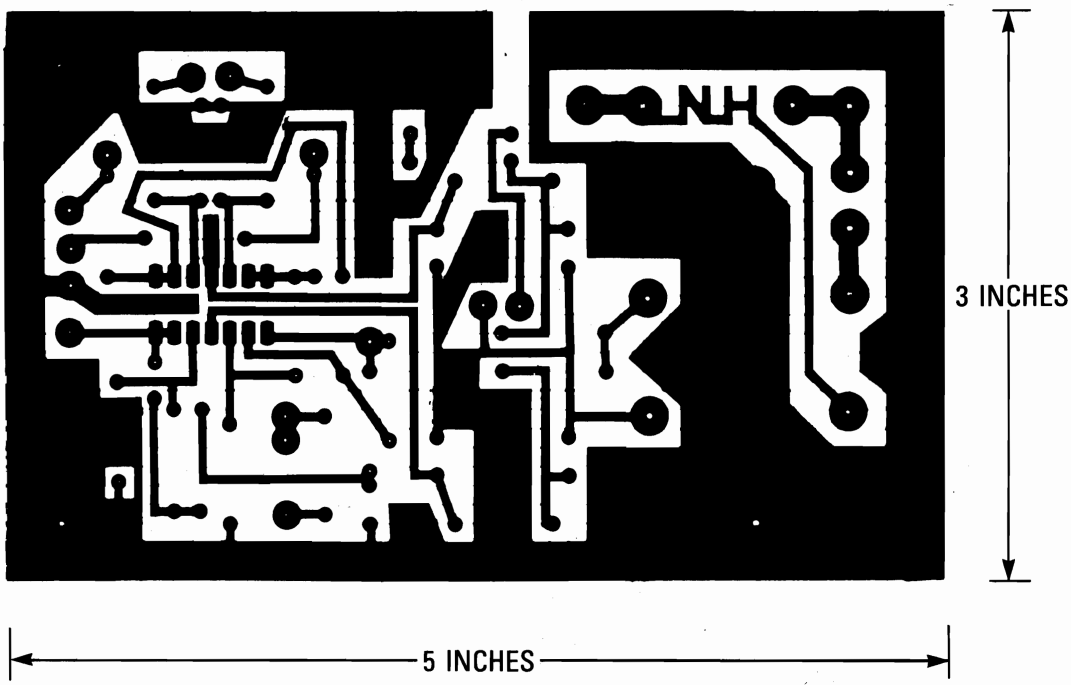
Setting up

Figure 7 shows how the surround-sound decoder can be set-up in a component video-sound system. Notice that by having the decoder connected between the preamplifier and the power amplifier it can also be used to decode signals that originate in a conventional tuner, a CD player, or whatever. Also note the use of left-rear and right-rear speaker signals even though the left and right decoder outputs, as previously discussed, are the same. Obviously, your particular video-audio system will be different, but Fig. 7 will give you a good idea of the various ways in which signal sources and amplifiers can be combined with the surround-sound decoder. As shown in Fig. 8, all of the decoder's inputs and outputs are made through phono jacks that match the conventional phono-plug patch cords that are used for all home video-sound connections.

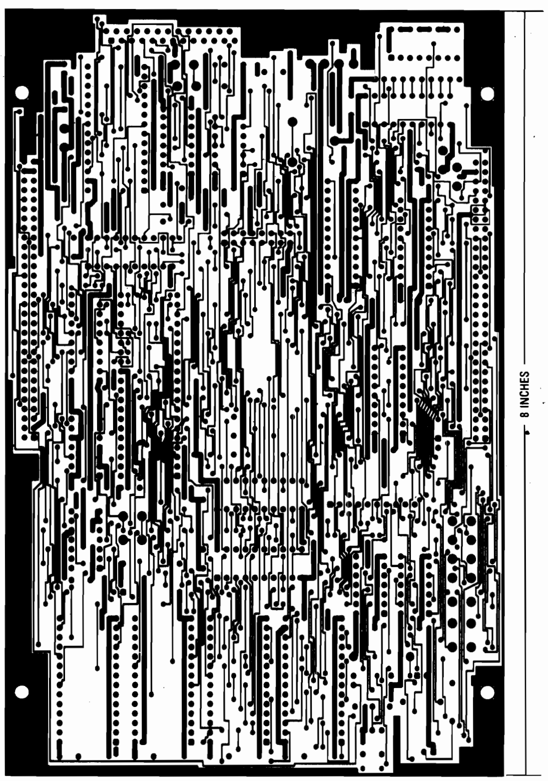
If your system doesn't use components, and the stereo outputs of your your VCR or TV normally drive an integrated amplifier or a receiver, simply connect the decoder between the VCR or TV and the amplifier's or receiver's AUX or VIDEO-SOUND inputs.

If all four speakers were identical, and if their driving amplifiers had identical gain, and if the front and rear speakers were equidistant from the viewer, no level balancing or adjustment would be necessary. But that's a lot of ifs. More than likely, you'll spend some time fiddling with the amplifier controls. A better way to calibrate Hi-Fi surround-sound system is to use a 7-minute video calibration tape called *Kwik-Kal: A Seven Minute Surround Test*, which features on-screen indication of speaker placement, the channel being encoded, and its weighting. The tape is available in the Beta and VHS formats and contains a Hi-Fi track. (See Parts List.) Once the system is optimized by using the tape, there's no need to fiddle with any adjustments. Simply sit back, relax, and enjoy the show. **R-E**

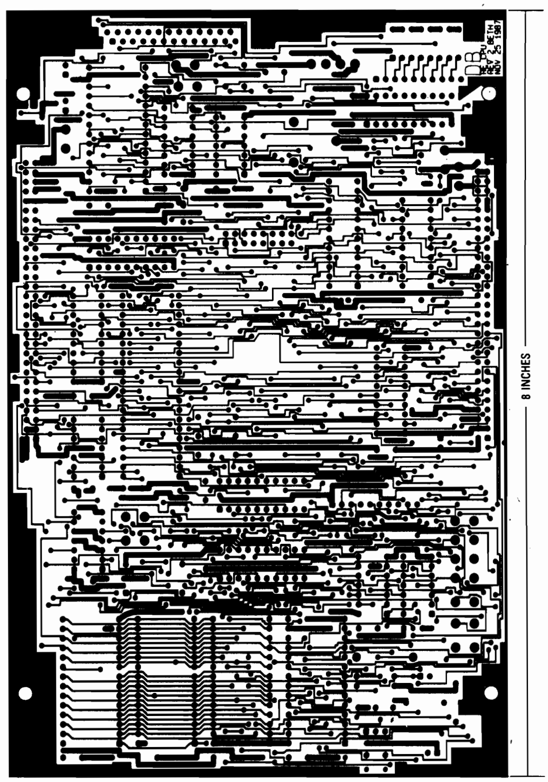
PC SERVICE



HERE'S THE PATTERN for the Surround-Sound decoder.



SOLDER SIDE for the REACT's board. It is shown half size.

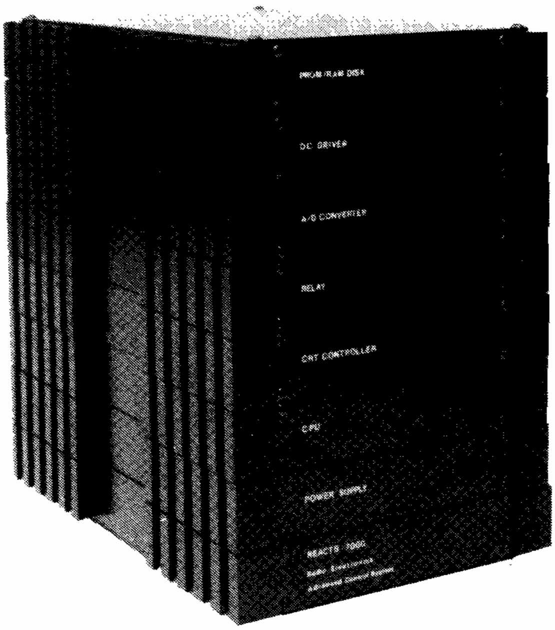


COMPONENT SIDE of the REACT's board. It is shown half size.

Build REACTS:

THE RADIO-ELECTRONICS ADVANCED CONTROL SYSTEM

This month, we add a way to store your programs and data.



H. EDWARD ROBERTS, M.D.

Part 4 LAST TIME, WE promised to show you a battery back-up system for the REACTS system. Well we will—but not this month! Instead, we will show you a powerful high-speed semiconductor-disk system for the control computer. That module will allow us to expand the size of our disk storage from the 20K provided by the CPU board to virtually any size desired, and to add read/write (RAM) storage to the disk system. And best of all, it will operate at speeds that are up to 1000 times faster than magnetic-disk systems.

Using semiconductor disks

To the computer programmer/user, semiconductor disks are manipulated in the same way as floppies or hard drives. That is, all commands that are used with floppies or hard drives (DIR, ERA, etc.) work the same way with semiconductor disks, only much quicker.

For many reasons, semiconductor memory is ideal for process-control applications. It is low in cost compared to magnetic storage, except for applications requiring huge amounts of memory. It is much more tolerant of temperature and vibration extremes than magnetic memory. Power consumption is only a fraction of what typical magnetic-disk systems require. The size of a small semiconductor-disk system is much less than that of a conventional disk drive. The only real disadvantage of that type of disk is that the read/write portion of the disk is volatile; that means that it must be powered continually. That's really not much of a problem, and it can be made even simpler when you build the battery-backed switching power supply that will be discussed in a future article. For now, it only means that you will have to keep your computer powered up if you want to save the contents of the RAM memory permanently.

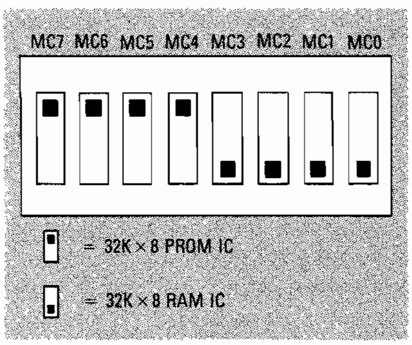


FIG. 1—THE SYSTEM IS CONFIGURED for RAM, PROM, or a combination of memory IC's using an 8-position DIP switch, S2.

The soft/hardware concept that we previously introduced is very evident in this module. For instance, you can configure the system to appear as 1 to 8 disk drives by simply setting switches on the back of the module. Also, multiple modules may be combined to increase the size of a single disk to over 2 megabytes. Any number of disks may be added to the system. By leaving out memory IC's, the cost of the disk system can be minimized in cost-sensitive applications. Once the hardware is customized as desired, the operating system can be changed using a configuration utility (more on that shortly). That will provide a truly customized system.

The REACTS drives

The PROM/RAM disk portion of this month's module provides 256K of PROM and/or RAM space. It contains 8 IC sockets into which either 32K RAM or PROM IC's can be inserted; any 32K × 8-bit RAM or PROM can be used. You let the system know which type of memory each socket location contains by setting a DIP switch. As shown in Fig. 1, a PROM/RAM location is configured for a PROM if its corresponding switch is up, and configured for a RAM if the switch is down.

Throughout the rest of this article, we will refer to the disk system as a drive or drives. That allows us to follow the accepted convention used in most operating systems. For instance, the first floppy in a conventional PC is known as drive A: and the hard disk is usually labeled drive C. Our system will support up to 16 drives, identified

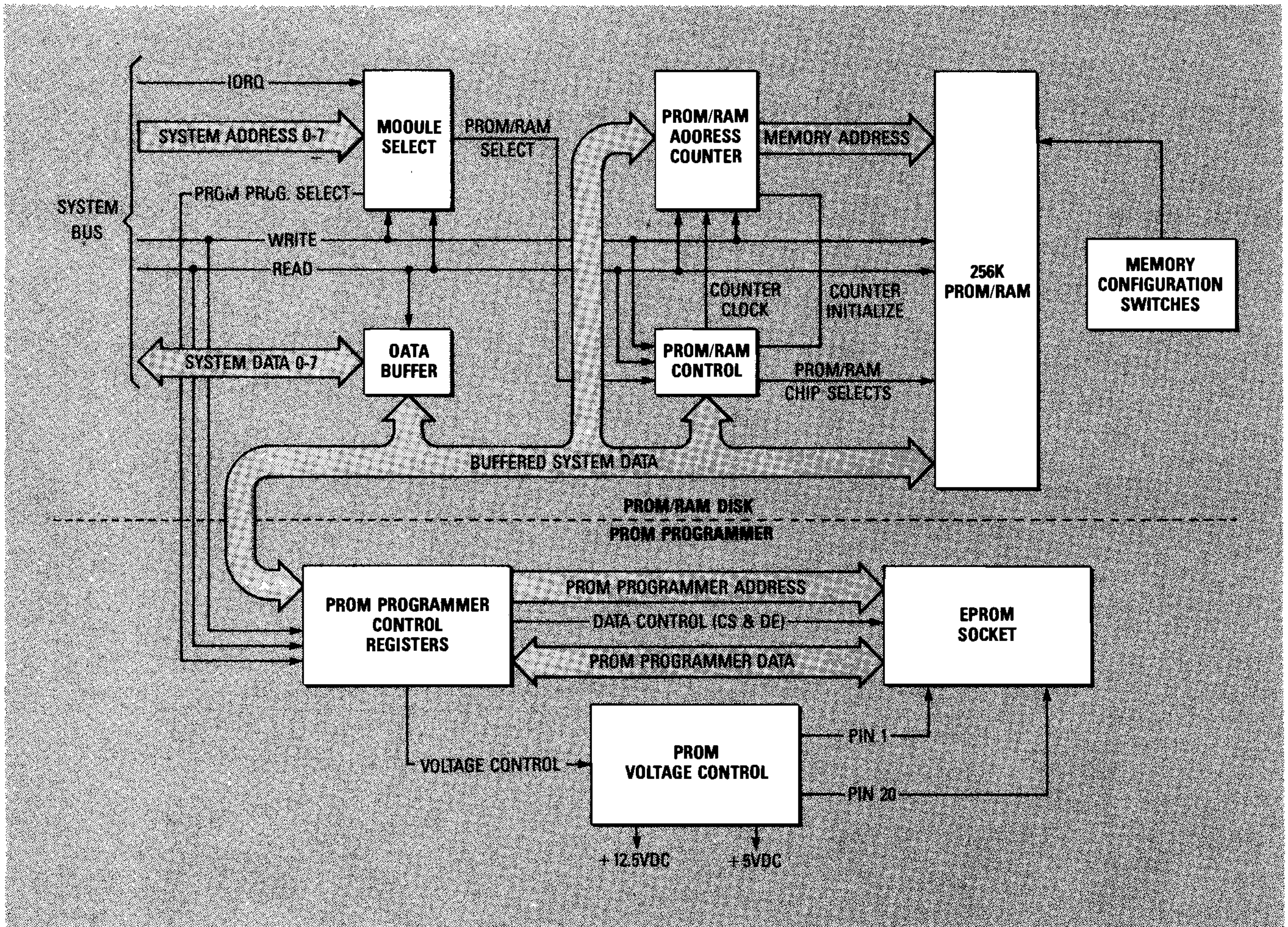


FIG. 2—THE PROM/RAM DISK and optional PROM Programmer are shown here in block-diagram form.

as drives A:–P:. Further, the REACTS operating system has been designed to divide the PROM or RAM disks into tracks and sectors, with each track made up of 8 sectors of 128 bytes each. That compares with the tracks and sectors of a floppy or hard disk.

Certain drives are reserved by the system for special operations; specifically, drive P: is reserved as the boot drive on the CPU. Future articles will provide the information necessary to add floppy and hard disk drives to the system. Those drives will be labeled in exactly the same manner. Indeed, a random mixture of floppies, hard disks, RAM and PROM disks is perfectly satisfactory from the system's standpoint.

It is not necessary to have all of the PROM or RAM space of a module designated as one drive. Up to 8 separate drives may be specified on one module; the only constraint is that the memory allocated to each drive must be a multiple of 32K (the size of a single memory IC). As an example, you could configure the system to have 128K of PROM memory set up as drive A: and 128K memory set up as drive B:. Another option would be to have two 64K drives of PROM and two 64K drives of RAM; the four could be designated as drives A:, B:, C:, and D:, where A: and B: would be RAM drives and C: and D: PROM drives.

It is even possible to have eight different drives on one PROM/RAM module, each containing 32K of PROM or RAM memory. At the other extreme it is possible to have more than two megabytes of PROM or

Sources

The following items are available from DataBlocks,Inc.,579 Snowhill Road, Glenwood, GA. 30428; 800 652-1336 (in Georgia call 912 568-7101

DP-P/R/PP: Design package of schematics and instructions; $10.00

PC-P/R/PP: PC Board for PROM/RAM/PROM Programmer module, includes design package; $37.00

PROM/RAM: Complete set of parts, PC board, IC's for PROM/RAM portion of module (excluding PROM and RAM memory IC's); $114.00

PROM PROG: Parts needed to add PROM Programming capabilities to module; $59.95

MEMORY IC's: 32K × 8 bit UV EPROM's; $10.50. 32K × 8 bit RAM IC's $12.50.

SOFTWARE: Software is available on UVEPROMs as well as on 5¼-inch floppies for downloading from an IBM PC, XT, AT or compatible. Please call for prices on the different software packages.

Other REACTS systems and components are available; please call for information and prices.

Please add $10 shipping and handling per order. Georgia Residents must add sales tax.

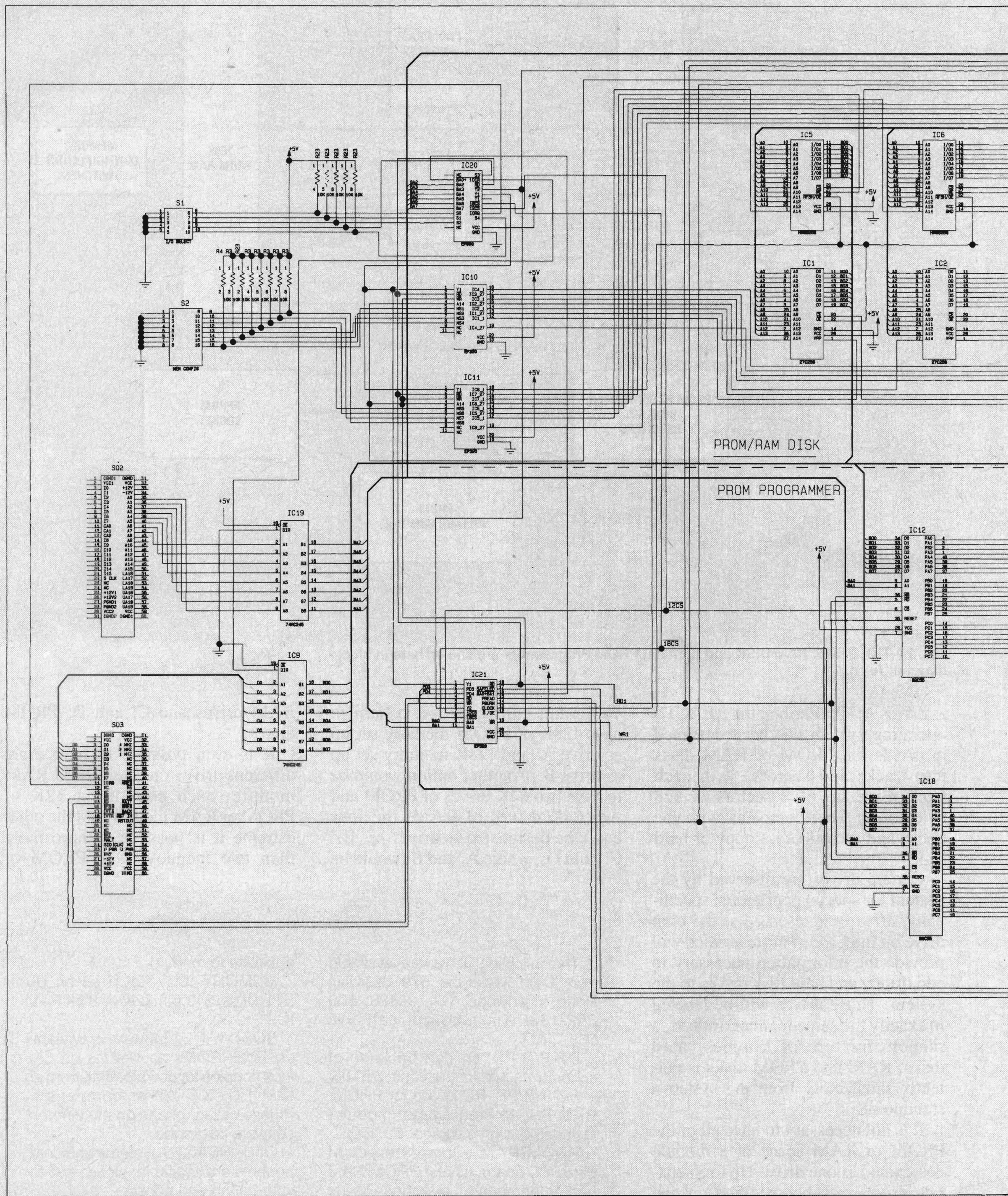
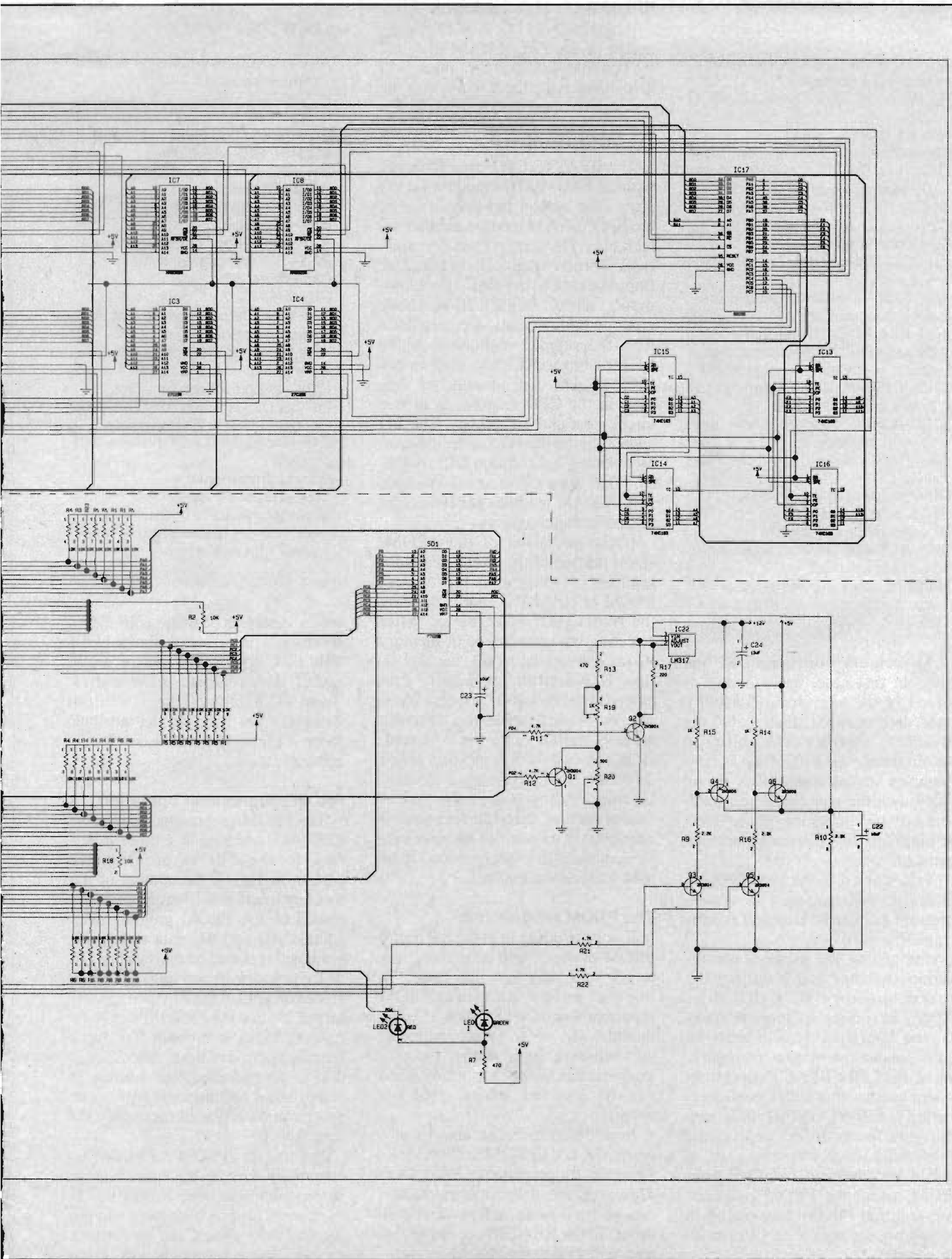
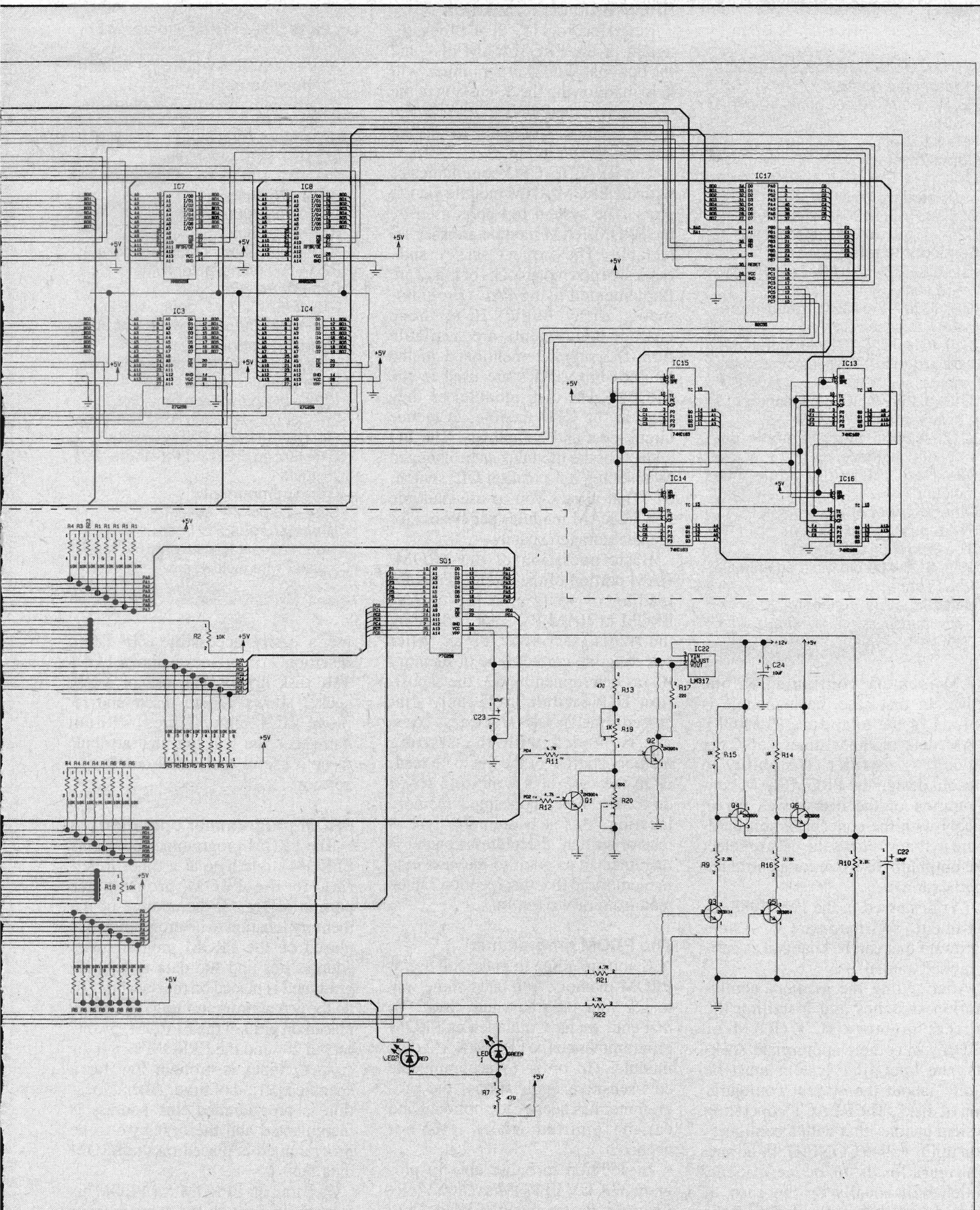


FIG. 3—THE MODULE'S CIRCUIT DETAILS are shown in this schematic.


IC17
IC7
IC8
IC3
IC4
IC15
IC13
IC14
IC16
74HC163
SD1
IC22
LM317
C24
C23
C22
R2
R7
R9
R10
R11
R12
R13
R14
R15
R16
R17
R18
R19
R20
R21
R22
Q1
Q2
Q3
Q4
Q5
Q6
LED1
LED2
RED
GREEN
+5V
+12V


IC7
IC8
IC3
IC4
IC17
IC15
IC13
IC14
IC16
SO1
IC22
LM317
ADJUST
VOUT
27C256
8255
74HC163
+5V
+12V
C22
C23
C24
R2
R4
R5
R6
R7
R8
R9
R10
R11
R12
R13
R14
R15
R16
R17
R18
R19
R20
R21
R22
Q1
Q2
Q3
Q4
Q5
Q6
2N3904
2N3906
LED2
LED1
RED
GREEN
10K
4.7K
470
220
1K
2.2K
500

Parts List—PROM/RAM Drive

All resistors ¼ watt, 5% unless otherwise noted
R3, R4, R23—10,000 ohms, 10-into-1 SIP
R7—470 ohms
Capacitors
C1–C11, C13–C17, C19–C21—0.47 µF, ceramic disc
C25—47 µF, 16 volts, electrolytic
C26—47 µF, 10 volts, electrolytic
Semiconductors
IC1–IC8—32K × 8 RAM or EPROM, see text
IC9, IC19—74HC245 octal three-state transceiver
IC10, IC11, IC20, IC21—Programmable array logic IC's, custom components, see text
IC13–IC16—74HC163 synchronous binary counter
IC17—82C55 programmable peripheral interface
D2—Red LED, right-angle PC-mount
Other components
S1—5-position DIP switch
S2—8-position DIP switch
SO2, SO3—60-pin male and female bus connector set
Miscellaneous: PC board, IC sockets, hardware, case, wire, solder, etc.

RAM memory configured as one drive. In that case, each module is only ⅛ of the total drive (8 PROM/RAM disk modules times 256K per module = 2048K). The ability to custom design the PROM/RAM configuration of the disk drives fits in nicely with the concept of soft-hardware in that it allows the user to select the optimum disk size for a particular application.

(As discussed in the Feb. 1988 installment, soft-hardware is system hardware that can be changed as easily as software.)

After setting the memory configuration switches and installing the correct memory IC's (RAM or PROM) into their appropriate sockets, the operating system must be "told" about the current configuration of disks. The REACTS operating system handles that with a configuration utility called CONFIG. If the new configuration is to be permanent, which will usually be the case, it should be "burned" onto a boot PROM, using the PROM programmer, and that PROM then placed in the appropriate socket on CPU-module board.

PROM/RAM disk operation

Referring to Fig. 2, a block diagram of the PROM/RAM disk and the optional PROM programmer, will help in following the discussion of the operation of the PROM/RAM portion of the module. Complete circuit details are shown in Fig. 3.

The REACTS CPU communicates with the PROM/RAM module via I/O ports. The system I/O ports used by the PROM/RAM module are user-selectable. The MODULE SELECT and PROM/RAM CONTROL blocks of Fig. 2 are implemented using PAL (*P*rogrammable *A*rray *L*ogic) IC's; those custom components are available from the supplier mentioned in the Sources box. PAL's are used in the PROM/RAM disk module, as they were in the CPU module, to reduce circuit cost and complexity. The I/O address of the module can be changed by resetting a 5-position DIP switch, S1. That allows you to use multiple PROM/RAM modules per system for greater storage capacity.

Basic operation of the PROM/RAM portion of the module calls for reading or writing to the correct PROM or RAM IC. That is done by the PROM/RAM CONTROL block. After selecting the correct type of memory IC (as determined by S2, the 8-position DIP switch previously discussed), the PROM/RAM ADDRESS COUNTER is loaded with the starting memory address of the file to be read, or in the case of a write to a RAM drive, the first unoccupied memory location. As each memory byte is read or written, the address counter is incremented to point to the next byte in memory so that that location can be read from or written to.

The PROM programmer

If you are going to make full use of PROM memory, you need some way to get your programs into them. To that end, we have included an PROM programmer on the PROM/RAM disk module. (In order to accommodate cost-sensitive applications, the programmer has been made optional and can be omitted where it is not needed.)

In addition to being able to program 32K UV EPROM's (*U*ltra *V*iolet *E*rasable *P*rogrammable *R*ead *O*nly *M*emory), the programming socket can be used as an independent disk drive. To the REACTS operating system, a PROM in the PROM programmer's easily accessible ZIF (*Z*ero *I*nsertion *F*orce) socket looks like a 32K disk drive. (The use of a ZIF socket allows you to insert and remove IC's many times without damage.) You will find that attribute to be a useful tool when developing software.

Parts List—PROM Programmer

All resistors ¼-watt, 5%, unless otherwise noted
R1, R5, R6, R8—10,000 ohms, 9-into-1 SIP
R2, R18—10,000 ohms
R9, R10, R16—2200 ohms
R11, R12, R21, R22—4700 ohms
R13—470 ohms
R14, R15—1000 ohms
R17—220 ohms
R19—1000 ohms, potentiometer
R20—500 ohms, potentiometer
Capacitors
C12, C18—0.47 µF, ceramic disc
C22–C24—10 µF, 16 volts, tantalum
Semiconductors
IC12, IC18—82C55 programmable peripheral interface
IC22—LM317 voltage regulator
Q1–Q3, Q5—2N3904 NPN transistor
Q4, Q6—2N3906 PNP transistor
D1—Green LED, right-angle PC mount
Other components
SO1—28-pin ZIF socket
Miscelaneous: PC board, IC sockets, 28-pin hi-rise socket, hardware, wire, solder, etc.

PROM-programmer operation

The PROM programmer programs EPROM's one byte at a time. Referring to the PROM programmer portion of Fig. 2, the address of the memory location to be programmed is placed on the PROM programmer's address bus and the data to be programmed is placed on the data bus. To do the actual programming, 12 volts is placed on pin 1 of the EPROM, 5 volts on pin 28, and the EPROM's CE (CHIP ENABLE) input is brought low for a specified period of time. After a location is programmed, the address is incremented and the next byte to be programmed is placed on the PROM data bus.

Reading an EPROM (or PROM) is essentially done in the same manner as programming one, except that 5 volts are placed on both pin 1 and pin 28, and both the CE and OE (OUTPUT ENABLE) pins are brought low. Once

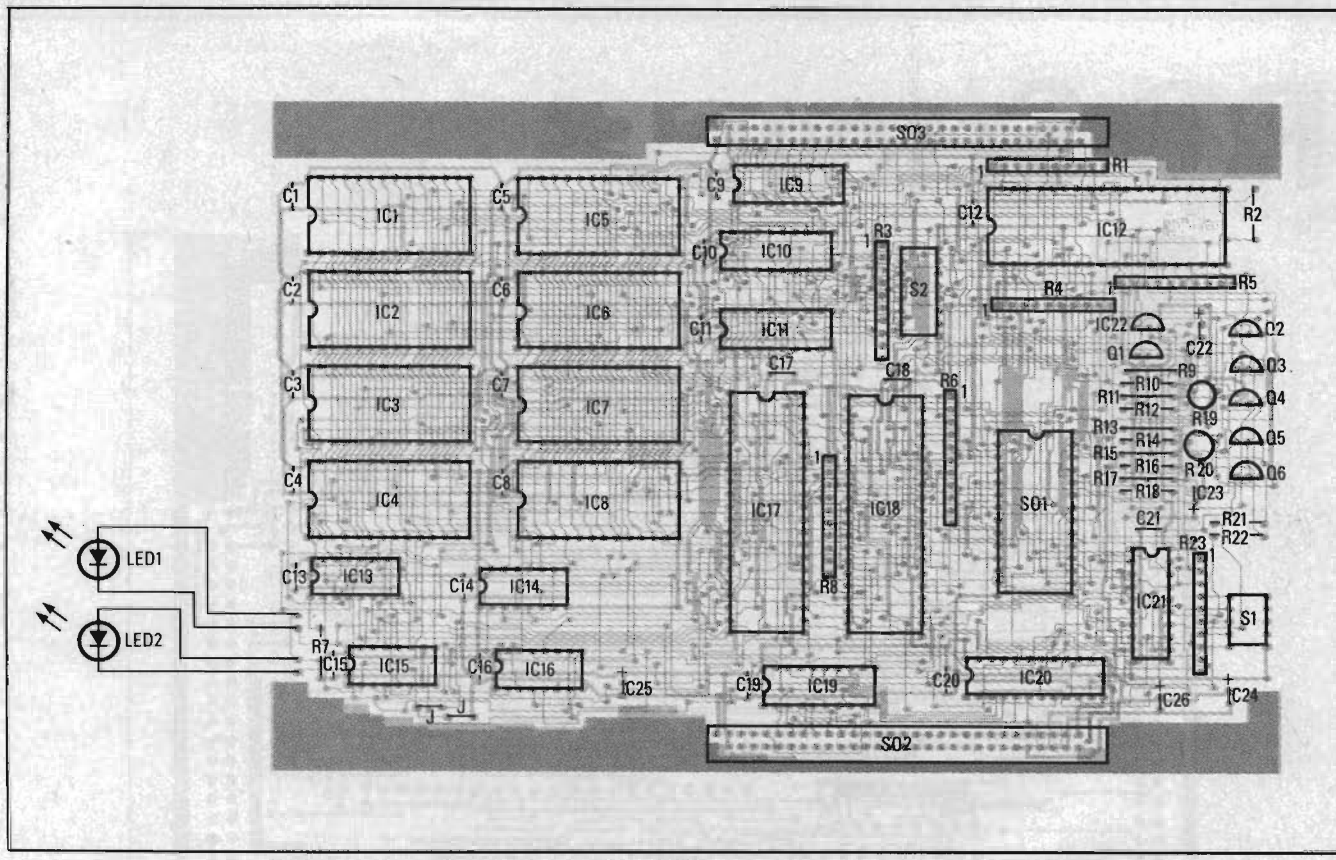


FIG. 4—CONSTRUCTION OF THE MODULE is fairly straightforward. Use this Parts-Placement diagram when mounting the components.

again, to keep circuit complexity and cost as low as possible, much of the programmer's circuitry is incorporated in a custom PAL. (That IC is available from the supplier.)

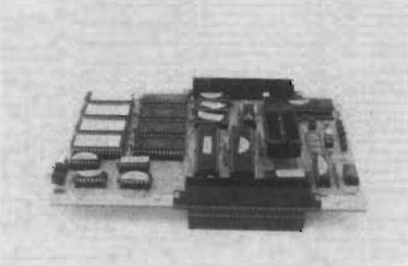
The REACTS operating system contains a utility called RBURN that makes programming EPROM's easy. That utility uses prompts to allow the user to select the file to be programmed, the drive that the file is located on, and the number of kilobytes to be programmed. In addition to burning EPROM's, the RBURN utility can be used to check an EPROM for complete erasure (i.e. to make absolutely sure that a PROM is blank before programming).

Two LED's are used by the module to display the state (idle or busy) of the PROM programmer. If the red LED is illuminated, the EPROM is either in the process of being burned or read and should not be disturbed. To avoid damaging the device, EPROM's should be placed in or removed from the programming socket only when the green LED is on.

When handling EPROM's, remember that they are CMOS components and all of the usual precautions regarding static electricity should be followed to avoid inadvertently zapping them. Also, be sure that any EPROM's are oriented correctly in the socket before attempting to burn or read them otherwise, the device may be damaged.

Building the module

Building the combination PROM/RAM disk and PROM programmer is basically a straightforward operation. Follow the parts-placement diagram shown in Fig. 4 when mounting the components. The pattern for that board can be found in PC Service; a pre-etched board is available from the supplier that is mentioned in the Sources box.



THE FINISHED PROM/RAM DISK. Note the ZIF socket near the center of the unit.

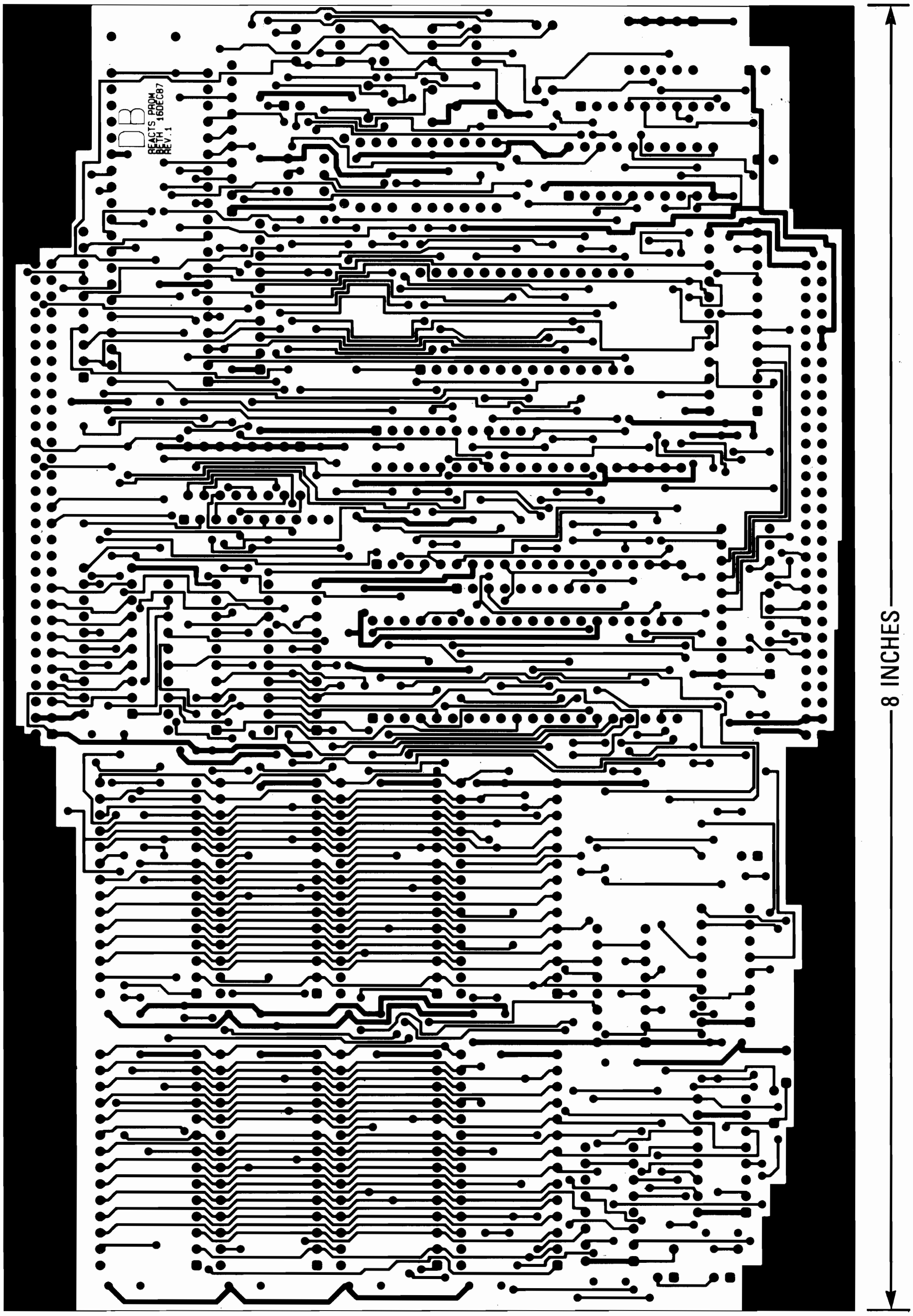
Be sure to observe the cautions outlined last time when installing the two 60-pin PC-board connectors. (Though they are rugged units once assembled, they can be easily damaged during assembly.) Also, take the usual precautions when handling the CMOS IC's. Finally, because we are putting so much circuitry in a small area, we have chosen to use *S*ingle *I*nline *P*ackage (SIP) resistors where applicable. Those are reasonably new components and you may not have used them before. However, SIP resistors are becoming more commonplace so they should not be to difficult to locate.

Software

Although we have not yet built any actual process-control modules, we can now start program development. We have a central processing unit, a means of communicating with it (a dedicated terminal or an IBM or compatible configured to act as a terminal; see the April issue for more on that), and at least one way of saving programs, that being the PROM-programmer/semiconductor-disk module. In order to create any programs

continued on page 82

PC SERVICE

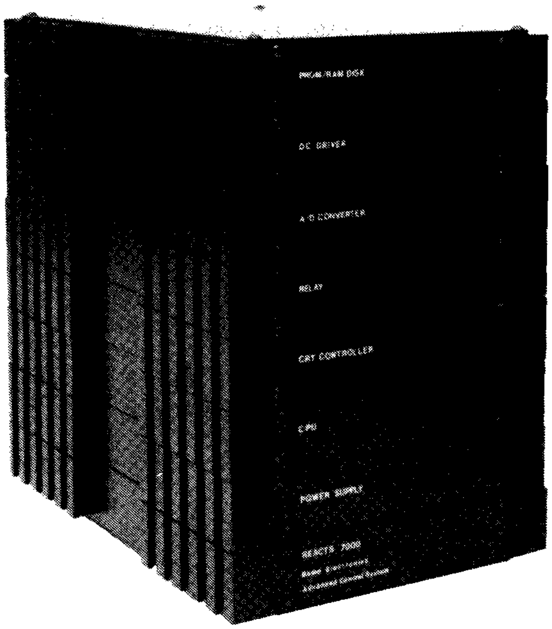


COMPONENT SIDE for the REACTS RAM/PROM Disk and PROM Programmer. The solder side will appear next time. See page 100 for more PC patterns.

Build REACTS:

THE RADIO-ELECTRONICS ADVANCED CONTROL SYSTEM

This month, we show you how to put REACTS to work to control the outside world.



H. EDWARD ROBERTS

Part 5 IN THE PREVIOUS articles in this series we built the basic REACTS control computer. In this article we will discuss how the REACTS can perform useful tasks; that is, provide automatic control for the home. We will build an electronic lock and provide automatic control of various appliances in the home. In a future article we will add a security system and we will also provide automatic environment control.

What makes control applications like those possible is an eight-channel input and output block that we will use to provide partial automation of a home. Specifically, we will build an electronic door lock and provide for the automatic sequencing of appliances and lights.

What we will show you this month is just a small portion of our ultimate home-control system. In future articles we will add capabilities such as environmental control and a security system. The reader should notice that the application shown is designed to be both practical as well as illustrative; we hope it will

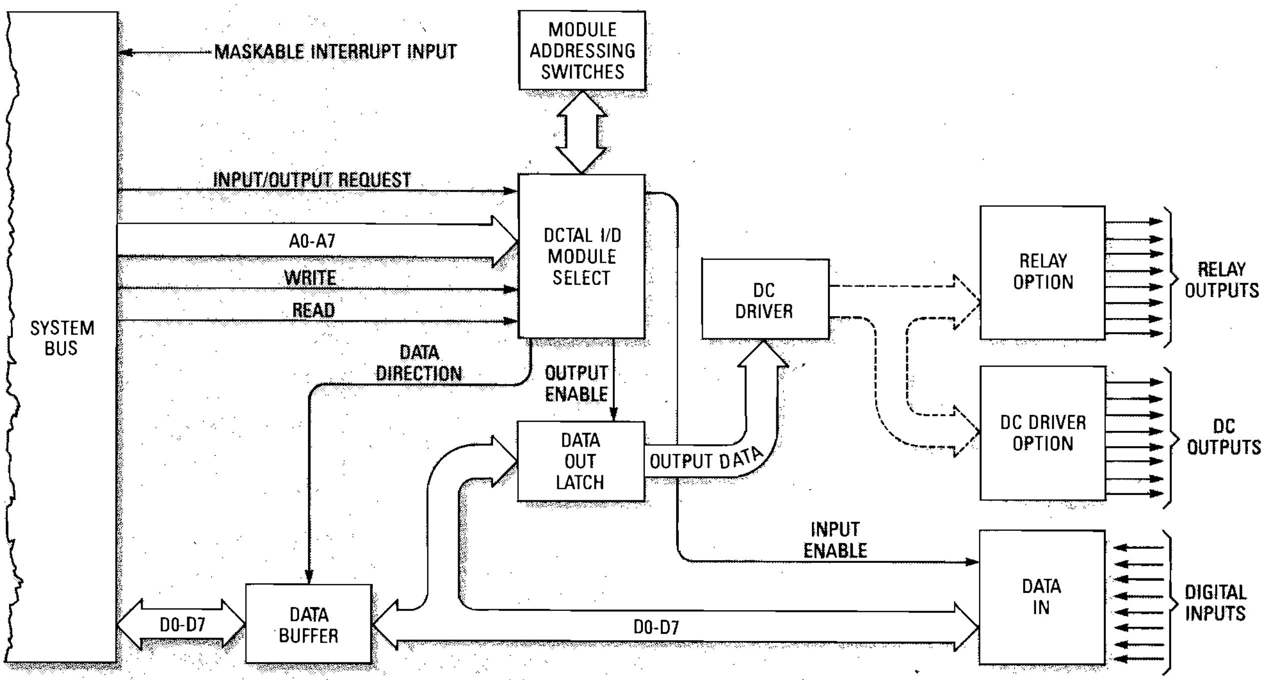


FIG. 1—THE REACTS OCTAL I/O MODULE allows you to interface the system to the outside world. The optional relay outputs even allow you to control AC devices.

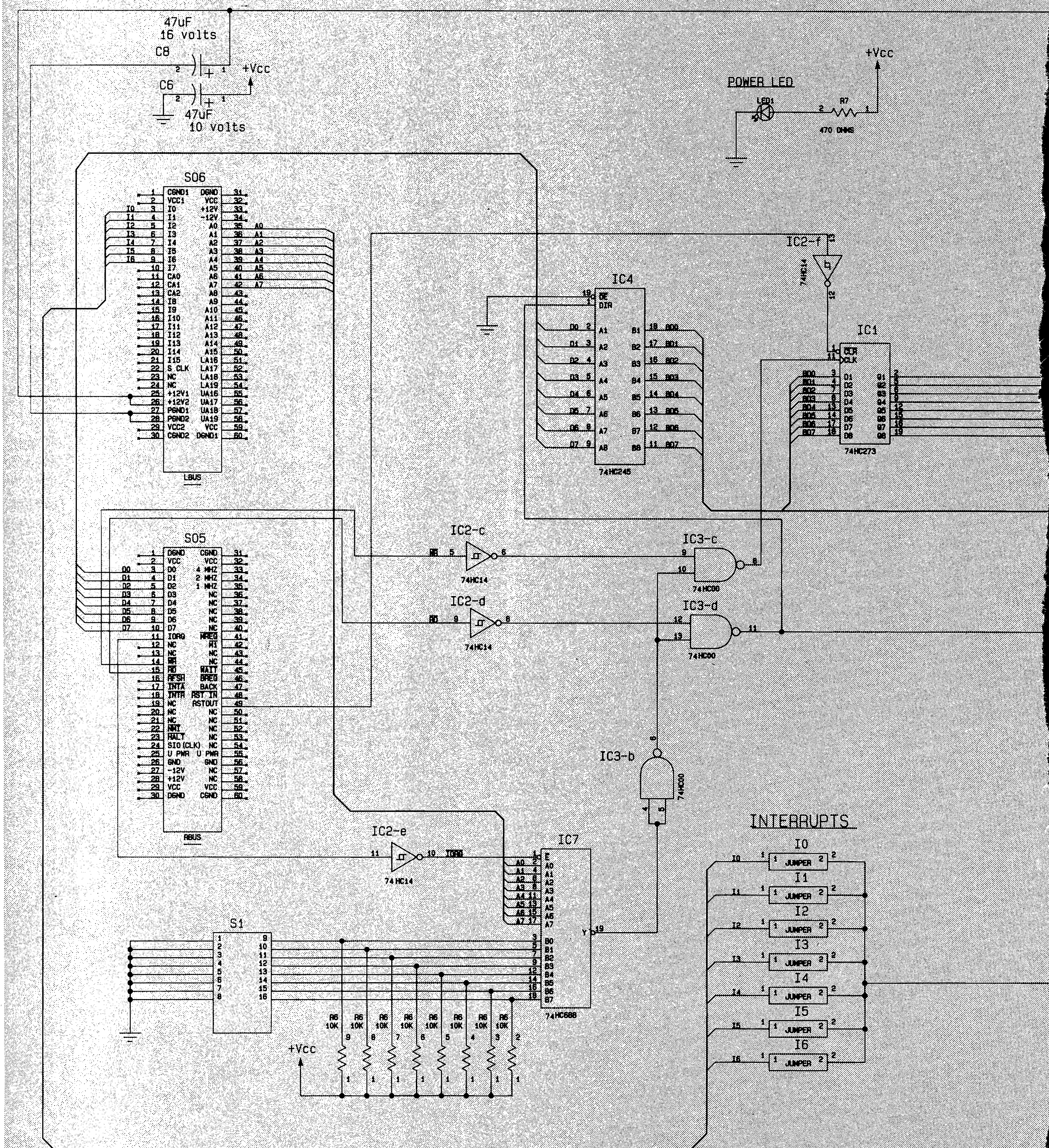
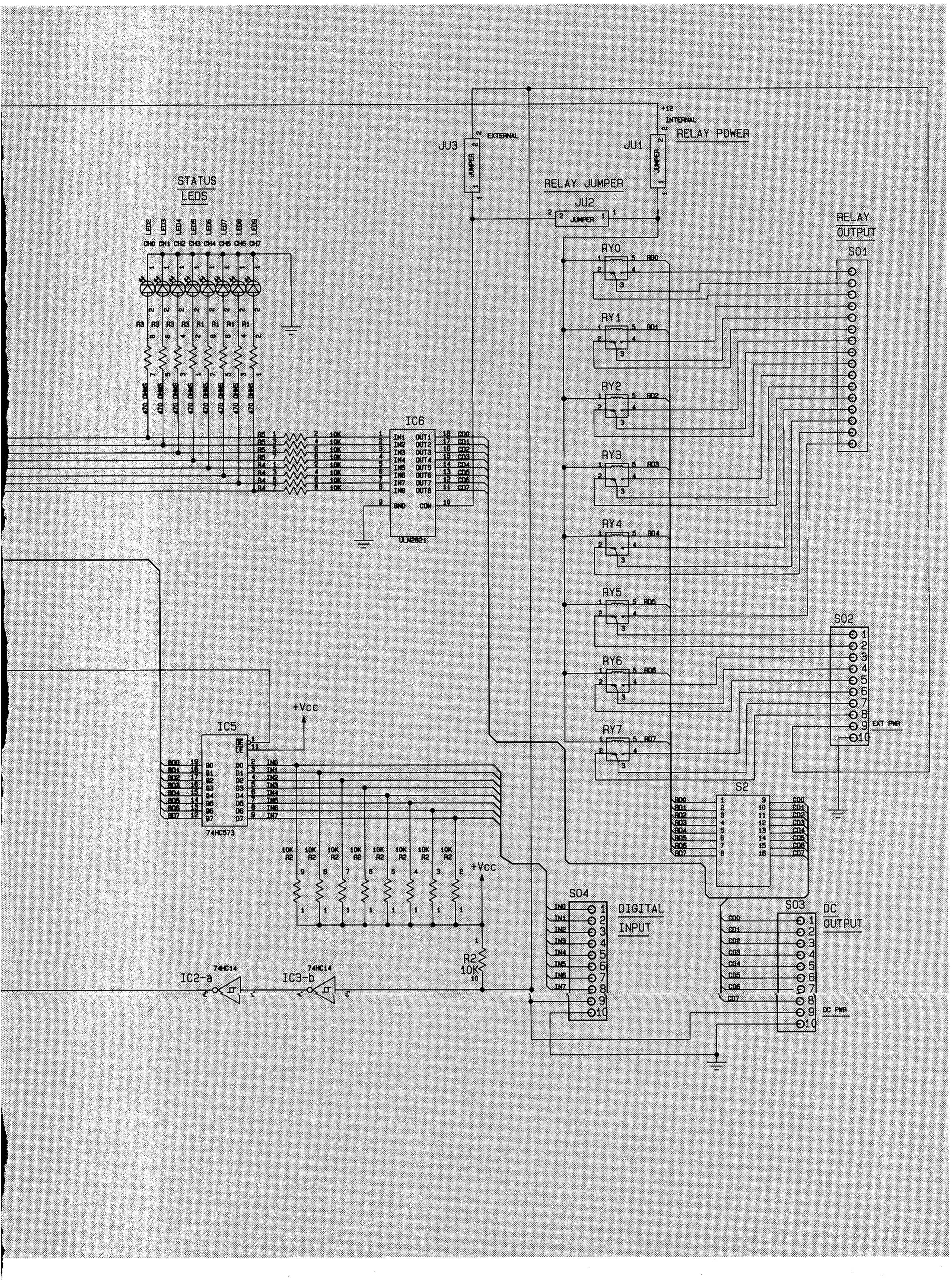


FIG. 2—CIRCUIT DETAILS of the module are shown in this schematic. Switch S1 allows you to change the module's port address as needed.


STATUS LEDS
RELAY JUMPER
RELAY POWER
RELAY OUTPUT
EXTERNAL
INTERNAL
JU1
JU2
JU3
JUMPER
+12
IC6
ULN2821
IC5
74HC573
+Vcc
S01
S02
S03
S04
S2
EXT PWR
DC PWR
DIGITAL INPUT
DC OUTPUT
RY0
RY1
RY2
RY3
RY4
RY5
RY6
RY7
IC2-a
IC3-b
74HC14
R2
10K
470 OHMS

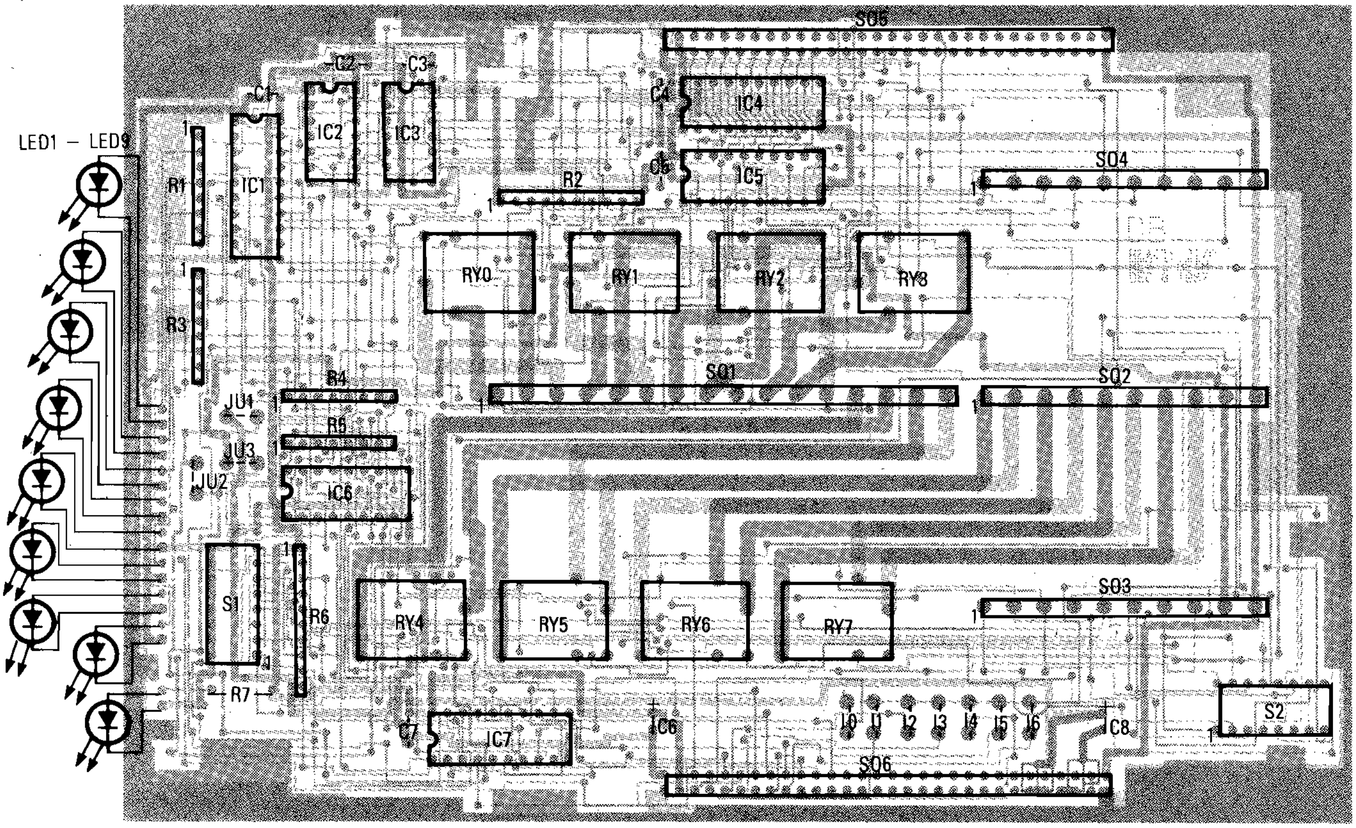


FIG. 3—BUILD THE MODULE following this parts-placement diagram. The PC pattern can be found in PC Service.

PARTS LIST

All resistors ¼ watt, 5%, unless otherwise noted

R1, R3—4 × 470 ohms, isolated resistor network, SIP
R2, R6—10,000 ohms, 9-into-1 resistor network, SIP
R4, R5—4 × 10,000 ohms, isolated resistor network, SIP
R7—470 ohms

Capacitors

C1–C5, C7—0.47 µF, ceramic disc
C6—47 µF, 10 volts, electrolytic
C8—47 µF, 16 volts, electrolytic

Semiconductors

IC1—74HC273 octal D flip flop
IC2—74HC14 inverting Schmitt trigger
IC3—74HC00 quad 2-input NAND gate
IC4—74HC245 octal three-state transciever
IC5—74HC573 octal three-state D-type latch
IC6—ULN2821 Darlington transistor array
IC7—74HC688 8-bit magnitude comparator
LED1—red LED, right-angle PC mount
LED2–LED8—green LED, right-angle PC mount

Other components

S1, S2—8-position DIP switch
SO1—16 position terminal strip
SO2–SO4—10 position terminal strip
SO5, SO6—60-pin male and female bus-connector set
RY1–RY8—6-amp relay, PC mount (optional, see text)

Miscellaneous: PC board, hardware, IC sockets, case, etc.

give you some idea on how to use the REACTS control computer.

The I/O module has eight universal inputs and outputs. The outputs are eight Darlington-array DC drivers (an option of eight 6-amp relays is also available). The inputs are TTL level.

How it works

Before we proceed, a brief review of the input/output system of the Z80 is in order. As you may remember, REACTS has 256 input and output channels. Each channel has eight lines that may be set on or off independently. That gives us a total of 2048 (256 × 8) potential switch closings and 2048 input sensing lines; more than enough for even the most complex application.

Each input and output channel has a specific address; i.e., address 0 through 255. Let's look at how a computer performs a typical input operation. All microprocessors operate using a cycle system. Each cycle performs one type of operation, such as a memory read, a memory write, an I/O input, an I/O output, etc. Using the status lines, the processor "tells" the rest of the system what it is going to do at the beginning of each cycle. The signals on the status lines identify the type of operation to be performed. For instance, if the system is going to perform an input, the IORQ (INPUT/OUTPUT REQUEST) line becomes high. That tells the system that the operation is to be either an input or an output. The system decides whether or not the operation is an input or an output by "looking" at the RD (READ) and WR (WRITE) lines. If the RD line is low the cycle is an input, if the WR line is low the cycle is an output.

So far we have determined that the particular cycle is an input based on the IORQ and the RD line. But which of the 256 possible inputs is to be used. That is determined by simply looking at the lower eight address bits. Those bits will be set with the correct address (i.e., the address of the required input channel) at the beginning of the cycle. That's all there is to it. Note that all eight lines will be read in at one time. Now let's look at how that operation appears to the software. Using BASIC, the command to input the data would be as follows:

```
40 X = INP(23)
```

That command instructs the computer to input channel 23 and store its eight lines in a variable called X. You will see later how we are able to separate the individual lines from the variable.

SOURCES

The following items are available from DataBlocks, Inc
579 Snowhill Road, Glenwood GA. 30428. Or call (800) 652-1336; in Georgia call (912) 568-7101.

- DP-OCT: Design package with schematic and instructions; $10.00.
- PC-OCT: PC board for the octal I/O, including design package; $37.00.
- REC-OCT: Complete set of parts including PC board, IC's, terminal strips, etc., but excluding relay option listed below; $109.00.
- REC-REL: OCTAL I/O relay option includes eight, 6-amp relays and terminal strips; $42.00.
- REC-CASE and PANEL/REC-OCTAL: Aluminum case assembly including front and rear panels; $19.50.
- REC-KEY: 12-button keypad with scan-code decoding program; $9.00
- VAC-PROG: vacation home-control program; free with SASE.
- Other REACTS products are available; contact DataBlocks directly for information and pricing.

Please include $5.00 postage for any order of $37.00 or less and $10.00 postage for any order over $37.00. Georgia residents must add appropriate sales tax.

Module operation

Referring to Fig.1, a block diagram, and Fig. 2, a detailed schematic of the I/O module, will be of great help to you in following the module operation discussion.

The octal I/O module provides eight digital inputs and a choice of eight DC or relay-isolated outputs (note the dotted bus line on the block diagram of Fig. 1). Only one of the module's output options (either DC or relay) can be used at a time; however, room is left on the printed-circuit board to install terminal strips (the external devices to be controlled are connected to those) for both. Selecting between the DC or relay option is accomplished by simply moving a few power jumpers and setting some switches. That capability to readily change between output options is useful when developing an applications system.

The module's circuitry performs three main functions: module selection, data output (activation of the DC or relay outputs), and data input.

Module selection occurs when the A (A0–A7) inputs of IC7, a 74HC688 eight-bit magnitude comparator, match its B (B0–B7) inputs and the IORQ line is active. Note that the B inputs of IC7 are connected to a bank of eight switches while the A inputs are connected to the lower eight address lines. That allows the input/output port used by the module to be user-selectable. That is, the module will be selected only when the IORQ line is active and the I/O address on the lower eight address lines matches the I/O address selected via the bank of switches.

Data input occurs when the module is selected and the RD line is low. That places the data at terminal strip SO4 onto the system data bus by enabling the 74HC573 latch (IC5) and changing the data direction flow of the 74HC245 buffer (IC4) so that data flows from the B side to the A side. Since no CMOS inputs should be left floating, pull-up resistors have been connected to the inputs of IC5.

With the inputs pulled high, an open input (i.e. one that is not connected to any external device) will cause the corresponding bit to be high (logic 1). The fact that the inputs are pulled high should be taken into consideration when developing software for the module. That is, we normally think of a logic 1 as being "on" and a logic 0 as being "off". With an open input that is pulled high, or with an open switch connected to the input, a logic 1 will be input. Although that is perfectly legal, it tends to be contrary to our normal way of thinking. The simplest way to remedy this is to complement the input(s). Another solution would be to use a normally closed switch. In the doorlock-application programs we will discuss shortly, we use the NOT command to complement the inputs.

When the module is selected and the WR line is low, the data on the system's data bus is latched into IC1, a 74HC273 octal D-type flip-flop. When data is being written to the module, the DIRECTION (DIR) input of IC4, a 74HC245 octal three-state transceiver, is high, causing the direction of data flow to be from the A side to the B side. Note that the outputs of IC1 are connected to eight status LED's. Those provide instant checking of the status of each output (whether they are on or off) at a glance. The outputs of the IC1 are also connected to IC6, a ULN2821 that contains eight Darlington drivers.

If the DC option is used, the devices to be controlled by the module are connected indirectly (via SO3) to the outputs of the Darlington-array package (ULN2821). The ULN2821 is a sink device, which basically means that the Darlington drivers turn on the device(s) being controlled by completing the electrical path to ground. When the Darlington pairs are turned off, the electrical circuit is opened (no path to ground), thus turning the device off. Each DC output will sink up to 130 mA at up to 95 volts. Higher current outputs are attainable by paralleling two or more outputs. To select the DC option, a jumper is placed in jumper location JU3 and all switches of S2 are closed.

The relay option uses IC6 to control the coil pull-in voltage of each relay. Each relay has normally open, normally closed, and center contacts. The +12 volts (pull-in voltage) needed to activate the relays can be provided by the REACTS system's power supply or an external source. If the system power supply is used, care should be taken not to exceed its maximum current ratings.

To select the relay option, a jumper is installed in location JU2 and all switches of S2 are opened. To select an external power source for the relays, a jumper is installed in location JU3; to use the system power supply as the relay power source, install a jumper in location JU1.

Because many process-control applications require instantaneous response, provisions have been made on the module for using one of seven (I0 through I6) of the system's eight maskable interrupt inputs. The system's maskable interrupts are prioritized; that is, interrupt 0 has precedence over interrupt 1, interrupt 1 has precedence over interrupt 2, etc. The interrupt input to be used by this month's module (if any) would depend on the importance of the task the module is performing as compared to tasks performed by other modules.

Construction and checkout

Building the octal I/O module is fairly simple. The main concerns are the installation and soldering of the two 60-pin bus connectors and making sure that the input and output terminal strips are correctly oriented. The available kits listed elsewhere in this article provide the quickest and simplest way of building the module. The kits come with all the necessary components, a silk-screened printed-circuit board, and complete construction and module operation instructions. For those who do not wish to go the kit route, a PC pattern will be provided next month in PC Service; the board should be built following the parts-placement diagram shown in Fig. 3.

After building the module, it must be tested. Before testing you must decide which of the system's I/O ports you want the module to occupy. When choosing an I/O port, do not choose any that are already in use by the CPU or any other module in your system. It is a good idea to keep an I/O-configuration log that lists all the I/O ports used by each module. That way, as new modules are added, you need only consult the I/O-configuration log to determine which of the I/O ports are unused.

If the module has been configured for the relay option, a simple test is to connect one lead of an ohmmeter to the center output of one of the relays and the other lead to the normally open output of the same relay. With the relay open, the resistance reading should be infinite, and when the relay is closed, the ohmmeter should read zero. To close a relay, a logic 1 must be placed in the correct position of the byte output to the I/O port used by the module. For example, in the following BASIC program, the module is located at I/O port 24, and the relay to be closed is relay 2. The module is set to I/O port 24 by placing the fourth and fifth switches from the left on the modules I/O-addressing switches (S1) in the up position and the other six switches in the down position. Note that a 1 is at bit 2 of the byte being output to the module; that will cause relay 2 to close.

```
10 OUT 24, &X00000100

20 END
```

Note: The two characters, &X, preceding the binary number, 00000100, in the program tell the interpreter or compiler that the number is in binary form. The characters &X are used by the ZBASIC programming language; the syntax may differ with other BASIC programming languages.

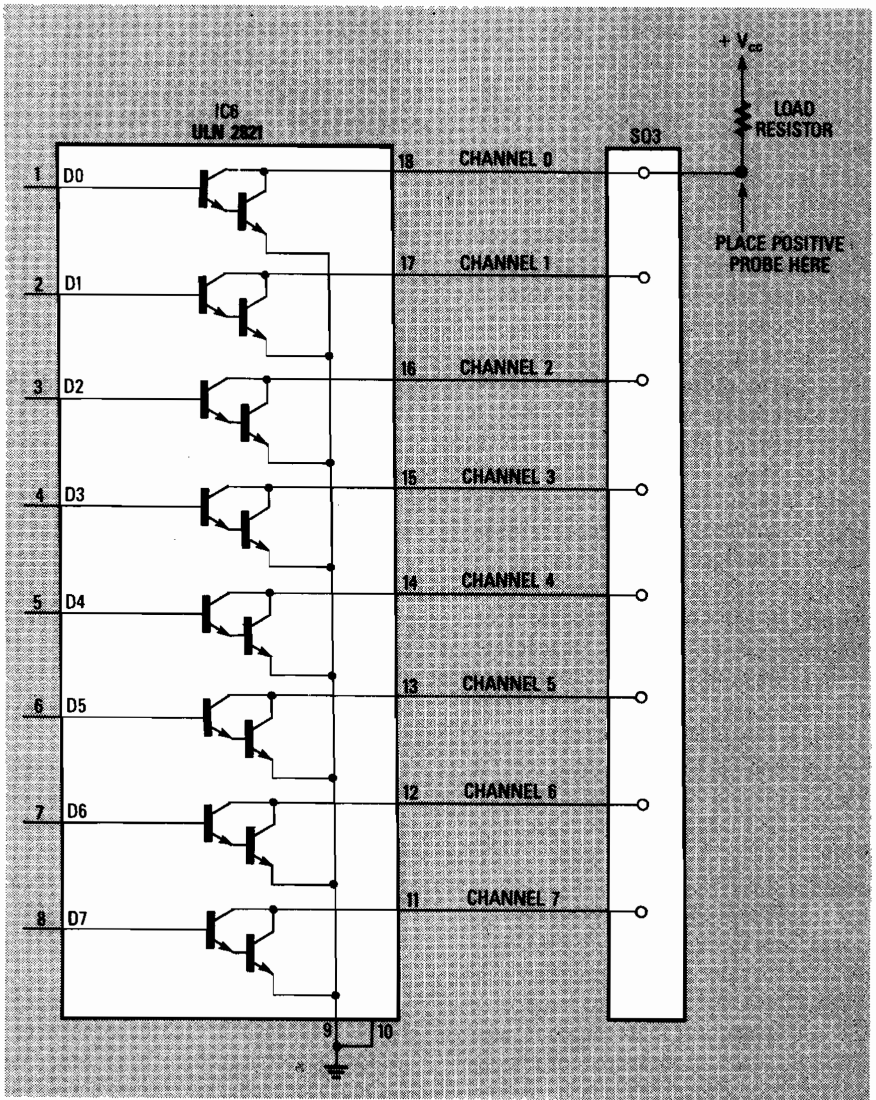


FIG. 4—THE DC OUTPUTS can be tested using the arrangement shown here.

Since the outputs of the DC option of the module are open-collector outputs, a resistor that will simulate a load must be connected between the external power supply and the output(s) being tested; see Fig. 4. When choosing a resistor for that purpose, remember that each DC output's maximum current-handling capability is 130 milliamps. Also, calculate the amount of power that must be dissipated by the load resistor and make sure the resistor's wattage rating will accommodate at least twice that.

For example, if the power supply is set to 12 volts DC and an 800-ohm resistor is used, the amount of current drawn will be approximately .015 amp, or 15 milliamps (12 volts divided by 800 ohms), which is well below the output's maximum current rating. Also the amount of power to be dissipated by the resistor is approximately 0.18 watts (.015 amps times 12 volts), so a resistor capable of dissipating at least 0.36 watts is needed.

After you have connected the external power supply and the resistor, (or load) to one of the DC outputs, connect the positive lead of a DC voltmeter to the DC output slot on the

continued on page 62

the SI, and that would be universally accepted.

Approximately every 10 years, physicists around the world get together to adjust the fundamental constants, such as the charge on an electron, the mass of a proton, and Plank's constant. Those changes reflect improved measuring techniques, and new ways of measuring those quantities. The constants relating to the volt and the ohm, will be readjusted in 1990. The cutoff date for values of $2e/h$, and h/e^2 is June 15, 1988.

To produce an exact voltage, the NBS places a Josephson junction, shown in Fig. 10, in an environment near absolute zero (−273.15°C). At that temperature, the junction behaves like two superconductors separated by a thin nonconducting film. The junction is then irradiated with microwaves from a klystron tube that is phase-locked to NBS WWVB atomic time signals, while a stable DC current source is used for bias. The DC bias current is increased at a constant rate until a critical value is reached. The voltage output from the Josephson junction is zero until the critical DC bias current, at which point, the voltage output will jump to a specific value ΔV dependent on the microwave frequency. If the DC bias current is increased to a slightly greater value, the junction voltage will jump to $2\Delta V$, and so on, each voltage jump in perfectly equal ΔV steps. At the point where the ΔV voltage is at the correct level, it is then interfaced to an external voltage comparator, which uses an NBS standard cell as a reference.

The Josephson junction produces a DC voltage of between 1 mV and 10 mV depending on the injected microwave frequency. If we remember that the microwave frequency is phase-locked to atomic time, then we have a highly repeatable and accurate method that produces a DC voltage approaching the accuracy of the atomic-time standard. Furthermore, that is a portable system, limited only by reception of WWVB, or equivalent time signals. In 1984, NBS researchers and the West German National Standards laboratory have developed ways to link-up about 1,500 Josephson junctions on a single integrated circuit to produce a reliable 1-volt standard. And in 1985, a NBS researcher integrated an array of 2076 junctions that can be tuned up to more than 2 volts. The feasibility of commercial manufacture of Josephson junction arrays is presently being studied.

Real world calibrators

We have come a long way in electrical measurements during the last 40 years, from volt-ohm meters with accuracies of 3% to 10% to modern digital voltmeters with accuracies of 0.001%. That 10,000-fold increase in measurement accuracy was mainly fostered by both military and aerospace demands for instruments having greater accuracy.

You will find real-world calibrators divided into two broad categories. Those are passive and active. A passive instrument requires no external power source, contains no signal amplification circuitry, and will generally yield greater accuracy than active ones. An example of a passive instrument is a decade resistor box. An example of an active instrument is a digital voltmeter. You will find passive instruments providing most NBS primary standards, with tertiary standards typically being provided by active instruments.

Because the SI definitions of the volt, ohm, and ampere are difficult to realize in practice, the NBS laboratories have historically used practical representations of them based on artifacts to act as primary standards. For example, the electrochemical standard Weston cell (1.018 volts) may serve to define the laboratory voltage standard. Similarly, a group of precision wire-wound resistors (1 ohm) may serve to define the laboratory standard of resistance. The laboratory unit of current (1 ampere) is defined in terms of a lab volt, and a lab ohm. Unlike the ohm (wire-round resistor), or the volt (Wetson voltage cell), the ampere does not have its own artifact that you can carry from one place to another.

If the primary standards are maintained by artifacts, what maintains their accuracy over time? The answer is intrinsic standards. Most national laboratories now use the AC Josephson effect in superconductors to maintain their unit of voltage. While not yet in as wide use as the Josephson effect, the quantum Hall effect promises to do for resistance what the Josephson effect has done for voltage. R-E

REACTS

continued from page 56

terminal strip. Then connect the ground lead of the voltmeter to the ground slot of the terminal strip. When the DC output being tested is turned off, the voltmeter should read approximately the same voltage as is being supplied by the power supply. When the output is turned on, the voltmeter should read from 0.7 to 1.1 volts. Earlier, we noted that the DC outputs are sink devices that complete the connection to ground to turn the load being controlled on. Since the devices are transistor arrays, there will still be a small voltage drop across them when they are turned on.

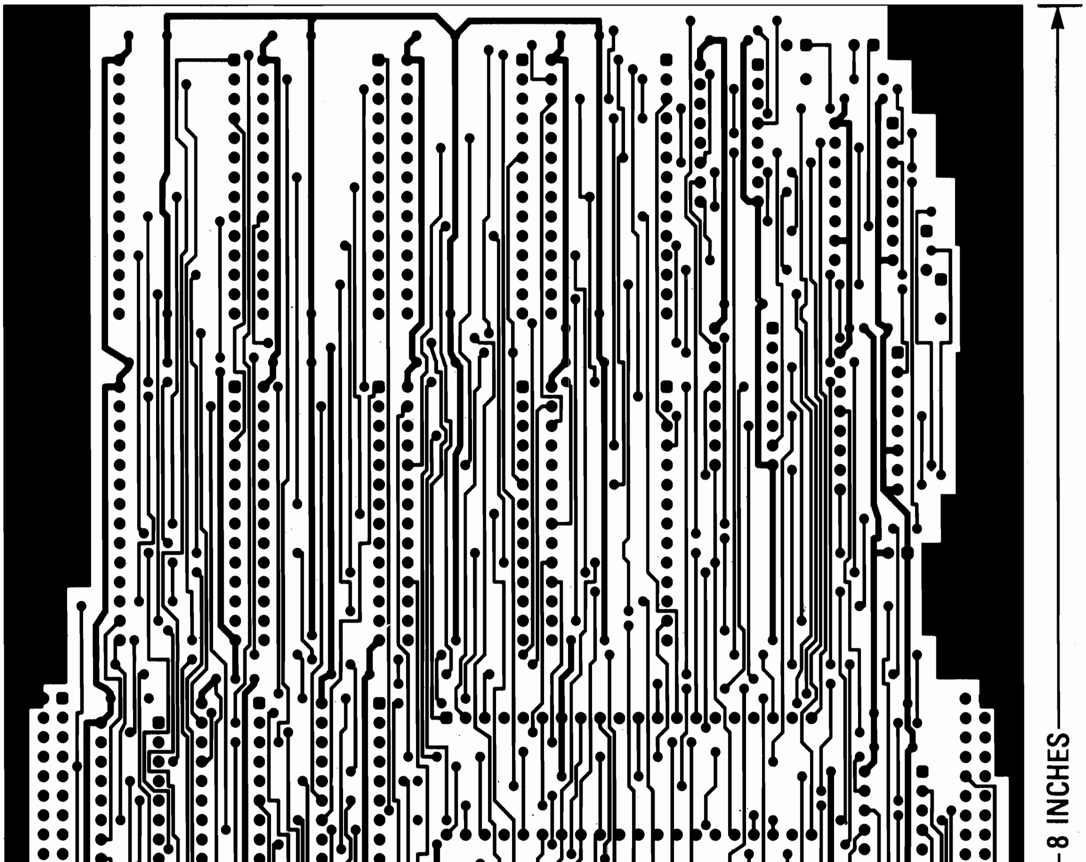
When testing the input portion of the module, keep in mind that all the inputs are pulled high. With none of the inputs connected to anything, an input from the module should result in a binary reading of 11111111 (all inputs high). With channel 0 grounded, an input from the module should result in a binary reading of 11111110 (channel 0 is low). Unground channel 0 and ground channel 1, and an input from the module should result in a reading of 11111101 (channel 1 is low). Proceed with that method and test each channel. The following BASIC program can be used to perform the test:

```
10 REM MODULE IS ADDRESSED AT I/O PORT 24

20 REM IP = INPUT FROM MODULE

30 IP = INP(24)

40 PRINT BIN$(IP)

50 END
```

Note: Statement 40 tells the CPU to display the input data in binary form. The syntax may vary between different versions of BASIC.

That's all we have space for this month. Next time we'll take a look at some applications. Among them will be a digitally controlled combination lock that will reliably protect your home. We'll also show you how to control household appliances. For example, you'll be able to automatically turn your lights on and off for a lived-in appearance. R-E

PC SERVICE

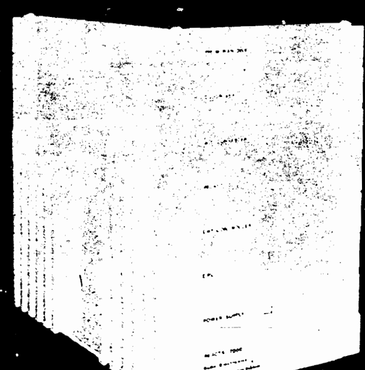


THE REACTS PROM/RAM DISK BOARD. The solder side of last month's board is shown here right reading and full size. The patterns for the REACTS I/O board will be shown next time when we discuss applications using it.

BUILD THIS

Build REACTS: THE RADIO-ELECTRONICS ADVANCED CONTROL SYSTEM

This month, we show you how to put REACTS to work to control the outside world.



H. EDWARD ROBERTS

Part 6

WE LEFT off last month promising to show you how to put REACTS to work. To get your imagination rolling on what you can do with a control computer, we'll show you how to build a digital door lock, an appliance controller, a security system, and a sprinkler controller for your garden.

The digital lock

The input/output port consists of 8 wires (bits) that are labeled from 0 to 7. Using the INP command we discussed earlier, we can read the data present at the inputs. Incidentally, similar commands exist in Assembly, C, FORTRAN, and other languages.

As mentioned earlier, we input or output eight lines or bits each time we perform an input or output. In the following example we will see how to mask each line (or bit) out independently when needed.

Figure 1 shows a simple eight-pushbutton combination lock. In order to make that lock work, we simply read the input channel that is connected to the switches. That data is then compared with the combination stored in the program shown in Listing 1. If the data matches, the system releases the lock by outputting the correct command to the output line, which controls the latch solenoid. That solenoid is activated to stay open for some predetermined period, say 2 seconds. If the door is not opened by that time the door will relock.

That program will work quite adequately for a lock and you can program 256 different combinations. It has one major disadvantage; it uses all eight inputs of the I/O module. Listing 2 shows an electronic entry program that only uses five switches and allows for over 6500 combinations. The door-lock program will be only one subroutine in your final home-automation program. Along with the door-lock program, the final program may contain subroutines for controlling appliances and lights, a security system, and a lawn-watering system; monitoring and controlling the home environment; and allowing remote control using a modem. It takes approximately 100 microseconds to make one check of the switches. So even if the switches are checked 20 times a second, the door-lock subroutine will only take up approximately 2 milliseconds.

It is important to understand that when outputting to a channel, all eight bits of the channel are affected. That is true even when you want to change only one bit. The technique we use is to store all the data that is output to the channel in a common location. Then, each routine that uses the storage location is responsible for seeing that only the bits relevant to that particular routine are changed. For example, in line 160 of Listing 1, we were only interested in changing the output bit (bit 0) that activated the relay that the door-lock solenoid was wired to. To keep from changing any of the other bits, we performed a masking operation by ORing them with 0. The relay was closed by ORing a 1 at bit 0.

The two electronic door-lock programs discussed and listed in this article require simultaneous pressing of two or more switches or pushbuttons for many of the numerical entries. However, a ten or more digit keypad that uses some type of scan code easily could be implemented so that only one pushbutton is pressed for each entry number. Those types of keypads are available from several electronics hobbyist and parts houses. The supplier listed in the Sources box offers one such keypad. The keypad from that vendor comes complete with a listing of a scan-code decoding program for serial digit entry.

Figure 2 shows the connections of the external solenoid for the door control. The technique you use to actually lock and unlock the door is up to you. If you desire, an electric lock is available from commercial suppliers. Please send us any ideas that you have that might be of interest to others; we will pass those along to other users of the system. Ideas can also be shared on the Radio-Electronics Bulletin Board (516-293-2283, 8 data bits, 1 stop bit, no parity).

Controlling appliances

Because the relay option of this month's module can be connected directly to 117 volts AC, household appliances and lighting can be controlled by REACTS with the module as long as the current drawn from each relay does not exceed 6 amps. An ideal application is to use the system to turn-on and turn-off lamps while you are on vacation to give your home a "lived in" appearance to would be burglars. Securing the leads

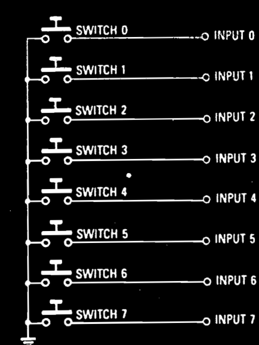


FIG. 1—A SIMPLE PUSHBUTTON LOCK. The data on the input lines is read by the computer and then compared to a combination in the lock program.

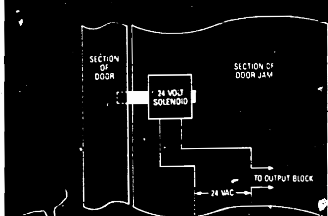


FIG. 2—ONE METHOD of implementing the door lock uses a 24-volt solenoid under computer control. Of course, many other schemes are possible.

LISTING 1

```
10 REM Sample lock program, correct combination is 131
20 REM (10000011 binary)
30 REM
40 REM      TEST = Variable at which input byte is stored
50 REM      ENT = Entry from switches
60 REM      LCK24 = Controls output portion of module
70 REM
80 TEST = 0
90 ENT = 0
100 GOSUB "FETCH ENTRY"
110 IF TEST = 0 THEN GOTO 100
120 IF TEST > ENT THEN ENT = TEST
130 GOSUB "FETCH ENTRY"
140 IF TEST <> 0 THEN GOTO 120
150 IF ENT <> 131 THEN GOTO 80
160 LCK24 = LCK24 OR &X00000001
170 REM     This sets the last bit (bit 0) to a 1 without affecting
180 REM     any other bits. The output that controls the solenoid
190 REM     is connected to bit 0 of output channel 24.
200 OUT 24,LCK24
210 DELAY 6000
220 REM Causes program to wait for 2 seconds.
230 LCK24 = LCK24 AND &X11111110
240 OUT 24,LCK24
250 REM     This will clear the 0 bit which will relock the door.
260 REM     Here also, none of the other bits are affected.
270 GOTO 80
1000 REM ********************** FETCH ENTRY **********************
1010 "FETCH ENTRY"
1020 REM    This subroutine inputs the byte from I/O port 24 at
1030 REM    which the octal I/O is addressed.
1040 TEST = INP(24)
1050 TEST = NOT TEST
1060 REM    Because all the inputs are pulled high, the inputted
1070 REM    byte is complemented with the NOT command in order to
1080 REM    simplify programming.
1090 REM    TEST = TEST AND &X11111111
1100 REM    After complementing the input with the NOT command, a
1110 REM    16 bit signed integer is returned. To obtain the
1120 REM    original 8 bit complemented input from the I/O port
1130 REM    we must mask out the lower 8 bits.
1140 RETURN
```

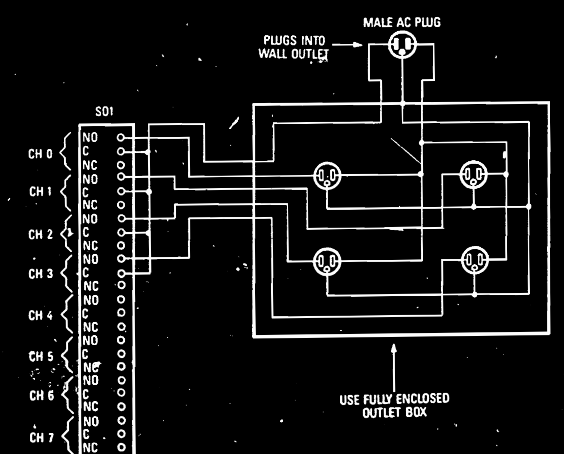


FIG. 3—FOR CONVENIENCE, use a relay interface like the one shown here when using REACTS to control household appliances.

LISTING 2

```
10 REM This program uses 5 bits of a single input channel to
20 REM produce a sophisticated combination lock which has over
30 REM 6500 possible combinations.  This program is designed to
40 REM use input channel 24.  The combination for this program
50 REM is 9312.  Any unused I/O channel or combination could
60 REM be used.
70 REM
80 REM          COUNT() = Array where valid entries are stored.
90 REM          IP_FLAG = Signifies that entry has been made.
100 REM         TEST = Variable at which input is stored.
110 REM         HI_IP = Temporary storage location for entry.
120 REM         DIGIT = Number of valid entries.
130 REM
140 DIM COUNT(5)
150 IP_FLAG = 0
160 TEST = 0
170 HI_IP = 0
180 DIGIT = 0
190 REM ******************** START LOCK **************************
200 "START LOCK"
220 GOSUB "READ INPUT"
230 IF BOUNCE_OK = 0 THEN GOTO "START LOCK"
240 IF TEST = &X00010000 THEN GOTO "CLEAR LOCK"
250 IF RDY <> 1 THEN GOTO "START LOCK"
260 IF TEST = &X00000000 AND IP_FLAG = 0 THEN GOTO "START LOCK"
270 IF TEST <> &X00000000 AND TEST > HI_IP THEN HI_IP = TEST
280 IF TEST <> &X00000000 THEN IP_FLAG = 1
290 IF TEST <> &X00000000 THEN GOTO "START LOCK"
300 COUNT(DIGIT) = HI_IP
310 HI_IP = 0
320 IP_FLAG = 0
330 DIGIT = DIGIT + 1
340 IF DIGIT = 5 THEN GOTO "CK COMB"
350 GOTO "START LOCK"
1000 REM ******************** READ INPUT **************************
1010 REM This subroutine reads the input and maskout the 5 least
1020 REM significant digits.
1030 "READ INPUT"
1040 TEST = INP(24)
1050 TEST = NOT TEST
1060 TEST = TEST AND &X00011111
1070 DELAY 10
1080 TEST2 = INP(24)
1090 TEST2 = NOT TEST2
1100 TEST2 = TEST2 AND &X00011111
1110 IF TEST = TEST2 THEN BOUNCE_OK = 1 ELSE BOUNCE_OK = 0
1120 RETURN
2000 REM ******************** CLEAR LOCK **************************
2010 REM Clearing lock and setting for new entry.
2020 "CLEAR LOCK"
2030 RDY = 1
2040 DIGIT = 1
2050 GOTO "START LOCK"
3000 REM ****************** CHECK COMBINATION ********************
3010 REM Checking for correct combination.
3020 "CK COMB"
3030 IF COUNT(1) <> 9 THEN GOTO "START LOCK"
3040 IF COUNT(2) <> 3 THEN GOTO "START LOCK"
3050 IF COUNT(3) <> 2 THEN GOTO "START LOCK"
3060 IF COUNT(4) <> 1 THEN GOTO "START LOCK"
3070 REM Opening lock.
3080 UNLCK = UNLCK OR &X00000001
3090 OUT 24,UNLCK
3100 UNLCK = UNLCK AND &X11111110
3110 DELAY 6000
3120 OUT 24,UNLCK
3130 GOTO "START LOCK"
3140 END
```

of the device (lamps) to be controlled involves stripping the insulation on the cords back leaving approximately ¼ inch of bare wire which is inserted into the appropriate slot on the relay terminal strip. The wire is then secured by tightening a screw against it.

If your spouse doesn't take too kindly to the idea of cutting the plugs off of your lamps and stripping the insulation back, there is another alternative. You can make an interface box with a 1900-series electrical (outside-type) box, a pair of duplex outlets, wire, and some hardware. A drawing of a typical interface is shown in Fig. 3. That device will allow you to easily connect appliances to the relay and possibly keep you out of the doghouse. If you are unsure about building the interface yourself, get an electrician to build it for you; it should be a cinch for him or her.

Note: Powerline AC voltages and currents can be dangerous if care is not exercised; you can easily damage the computer, or yourself. If you lack experience with powerline voltages and currents, we strongly recommend you consult an electrician.

With the REACTS CPU and the real-time clock option, the lamp(s) can be made to come on and go off at different times during the day and at different times on different days. Also, since the program controlling the lamps would reside on the system's boot PROM, in the case of a power outage, when the system came back on it would automatically reload the program and continue its job. A free copy of a vacation home-control program is available from the vendor listed in this article if you send a self-addressed stamped envelope.

Open window detector

The open-window-detector can be used in two different ways. First, it can be used to quickly check for open windows before leaving the house. Second, by connecting one of the octal I/O port's to an alarm, REACTS can be programmed to sound the alarm if a window is opened. Of course the alarm would have to be disabled at times when the window must be opened. That could be accomplished by using one of the CPU's sense switches or connecting a SPST switch to one of the octal I/O's inputs. The system would be programmed to periodically check the switch, and when the switch was open the alarm would be disabled; likewise, when the switch was in the opposite position, the alarm would be enabled.

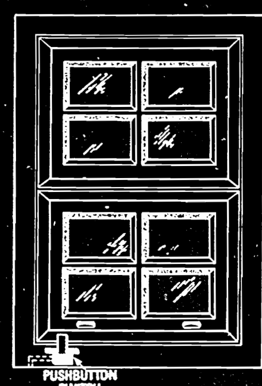


FIG. 4—A MOMENTARY SWITCH can be used to detect whether a window is open or not. When the window is fully closed, the button is depressed.

LISTING 3

```
10 CLS                                  'Clear the screen
20 FLAG = @X00000000                    'Initializing
variable flag as all windows are closed
30 OCT_IN = INP(100)                    'Inputting from
octal I/O
40 ALARM = OCT_IN AND &X00000001        'Inputting status of
alarm enable/disable
50 CHECKW = OCT_IN AND &X11111110 'Checking window
switches
60 IF FLAG = CHECKW THEN GOTO 30  'If status of
windows has not changed keep checking them
70 CLS                                  'Clear screen,
getting ready to display new window status
80 IF ALARM = 1 THEN OUT 100,1          'If alarm is
enabled, turn it on
90 REM *** Finding out which window(s) is open ***
100 WIND1 = CHECKW AND &X00000010
110 IF WIND1 <> 0 THEN PRINT "WINDOW #1 IS OPEN"
120 WIND2 = CHECKW AND &X00000100
130 IF WIND2 <> 0 THEN PRINT "WINDOW #2 IS OPEN"
140 WIND3 = CHECKW AND &X00001000
150 IF WIND3 <> 0 THEN PRINT "WINDOW #3 IS OPEN"
160 WIND4 = CHECKW AND &X00010000
170 IF WIND4 <> 0 THEN PRINT "WINDOW #4 IS OPEN"
180 WIND5 = CHECKW AND &X00100000
190 IF WIND5 <> 0 THEN PRINT "WINDOW #5 IS OPEN"
200 WIND6 = CHECKW AND &X01000000
210 IF WIND6 <> 0 THEN PRINT "WINDOW #6 IS OPEN"
220 WIND7 = CHECKW AND &X10000000
230 IF WIND7 <> 0 THEN PRINT "WINDOW #7 IS OPEN"
240 IF CHECKW = &X00000000 THEN CLS 'If no windows are
open, clear the screen
250 IF CHECKW = &X00000000 THEN OUT 100,0 'If no
windows are open turn off the alarm
260 FLAG = CHECKW
270 GOTO 30 'Go check windows again
```

To construct the open-window detector system, all that is needed, besides the REACTS system with the octal I/O module, is some two-conductor wire and a few window-mount SPST pushbutton switches. The switches that were used on our prototype are available for as little as $1.29 from Mouser Electronics (2401 Hwy 287 North, Mansfield, Texas),[5] and they're available from many of our other advertisers as well. They are mounted in the window sill so that the switch closes when the window is shut (see Fig. 4).

An alternative to using the pushbutton switches is to use magnetic reed switches. The magnetic half of the switch is mounted in the window frame, while the reed-switch half is mounted in the window casing. When the window is shut, the magnet will pull the reed switch open or closed depending on the contacts' arrangement (see Fig. 5).

There are two ways to connect the switches to the octal I/O module: Connect only one window switch to each of the octal I/O's inputs, or connect several window switches to each input. Each method has its advantages and disadvantages. If only one window is connected to each input, it is easy to determine exactly which windows are open. However, the number of windows that can be monitored by using that method is limited to eight per I/O module because the module has eight inputs. If several window switches are connected to one input, more windows can be monitored, but you won't know which window is open. However, all the windows in one room can be connected to the same input channel; that way, an open window could be narrowed down to one room in the house.

To connect only one window switch to each input, a two-conductor wire will have to be routed from the I/O module to each window switch. One of the two leads at the REACTS end of the wire is connected to the ground slot of the input-terminal strip on the I/O module, while the other end is connected to one of the eight inputs. The leads at the window end of the wire are connected to the switch's terminals. Assuming we are

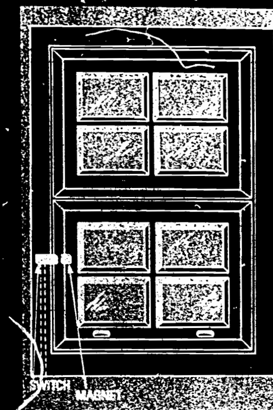


FIG. 5—A MAGNETIC REED SWITCH can also be used to make sure a window is closed. When the magnet is next to the reed switch, the switch closes. When the window is opened, the switch opens.

SOURCES

The following items are available from DataBlocks, Inc., 579 Snowhill Road, Glenwood GA. 30428. Or call (800) 652-1336; in Georgia call (912)-568-7101.

- DP-OCT: Design package with schematic and instructions; $10.00.
- PC-OCT: PC board for the octal I/O, including design package; $37.00.
- REC-OCT: Complete set of parts including PC board, IC's, terminal strips, etc., but excluding relay option listed below; $109.00.
- REC-REL: OCTAL I/O relay option includes eight, 6-amp relays and terminal strips; $42.00.
- REC-CASE and PANEL/REC-OCTAL: Aluminum case assembly including front and rear panels; $19.50.
- REC-KEY: 12-button keypad with scan-code decoding program; $9.00
- VAC-PROG: vacation home-control program; free with SASE.
- Other REACTS products are available; contact DataBlocks directly for information and pricing.

Please include $5.00 postage for any order of $37.00 or less and $10.00 postage for any order over $37.00. Georgia residents must add appropriate sales tax.

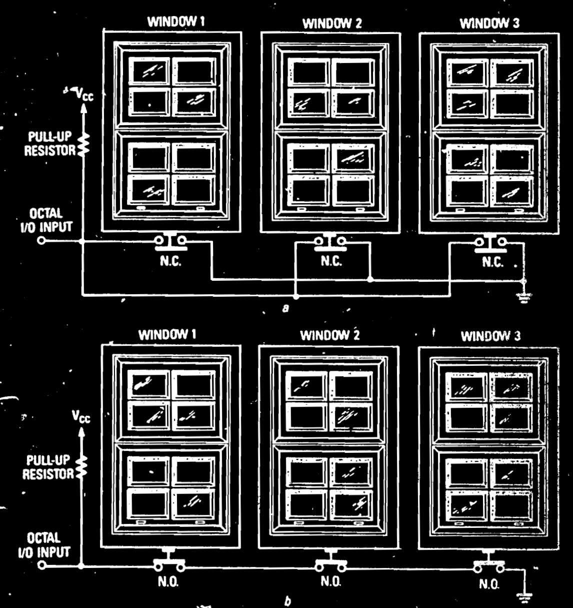


FIG. 6—SEVERAL WINDOWS CAN BE MONITORED using normally-closed switches in parallel (*a*) or using normally-open switches in series (*b*).

using normally-open switches. When the window switch is open, the input will be a logic 1 because all the inputs of the module are normally pulled high, and when the window is shut, the input will be a logic 0. If normally-closed switches are used, an open window will be a logic 0, and a closed window will be a logic 1.

The way in which the switches are connected to an input channel depends on the type of switches that are used. Normally-closed switches should be connected in parallel (see Fig. 6-*a*) so that when all of the windows are down, the switches are held open, resulting in a logic-1 input. If any of the windows are opened, the corresponding switch closes, causing a logic 0 to be read at the input.

The normally-open switches should be connected in series (see Fig. 6-*b*). When the windows are down, all of the switches are held closed, placing a logic 0 at the input. When one of the windows is opened, the corresponding switch is opened; a logic 1 is read at the input.

Listing 3 is a sample program that would control the open-window detector. For that example, we will assume the following:

- The octal I/O module is addressed at I/O port 100.
- Channel 0 (same as bit 0) of the octal I/O has a SPST switch connected to it that is used to enable and disable the alarm.
- The window switches are connected to channels 1–7.
- The alarm is connected to the channel-0 output.
- Normally open switches are being used.

The program could easily be changed to work in a system where several windows were connected to each channel. That could be accomplished by changing the print out statements to something like:

110 IF WIND1 0 THEN PRINT "A WINDOW IN ROOM #1 IS OPEN"

The program could also be made more personalized by printing: "A WINDOW IN THE UPSTAIRS BEDROOM IS OPEN"

Lawn watering system

Besides controlling and/or monitoring devices within the home, REACTS can also be used for controlling devices outside the house. Specifically, the relay outputs of the octal I/O can be connected to electrically-actuated water valves so that the system can be used to control a lawn watering system. Indeed, by using the system's real-time clock, it can provide an extremely accurate means of cycling sprinklers on or off at preprogrammed times during the day.

The real-time clock has fourteen registers which contain the tenths of seconds, units of seconds, tens of seconds, units of minutes, tens of minutes, etc., all the way up to tens of years. By using the real-time clock, the system can be programmed to go through a watering cycle daily, every other day, every third day, or any other schedule you like.

In most cases, turning the valves on or off during the watering cycle could be done on a hourly basis. Because of that, the amount of program time needed to control the system would be minimal. A statement that checks the units of hours from the real-time clock could be placed in the main body of the program. When the units change, say an hour has passed, the processor would then go into a subroutine to determine if it was time to turn on or off any of the water valves. Of course it probably would also be desirable to have a set of switches to manually turn the water valves on or off. Switches could be wired between the transformer and each relay to control the power to the relay, or if the inputs to the octal I/O were not being used, the switches could be wired to them. Each input would control an output.

Besides the REACTS system and the octal I/O module, some other parts that you'll need to construct the watering system are the electrically-actuated water valves, wiring, and a step-down transformer. Most standard electrically-actuated water valves that you will find use a solenoid that operates on 24-volts AC to open and close the valve.

To provide the power for those valves, you'll need a 117 to 24 volt step-down transformer, available in wall-mount packages. The required power rating of the transformer will depend on the power needed to acti-

continued on page 98

BUILD TH[IS] RADIA[TION] MON[ITOR]

We're always surrounded by ... we'll show you where it comes ... you, and how to use the Rada[lert] ... conditions.

JOE JAFFE, DAN SY...

HUMAN BEINGS HAVE BEEN EXPOSED TO naturally occurring ionizing radiation for millions of years because nuclear reactions take place on our sun and on other stars continuously. Their emitted radiation travels through space, and a small fraction reaches the earth. Natural sources of ionizing radiation also exist in the ground, the most familiar and most common ground-source being uranium.

Last month we showed you how to build a nuclear radiation monitor—the Radalert. Now we'll show you how to use it, and how to interpret its readings.

Ionizing radiation

Ionizing radiation is radiation that has the ability to remove electrons (the process of ionization) when it strikes or passes through an electrically neutral atom. It was first discovered about 100 years ago and given

gain. With an input of 300 mV, however, the OTA's output is enough to generate a significant I_D current, and the circuit's negative feedback automatically reduces the output level to 3.6 volts p-p, giving an overall gain of

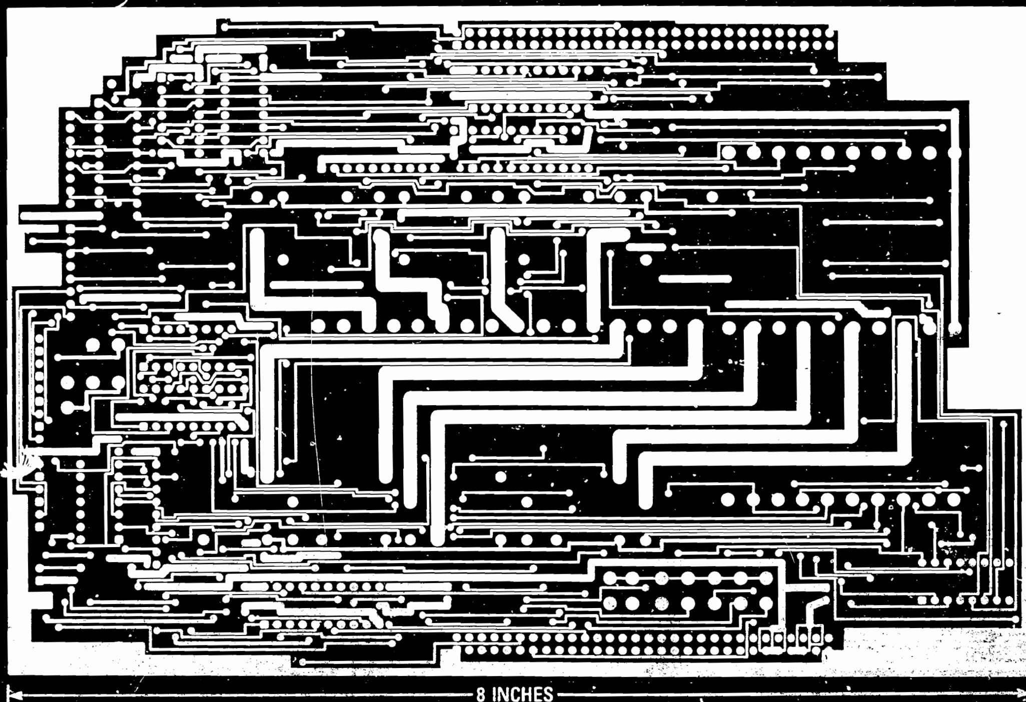
falls to 2 (an output of 6 volts p-p). The circuit thus has a 20:1 signal compression over that range.

Voltage-controlled resistors

One unusual application of the

Resistor (VCR), using the circuit shown in Fig. 10. The basic theory is as follows: An AC signal applied to the R_X terminals feeds into the inverting terminal of the OTA via C1, the output-buffer transistor, and the R5...

PC SERVICE



THIS IS THE PATTERN FOR THE solder side of the REACTS I/O board.

the one that's the most understandable is to simply decode the address bus. Whenever there's a high on any address line from A11 to A15, we're talking to RAM. We select either the RAM or the EPROM using OR gates, as shown in Fig. 2.

The 74HCT32, (or any quad OR gate), watches the upper 5 address lines and controls the enable pins of the EPROM. The only way the output of the final OR gate can be

should be tied to the BUSAK line since both are active low. That means that the Z80 would be able to control access to the external system, and allow it only when it is safe to do so.

DMA is a terrific way to put our memory system to work, but there are restrictions. Remember that the Z80 is taking care of the refresh operation for us, and while DMA is going on, the whole refresh operation is being put on the back burner. If it takes too long, you can

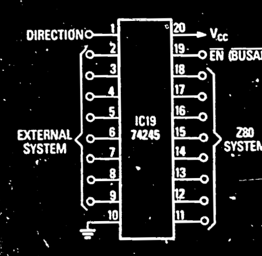
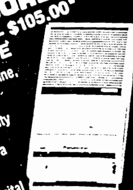


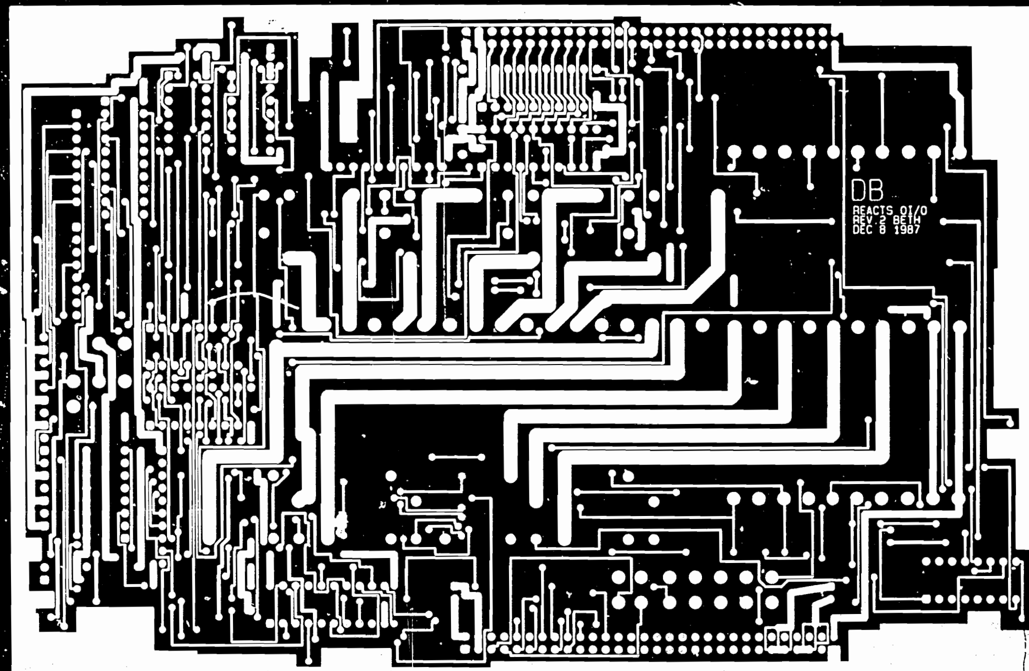
FIG. 3



ISS 7 JL 1988

UMI

PC SERVICE



8 INCHES

THE PATTERN for the component side of the REACTS I/O board is shown here.

VOL 59 ISS

ANTIQUE RADIOS

Antique radios for antique autos

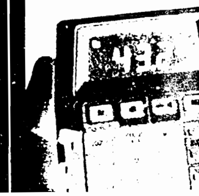
RICHARD D. FITCH

tion. Different types of magnetic and dynamic reproducers were used in those early autos. The field coil of an electro-dynamic reproducer to be used in an automobile had to be specially wound to operate from a six-volt battery, which, because of a large power drain, was not a particularly desirable situation. An electro-dynamic loudspeaker, however, can usually be replaced by a regular speaker, which is a big plus for the auto's storage battery.

There were many efforts to come up with a suitable aerial. Because extreme height was imprac-

tems would drown out your radio. Also, the high-tension circuit wasn't the only source of interference. Electrical disturbances from the generator, and even the wheels, caused radio interference. Putting a condenser across the generator brushes and a grounding spring between the axle and the wheel helped to solve those problems.

Moving out of the 1920's, the vibrator power supply became popular. There were two basic types of vibrators that were used in auto-radio supplies: The non-synchronous, and the synchronous.



REACTS

continued from page 50

vate the water valves, and how many valves will be activated simultaneously. When you are calculating the power that is needed, remember to consider the amount of current that is used by the water valve's solenoid when it is first activated (the inrush current), which will be somewhat higher than the holding current of the solenoid.

In most cases, each of the octal I/O relays would activate only one valve. Therefore the relays, which are rated at 6 amps, will have no trouble at all, as the average consumer water valve requires less than 1 amp. Be sure to protect the transformer against shorts and overloads with the proper size fuse.

Also, follow the water-valve manufacturer's guidelines pertaining to the minimum diameter of the wire to be used, and their recommendations on waterproofing the wire and the connections. Finally, be sure to adhere to any local building codes that may apply to the installation. It's worth it in the long run.

For information on water valves, sprinkler heads, etc., contact your local lawn and garden supply store. Two major manufacturers of consumer sprinkler-system products are Toro and Rainbird. You should be able to find them, and many other manufacturers of such equipment, in the yellow pages.

In conclusion

By now, we hope that the wheels in your head have started turning as to what the REACTS modular system can do for you. It is no secret that computers are now being used for more than just number crunching, word processing, drafting, etc. The system we have been showing you makes computerized home control affordable. It also bridges the gap between process controllers and personal computers.

In the articles to come, we will discuss and build more of the process-control modules. We will also construct modules that will enhance the system, such as the battery-backed power supply and the CRT controller/printer interface. We hope you will stay with us. **R-E**

LC METER

continued from page 45

and work with a simple binary counter scheme composed of inexpensive CMOS chips, required an Apple II program that ran for almost 4¼ hours.

The program that calculated all the required function addresses for the EPROM for each of the approximately 15,000 solutions to the equation ran for another 2½ hours. And after all the time spent calculating the EPROM data, the builder would need access to an EPROM programmer. To simplify everything, a programmed EPROM is available from the source listed in the Parts List.

Just in case you decide to test the EPROM data against your own calculations, bear in mind that the EPROM data have been compensated for the parasitic capacitive and inductance values that exists across the instrument's input terminals.

That's it for the theory. Next month we'll build and align the LC meter so that you can begin using it in your lab. **R-E**

GPIB

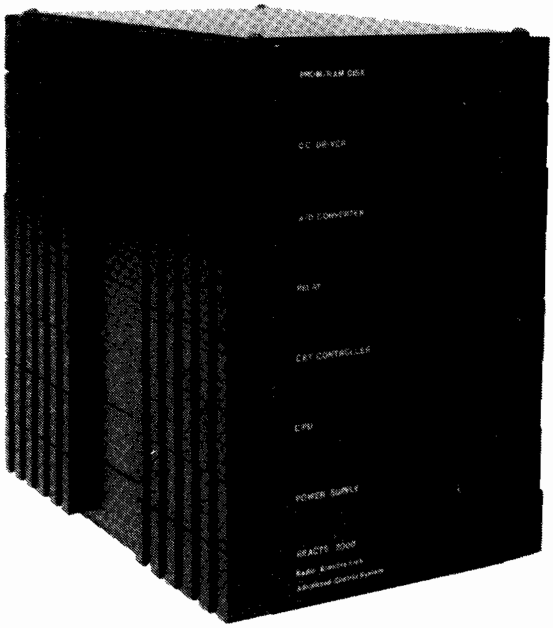
continued from page 60

There are system and measuring-related constraints that also affect the data-transfer rate. Examples of those are high resolution (5 1/2 to 6 1/2 digit) voltmeters that use the very slow integrating A-to-D (analog-to-digital) techniques, precision low frequency counting, and narrow band spectrum analysis. Those slow real-time measuring processes were fine when a human could wait 5 or 10 seconds for a measurement to come up; however, with automated test instruments that make measurements within the millisecond range, that time-frame is no longer acceptable. As a result, the test-equipment industry is converting to high-speed sampling techniques, burst measurements, smart peripherals, block-memory transfers, direct-memory accesses, and other such methods that allow the data throughput rate to be bound only by the transfer-rate of the computer-interface combination.

When we continue, we'll look at GPIB accessories and how to analyze and troubleshoot the bus. **R-E**

Build REACTS:

THE RADIO-ELECTRONICS ADVANCED CONTROL SYSTEM



MICHAEL A. TUCKER

Part 7 IN ORDER to write programs and develop practical systems using REACTS, some type of terminal is required as part of the development system. Many end-user applications also require a terminal interface for normal operation. REACTS allows you to use any one of three devices for the terminal. The first choice is that you use a standard CRT terminal. The second alternative is to use an IBM PC as a terminal, and the third choice is to build a general-purpose CRT controller/keyboard interface module using the information supplied in this article.

The interface has 2 microprocessors; one is a general-purpose Z-80, and the other is a special-purpose CRT controller. The design is flexible enough to allow you to create a custom terminal by simply changing the on-board PROM. An added feature on the module is a Centronics-type parallel-printer interface, for an IBM PC compatible printer.

Standard CRT terminal

Over the past decade, computer terminals have come down in price (a quality terminal can be purchased for less than $400.00). A number of different terminals have been tested and found to be satisfactory. There are also a number of used terminals on the market that can be purchased for next to nothing. Just be sure that the one you buy is compatible with a standard RS-232 interface, and try not to buy any terminal that doesn't have its technical manual.

The following parameters and their settings were used with REACTS and the Qume 101 terminal. They will give some idea where to start even if you use a different monitor.

- EMULATION—QVT101C
- PARITY—OFF
- DATA BITS—8
- STOP BITS—1
- BIT8—0
- FDX/HDX (Full or Half Duplex)—FDX
- CHAR/LINE/BLCK—CHAR
- ON LINE/LOCAL—ON LINE
- BAUD RATE—9600

Your PC as a terminal

The **Radio-Electronics** bulletin board (516-293-2283, 300/1200 baud, 8N1) contains the software necessary to convert your PC into a terminal for controlling REACTS. That software is also available on a floppy disk with a manual from the source listed elsewhere in this article. The software package from that source contains a file-transfer program on a PROM, which, in addition to making your PC act like a "dumb" terminal, allows you to use your PC disk, hard or floppy, as storage devices for your REACTS. That capability is very useful when developing software and allows you to keep your investment in the system to a minimum.

The PC-terminal software is contained in the file TERMINAL.EXE. To invoke that software, just type TERMINAL at the system's prompt. The software will automatically clear the screen. The only requirements for the software is that it must be run on an IBM PC or compatible, and REACTS be configured for a baud rate of 1200. The IBM-terminal software is designed to emulate the Televideo-912 terminal, so any software designed for the Televideo terminal should work as long as the host system is set up for 1200 baud. The relatively low baud rate is due to the speed limitations of a standard IBM PC. The REACTS will accept and transmit data at rates of at least 10 times faster than the PC when the PC is operating in an emulation mode.

The PC and its keyboard act just like a standard terminal at this point. Any operation that you would perform on a standard terminal is possible using your PC. With the PROM-based file-transfer program there is

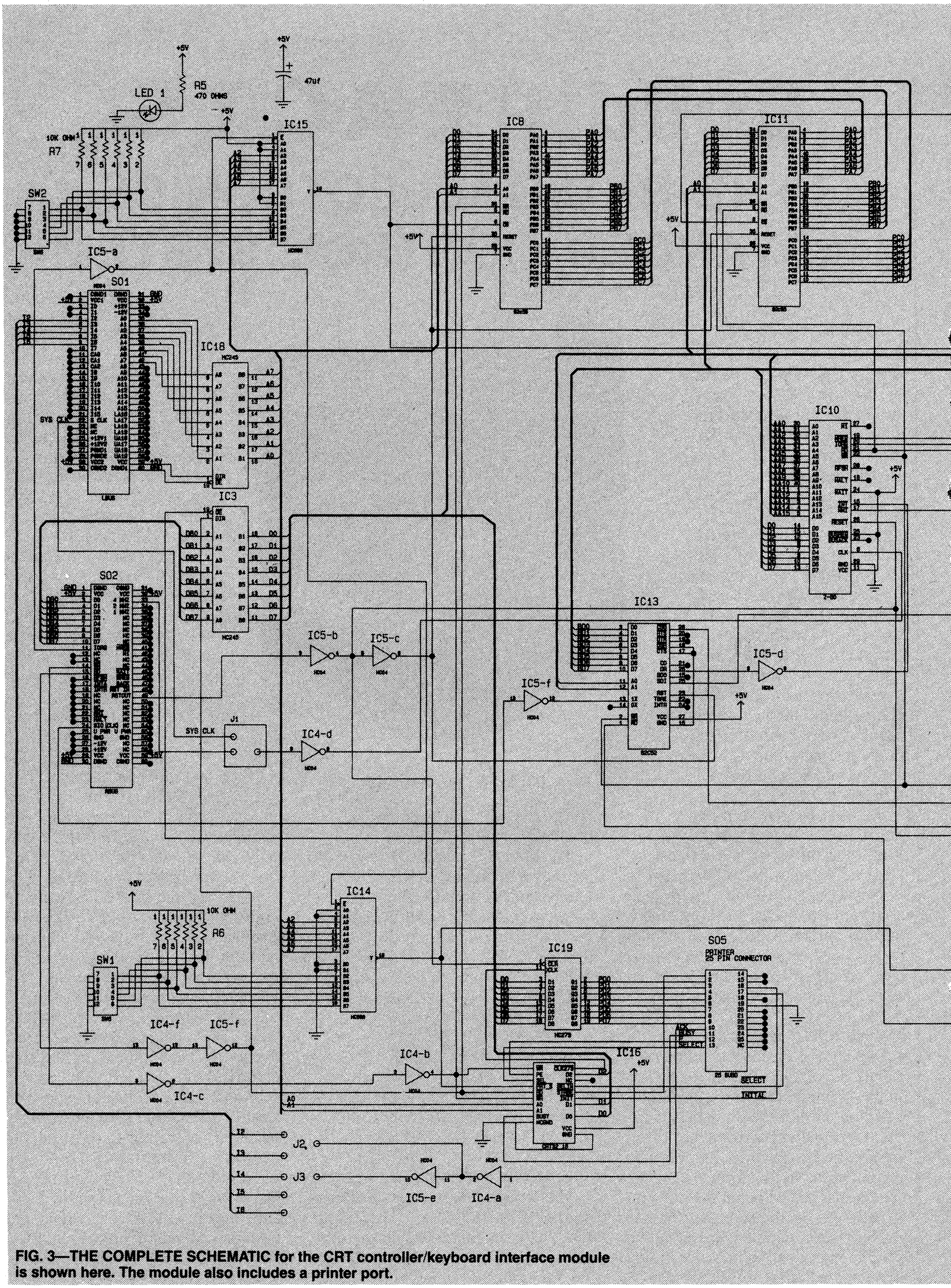
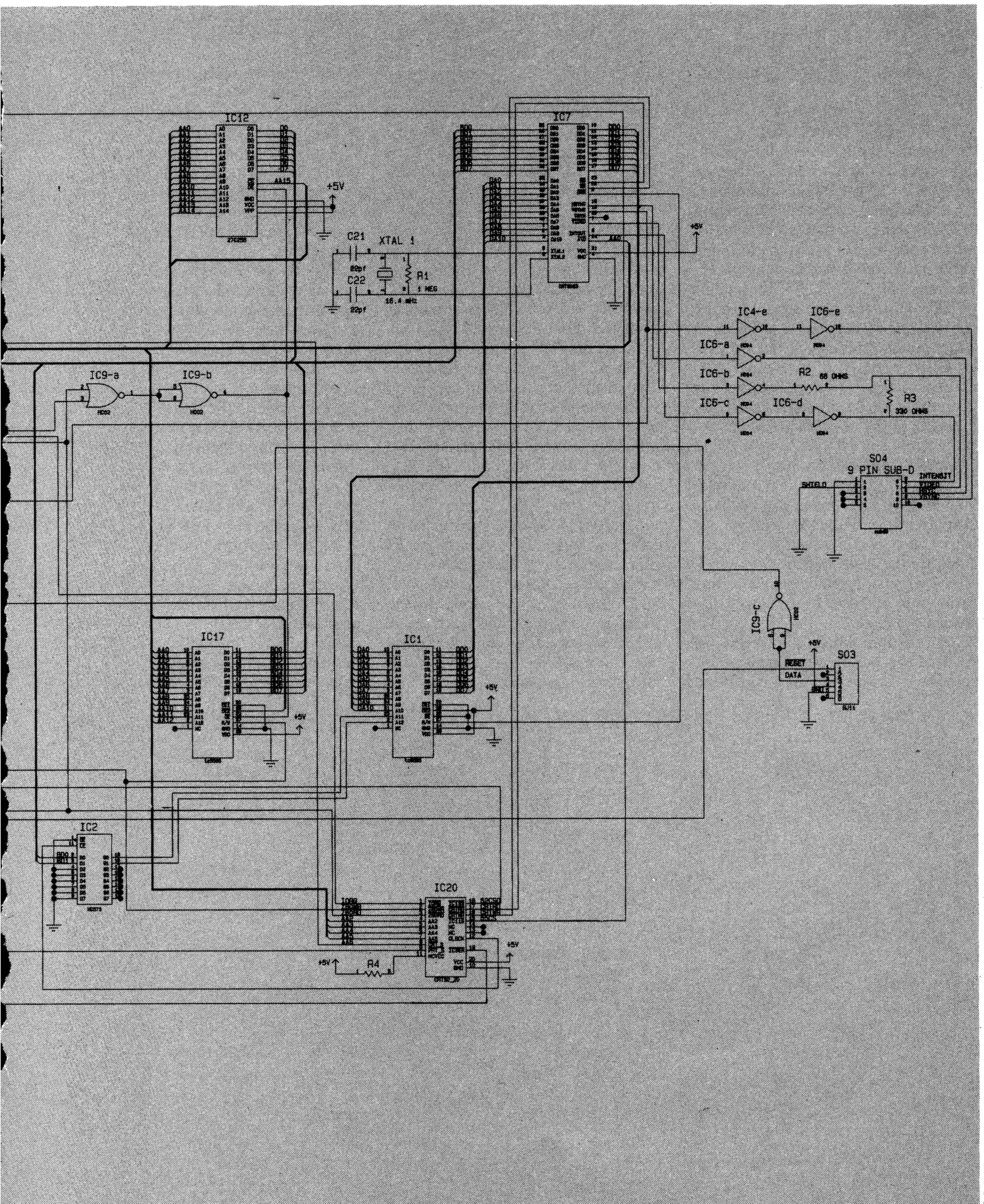


FIG. 3—THE COMPLETE SCHEMATIC for the CRT controller/keyboard interface module is shown here. The module also includes a printer port.


IC12
+5V
IC7
C21
XTAL 1
C22
R1
82pf
22pf
16.4 mHz
1 MEG
IC4-e
IC6-e
IC6-a
IC6-b
R2
68 OHMS
R3
330 OHMS
IC6-c
IC6-d
IC9-a
IC9-b
S04
9 PIN SUB-D
INTENSIT
VIDEO
SHIELD
IC9-c
S03
RESET
DATA
IC17
IC1
IC2
IC20
R4

the added capability of the PC's mass-storage potential; the terminal software can transfer your ASCII files from REACTS to the PC and vice versa.

The REACTS controller

In many applications it is desirable to have a terminal as part of the system. Also, the integral controller is the lowest-cost approach to the terminal problem. Additionally, many of the entrepreneur types will find it useful to develop special-purpose terminals using the REACTS and its CRT controller as the basis. Indeed, it is possible to emulate a multi-thousand dollar terminal—if you are willing to write the software. The monochrome monitor used by the controller can be any standard off-the-shelf IBM PC-compatible monitor.

Module considerations

As with all of the other modules we've discussed, this module will use two 60-pin module connectors. All bus signals pass from module to module through those connectors, which provide a good, sound connection and eliminate any external cabling between the modules.

A 32K ultra-violet erasable PROM is used to store the video controller's program. You can rewrite the software on the PROM and create your own custom terminal or emulate some other terminal. You would use REACTS itself as the development system to write the custom software. The program would then be burned onto a 32K × 8-bit PROM using the PROM-programmer option of the PROM/RAM module, covered in **Radio-Electronics**, May 1988.

System components

The block diagram in Fig. 1 shows the basic layout of the CRT/printer interface-module circuitry. The CRT-controller portion centers around an *E*nhanced *V*deo *T*erminal *L*ogic *C*ontroller (EVTLC), specifically the Standard Microsystem's CRT 9053. That integrated circuit will enable our video controller to display visual attribute features such as reverse video, intensity control, underline, and character blink. Also, the EVTLC is capable of producing limited graphics.

The EVTLC's built-in character set consists of the 96 standard ASCII characters and 32 special characters. Each ASCII character occupies an area of 9×12 screen dots. In the wide-graphics mode, the space that is normally taken up by one alphanumeric character is instead divided into six independently addressable segments. On a screen set up for 25 rows and 80 columns, the graphics mode would allow the independent addressing of twelve-thousand segments (80 × 25 × 6). In the thin graphics mode, the space that is normally taken up by one alphanumeric character is divided into four independently addressable segments layed out in a "cross-hair" fashion (see Fig. 2). The video controller will also allow a mixture of alphanumeric and graphic characters.

Two modes of screen scrolling are available: jump and smooth scrolling. The data on the monitor is moved up or down one row at the time in the jump-scroll mode, and one scan line at the time in the smooth-scroll mode. If desired, a non-scrolling status line can be enabled. The line will remain at the bottom of the screen at all times; that is, it stays stationary while the rest of the screen is scrolled. Other than non-scrolling, the status-line data can be manipulated in the same manner as the rest of the screen data.

As previously mentioned, screen attributes include reverse video (changing screen backgrounds), intensity control, character underline, and character blink. There are two modes of selecting the characters to be enhanced and the type of attributes to be used. In the first mode, called

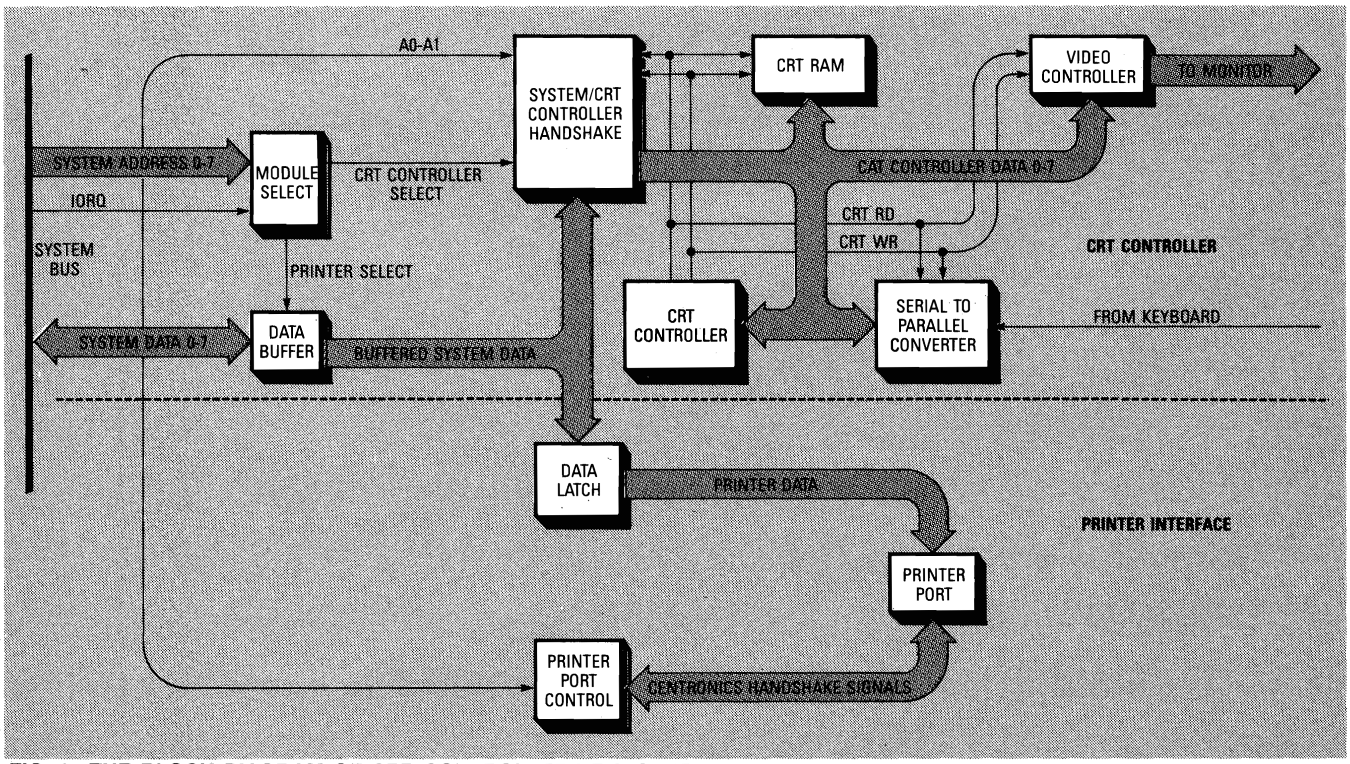


FIG. 1—THE BLOCK DIAGRAM OF CRT CONTROLLER/KEYBOARD interface module. This module allows you to interact with REACTS, and also allows REACTS to display messages on a CRT.

the 9 × 28 mode, each character to be enhanced is "tagged" with a tag bit, which is actually the most significant bit of the character byte. In that mode, only one attribute style can be enabled per screen. That is, all the "tagged" characters on any specific screen will be reverse video, intensified, underlined, or blinked. In the 9 × 56 mode, multiple attributes per screen and character are available. That is accomplished by sending an attribute byte before the character byte that will be enhanced with the desired attribute.

The fill-screen feature allows the entire screen to be filled with a given character without having to write to each display's memory address (that is ideal for quick clearing of the screen).

In addition to the EVTLC, other main components of the video controller include a Z80 microprocessor, two 8K × 8 CMOS-RAM IC's, two 82C55 programmable peripheral interfaces (PPI), an 82C52 universal asynchronous receiver/transmitter (UART), and an already mentioned 32K × 8 UVPROM.

The Z80 is the control center for the CRT controller. It manages the incoming and outgoing data between the 82C52 UART, the EVTLC, and the 82C55 PPI's. That is, it enables and/or disables the correct IC's at the correct time to maintain a smooth data flow. The Z80's clock input comes from the 8-MHz clock signal that is generated in the REACTS CPU and routed to one of the pins of the two 60-pin connectors. One of the 8K × 8 RAM chips is used by the Z80 as a scratch pad or buffer. The other provides the display memory for the EVTLC. As already mentioned, the PROM contains the video controller's control program. The 82C52 UART receives the incoming keyboard data in serial form and converts it to parallel form before transferring it to the EVTLC and/or PPI's. The PPI's act as the interface between the video controller and the CPU; that is, all data being passed between the CPU and the video controller pass through those IC's. By checking certain handshaking signals of those IC's, the CPU determines whether the video controller is ready to send data to it. Likewise, the CRT controller can determine when the CPU is sending data to it.

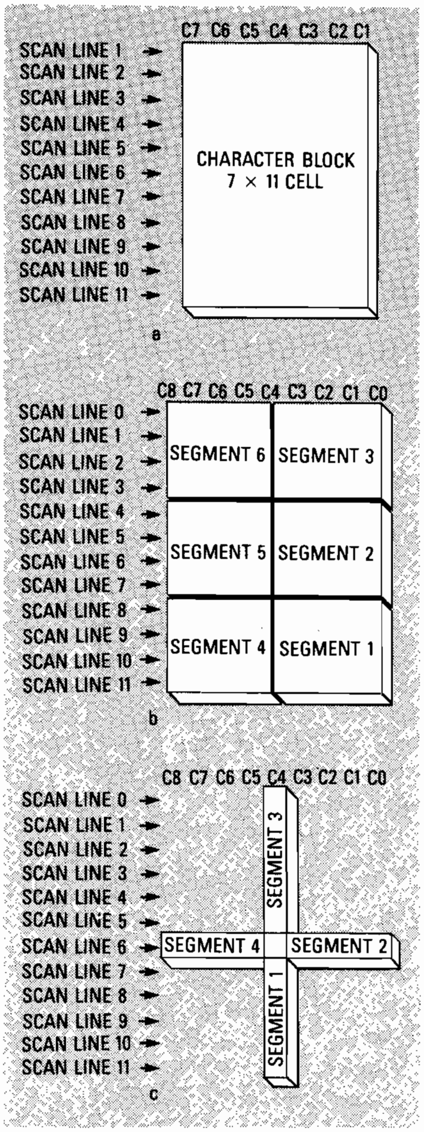


FIG. 2—The EVTLC's BUILT-IN CHARACTER SET consists of the 96 standard ASCII characters and 32 special characters. Depending on what graphics mode is being used, the screen space is divided as shown in (*a*) for the alphanumeric mode, (*b*) for the wide-graphics mode, and (*c*) for the thin-graphics mode.

Controlling the EVTLC

The EVTLC is connected to two data buses; one passes data between the Z80 and the EVTLC, and the other between the display memory and the EVTLC. That way we know that all the data we send or receive from the display memory passes through the EVTLC. The EVTLC contains several 8-bit software-programmable data registers which select the desired screen attributes, move and keep track of the screen cursor, transfer data, and select operating modes.

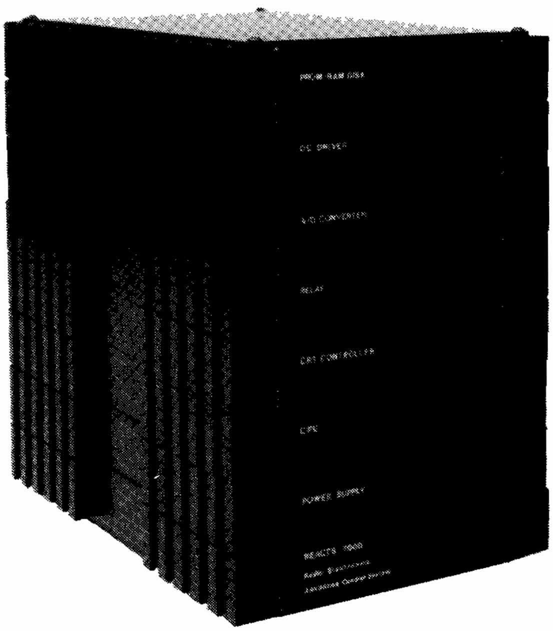
The EVTLC's programmable data registers are selected indirectly by the address register, which is selected when the input on the A/D pin of the EVTLC is high (see Fig. 3) and a write occurs. Data is sent to and/or read from the data registers when the input on the A/D pin is low. That is, first the desired register is selected by raising the A/D input and writing the correct register address in binary form, then the data is written to or read from the register with the A/D input lowered. When the A/D input is raised and a read is done, the status register's contents can be obtained. Bit seven of the status register is used to synchronize data transfers between the Z80 and the EVTLC. When bit seven is low, the EVTLC is busy and will not receive or send any data. Only when bit seven is high, is the EVTLC ready to send or receive data.

The data registers that are indirectly selected via the address register are: the top-of-screen address, cursor-low, cursor-high, fill-address, screen-attribute data, mode-1, mode-2, and character registers. We will give the correct address of each register and briefly discuss their functions. Unfortunately, that will have to wait unitl next month. We'll continue then with a discussion of the registers and then go on to building and programming the terminal interface. **R-E**

Build REACTS:

THE RADIO-ELECTRONICS ADVANCED CONTROL SYSTEM

This month we're going to show you how to build and program the terminal interface for your REACTS system.



MICHAEL A. TUCKER

Part 8 LAST MONTH we didn't get a chance to discuss the data registers. So, we'll start with that, and then build the interface.

By writing an eight (00001000) to the address register, the top-of-screen register is selected. That register contains the RAM address of the first character displayed at the top of the video screen.

The cursor-low register is addressed by writing a nine (00001001) to the address register. The cursor-low register holds the lower eight bits of the 11-bit RAM-cursor address, and the upper three bits of the cursor address are in the cursor high register.

As was mentioned, the upper three bits of the RAM-cursor address are located in the cursor-high register. The other five bits of the cursor-high register select the smooth-scroll offset value and disable/enable the non-scrolling 25th status line. That register is located at address ten (00001010).

The fill-address register is selected by writing an eleven (00001011) to the address register. That register is used to obtain the fill-screen feature. The fill-address register contains the memory-RAM address of the character following the last address to be filled. After writing to the fill-address register, the next character placed into the character register will be placed in each display-memory-RAM location between the address specified by the cursor-low and cursor-high register (the cursor address), and the address preceding the one in the fill-address register.

The screen attribute data register is at address twelve (00001100). The most-significant bit (bit 7) of the register determines whether the EVTLC is in the 9×28-graphics or 9×28-alphanumerics mode. If the alphanumerics mode is enabled, the other seven bits of the register will select the type of attribute to be exhibited by the "tagged" screen characters. Remember that in the 9×28 mode, only one attribute per screen is possible. If the EVTLC is in the graphics mode, no attributes are permitted; however, a combination of graphics and alphanumeric characters can be displayed on a single screen. In the graphics mode, a character is determined to be a graphics or alphanumerics character by the most-significant bit of the character byte. That is the same bit that determines whether or not an alphanumeric character is "tagged" in the 9×28 alphanumeric mode.

Only the most-significant bit (bit 7) of the mode-1 register is used. It is called the auto-increment bit and it determines whether or not the display-memory character address is automatically incremented by the EVTLC after every read/write of the character register. The mode-1 register is at address fourteen (0001110).

A fifteen (00001111) output to the address register will select the EVTLC's mode-2 register. Bit 0 places the EVTLC in either the 9×28 or 9×53 mode. Bit 1 enables or disables the cursor blink, and the other bits are not used.

The last register to be discussed is at register-address thirteen (00001101), and it is called the character register. All screen characters are written to and read from the display memory via that register. It should be noted that the register is accessible when the done-bit (bit 7 of the status register) is high. Tables 1, 2, and 3 show the assignments of each bit of the character register in the various programmable modes.

Control software

If you purchase the CRT-controller kit from the source listed in this article, you will receive a PROM with a CRT-controller program already loaded. We strongly recommend that you

TABLE 1—9 × 28 GRAPHICS MODE

Character-register bit	7	6	5	4	3	2	1	0
Character	1	---------- Character Data ----------						
Thin Graphics	0	0	X	X	SEG4	SEG3	SEG2	SEG1
Wide Graphics	0	1	SEG6	SEG5	SEG4	SEG3	SEG2	SEG1

TABLE 2—9 × 28 ALPHANUMERICS MODE

Character-register bit	7	6	5	4	3	2	1	0
Character (Attr. enabled)	1	---------- Character Data ----------						
Character (No attribute)	0	---------- Character Data ----------						

TABLE 3—9 × 53 OPERATION MODE

Character-register bit	7	6	5	4	3	2	1	0
Character	0	---------- Character Data ----------						
Attribute Character	1	0	0	Blank	Blink	Int	UL	RV
Thin Graphics	1	0	1	X	SEG4	SEG3	SEG2	SEG1
Wide Graphics	1	1	SEG6	SEG5	SEG4	SEG3	SEG2	SEG1

Ul = Underline Attribute
RV = Reverse Video Attribute

start with the preprogrammed PROM even if you decide to develop your own.

The software that you can purchase works in the following manner: When a key is pressed the DR (*D*ata-*R*eady) output of the 82C52 UART, which is connected to the maskable interrupt (INT) of the Z80, goes high notifying the CRT microprocessor that it has received a character. The Z80 will read that character from the UART and determine whether the character is alphanumeric or a special function. If the character is alphanumeric, the CRT's CPU will send it to the system CPU. The system CPU, after placing the character in the file or program currently called up, will then send it back to the CRT controller to be displayed. A special function can be one that is performed by the system's CPU such as CTRL-C, ESC, and CTRL-S, or one sent to the EVTLC to disable or enable a screen attribute such as reverse video, character blink, or move the screen cursor.

Programming the CRT controller

Before programming the CRT controller, various registers of the controller's IC's must first be initialized. All but one of the IC's of the CRT controller are hardwired to the I/O ports of the onboard Z80. The only one that is not addressed by the on-board Z80 is one of the 82C55 PPI's; it is addressed by four of the I/O ports of the CPU's Z80. The following list shows the addressable CRT controller IC's and their assigned I/O ports. All addresses are in decimal form:

- 82C52 UART—Ports 64–67
- Page register—Port 68
- 82C55 PPI (addressed by the on-board Z80)—Ports 72–75
- EVTLC—Ports 76 and 77
- 82C55 PPI (addressed by the CPU's Z80)—User selectable (uses 4 ports)

Those ports are selected by the on-board CRT-controller address switches. If the operating system is used, the switches should be set to 252. The address set by the switches will be the first of four, with the others following in consecutive, incrementing order. For example, if the switches are set to address 252, the first address used by the 82C55 will be 252 and then 253, 254, and 255.

The 82C52 UART is initialized by the on-board Z80 sending a 62 to I/O-port 65, a 34 to I/O-port 67, and a 35 to I/O-port 66, in that order. I/O-port 64 is the data port, and the control program should read the byte from there when the INT (maskable interrupt) input of the Z80 goes active (high), such as when a keyboard character has been received.

The CRT controller can display a maximum of two thousand characters on the monitor at one time. Each character being displayed uses one byte of the display RAM's memory. You will notice that the display memory is an 8K × 8 RAM IC. That is four times the amount of RAM needed to display one screen. Therefore the memory-page register allows the storing of four different screens within the display RAM. That is, the display

PARTS LIST

All resistors ¼-watt, 5%, unless otherwise noted.
R1—1 megohm
R2—68 ohms
R3—330 ohms
R5—70 ohms
R6, R7—9-into-1, 10,000 ohms, SIP resistor pack
Capacitors
C1–C20—0.47 µF, 25 volts, ceramic disc
C21, C22—22 µF, 25 volts, ceramic disc
C23—47 µF, 16 volts, electrolytic
Semiconductors
IC1, IC17—8K × 8-bit CMOS static RAM
IC2—74HC573 tri-state D-Type latch
IC3, IC18—74HC245 octal tri-state transceiver
IC4–IC6—74HC04 hex inverter
IC7—9053 enhanced video terminal logic controller
IC8, IC11—82C55 programmable peripheral interface
IC9—74HC02 quad 2-input NOR gate
IC10—Z80 8-megahertz CMOS IC microprocessor
*IC12—32K × 8-bit CMOS PROM with controller software
IC13—82C52 UART
IC14, IC15—74HC688 8-bit magnitude comparator
*IC16—CRT32—16 programmable array logic (PAL)
IC19—74HC273 octal D-type flip-flops
*IC20—CRT32—20 programmable array logic (PAL)
LED1—right angle PC-mount red LED
NOTE: Components that are marked with a * are customized parts that can be obtained from the source listed in last month's article.
Other components
SO1, SO2—60-pin male and female bus-connector set
SO3—6-circuit telephone jack
SO4—9-pin female D-sub printer connector
SO5—25-pin female D-sub connector
X1—16.4 MHz crystal
SW1, SW2—6-position DIP switch
Miscellaneous: 1 PC board, extruded aluminum case, 4 14-pin IC sockets, 8 20-pin IC sockets, 4 28-pin IC sockets, 4 40-pin IC sockets

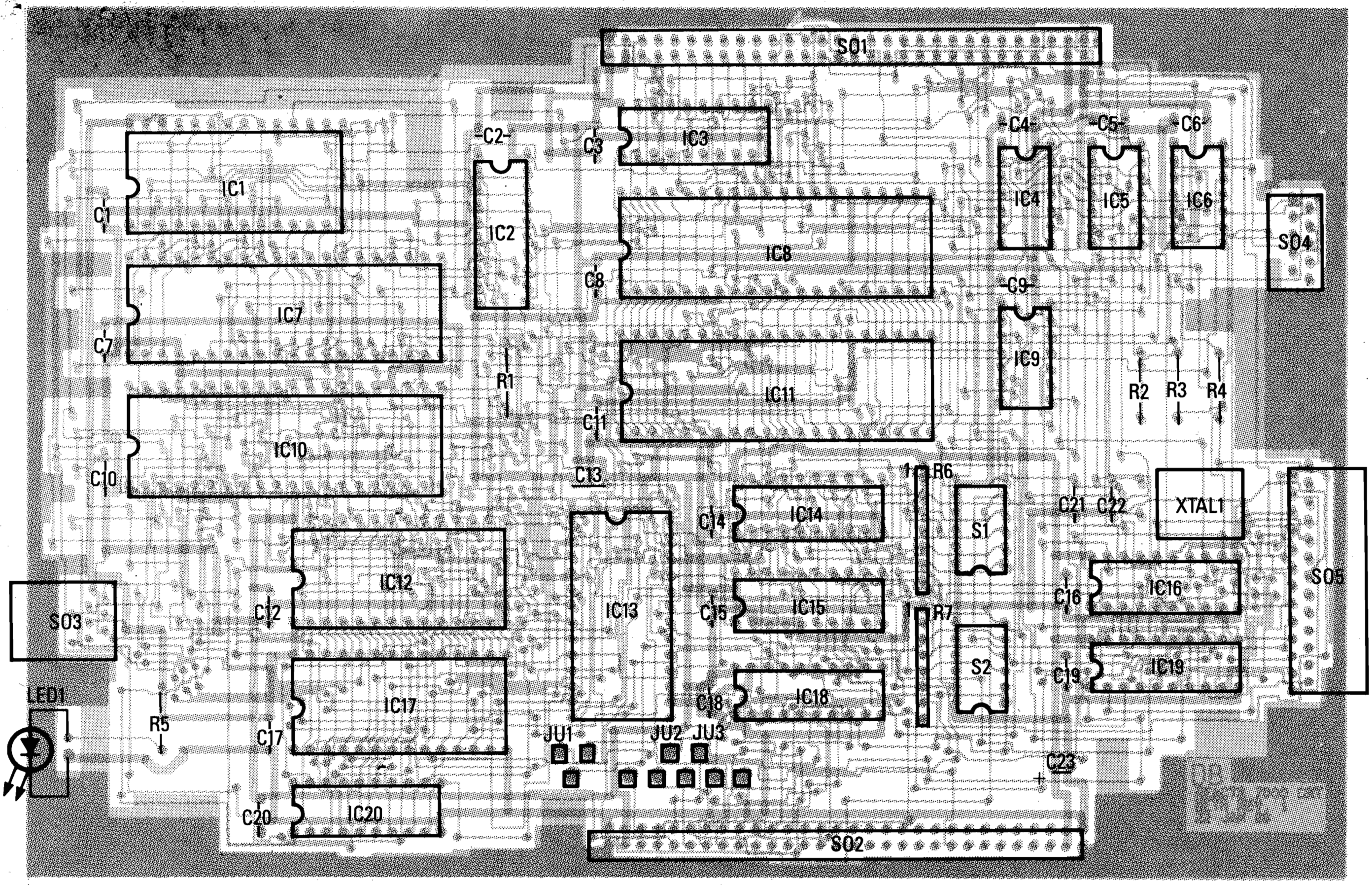


FIG. 1—PARTS-PLACEMENT DIAGRAM. The parts for the CRT controller/keyboard interface module are installed on the double-sided board as shown.

memory is divided into four pages with each page occupying 2 kilobytes. The memory page is changed by writing a 0, 1, 2, or 3 to the I/O port at address 68. Normally a 0 should be written to the display-memory paging register, to display the same screen all the time. However, when customizing your CRT controller, you may wish to use that feature in one way or another.

The 82C55 addressed by the onboard Z80 (IC11) is initialized by writing a 98 to I/O-port 75. That sets I/O-port 72 as an input port and I/O-port 73 as an output port, and also configures I/O-port 74 as a status or handshaking port.

The other 82C55 (IC8) is configured by writing a 166 to the highest I/O port used by IC8 (the I/O-port address selected by the switches + 3). The REACTS operating system, if used, will do that for you, and in addition, will handle the transfers between the PPI and the CPU. All you need to do is set the CRT controller's address switches to address 252. Remember that IC8 is controlled by the Z80 on the CPU.

The address and status registers of the EVTLC are accessed by writing to (to enter a data-register address) or reading from (to access the status register) I/O-port address 77. The EVTLC's data registers are selected by I/O-port 76.

After initialization, the CRT-controller program should be set to poll the status register, I/O-port 74, constantly checking for a character from the system CPU. If a character is received, it should be retrieved from I/O port 72, and the program should determine if it is a character to be displayed or one to change one of the screen's attributes or the EVTLC's operating mode. If an interrupt is received from the 82C52 (which means a character from the keyboard has been sent), the interrupt routine should fetch the keyboard character from I/O-port 64 and, depending on the type, send it to either the system CPU or the EVTLC. Customizing the CRT software can include using special keyboard or program inputs to change the EVTLC operating modes, change memory-display pages, perform graphics, etc.

Printer interface

The printer-interface design is straightforward. The magnitude comparator (IC14) enables the printer-interface portion of the module when the correct I/O address is on the address bus. Note that address signals A0 and A1 are not connected to IC14, and changing the least-significant switch on the addressing switch increments or decrements the first I/O port used by the printer interface by four. If we built a module that required eight I/O ports to operate, we would use the A2 signal along with the A0 and A1 signals to obtain them.

If you purchased the REACTS operating system (ARTDOS) containing the printer-interface driver, using the printer port after building the module is as simple as setting the module-addressing switches to 248 and connecting a parallel printer to the module's 25-pin port. If you didn't, you will have to design your own driver routine to use the interface.

Construction

The patterns for the double-sided PC board can be found in the PC-Service section of next month's magazine. Unfortunately, space does not permit our showing them this month. The board can also be purchased separately, as was mentioned in the "Sources" box that appeared last month.

When assembling the module, follow the parts-placement diagram shown in Fig. 4. And, we recommend that you use sockets for all the IC's. First Solder in all the components, then check all of the connections. Then double-check them.

Checkout

Before installing the CRT controller/printer module onto the REACTS CPU, make sure the system is off. If you are using the REACTS operating system, set the CRT controller's I/O-addressing switches to I/O-port 252 (all switches up). Also, set the printer-interface addressing switches to I/O-port address 248 (the rightmost switch down and the rest up). The fourth switch from the right on the row of CPU switches should be placed in the down position; that switch should be up if a terminal is being used as the console and is connected to the CPU's serial port.

Make sure that none of the pins of the CPU connectors are obstructed or bent. To install the module, align the female connectors on the bottom of the CRT/printer module with the male connectors on the top module of your system, making sure the modules are parallel to each other. Press the CRT/printer module onto the system (extreme force is not necessary). Using common sense and following proper procedures, the connectors will last indefinitely.

After the CRT/printer module has been properly installed, connect the monitor to the 9-pin D-type port on the rear of the module and the keyboard to the connector on the front. Turn on the power, and if everything is in order, the opening menu should be displayed.

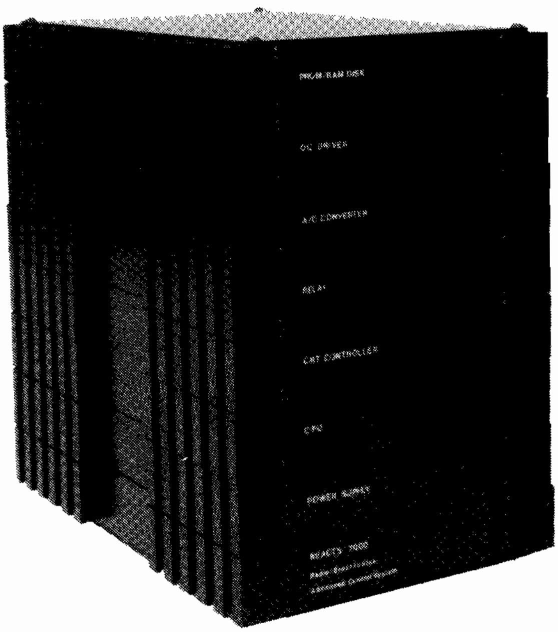
If that does not occur, check your switch settings. Also, make sure the correct switches for the CRT controller were set and not confused with the ones for the printer-interface I/O-address selection. Connect a parallel printer to the printer port to check out the printer interface. If you are using the REACTS operating system you can enter a CTRL-P, and any data entered afterwards will be printed. To get out of that mode, simply re-type CTRL-P.

In the next REACTS installment we will discuss the operation and construction of a module that will allow REACTS to remotely-control appliances, lights, etc., using the existing AC-power lines in your home, and inexpensive, easy-to-get control modules. **R-E**

Build REACTS:

THE RADIO-ELECTRONICS ADVANCED CONTROL SYSTEM

This month we discuss wireless home automation.



JIM BYBEE

LAST TIME WE BUILT THE OCTAL I/O module and covered a few of its applications. With the relay option, the octal I/O module provides eight outputs that can control up to 6-amps AC at 120 or 240 volts. While that module is excellent for controlling local devices (those near the system), controlling devices that are located further away from the system requires routing high-voltage wiring from the system to the device. We solve that problem this month when we build an X-10-compatible interface module. This new module will allow you to send control signals through the AC wiring in your house, so you can automatically turn lights, lamps, and appliances on and off from a central location.

For those of you who are unfamiliar with the X-10 system, it is a means of home control; appliances, lights, etc. The system includes a controller module that plugs into an outlet in your home. The controller module then communicates (through the AC wiring in your home) with accessory modules that are plugged into other outlets in the home. The system can then be programmed to control appliances that are plugged into those accessory modules.

The REACTS X-10 module takes the place of the original X-10 controller module so that the REACTS system can now communicate with the X-10 accessory modules. In addition, the module also contains eight LED's and eight toggle switches that can be assigned to one of the REACTS system's 256 I/O port addresses. The switches are for user-input, and the LED's are for indicating certain status information about the system; whether an X-10 module is on or off, whether one of the octal I/O's relays is closed or open, etc.

The REACTS module is connected to X-10's PL513 Power-Line Interface module using a four-conductor telephone cable with RJ14 plugs at both ends. You simply plug the PL513 into the nearest wall outlet, then plug one end of the telephone cable into the PL513 and the other end into an RJ14 jack (SO3) that is located on the back of the REACTS module (see Fig. 1).

There are two types of X-10 remote-outlet modules that are used to connect a lamp or appliance to the X-10 system. The first type is plugged into a wall outlet, and the appliance or lamp is then plugged into the module. The other type of X-10 remote-outlet module replaces the entire wall outlet, and the appliance or lamp is then plugged into that. To adapt an overhead light to work with the X-10 system, the wall switch that controls the light is replaced with an X-10 switch module. (Note that the X-10 modules come with several different power ratings, and that the amount of current drawn by the light or appliance being controlled must be considered when purchasing a module.)

After installing the modules, you must then set them to their own, unique address. Each X-10 remote module has two addressing dials. One dial (called the house-code dial) is labeled A through P, while the other dial (called the unit-code dial) is labeled 1 through 16. There are 16 unit-code addresses for each house-code address. That is, 16 of the X-10 modules can be set to the same house-code address, with the unit-code address providing the distinction between the modules.

REACTS and the X-10

The X-10 enhances the REACTS control system by simplifying the remote control of on/off devices, meaning that no routing of conductors is required. By using REACTS with the X-10 modules, a basic home-control system, complete with decision-making capabilities, can easily be installed in an already-existing home. Indeed, installing the hardware needed to automate a home could

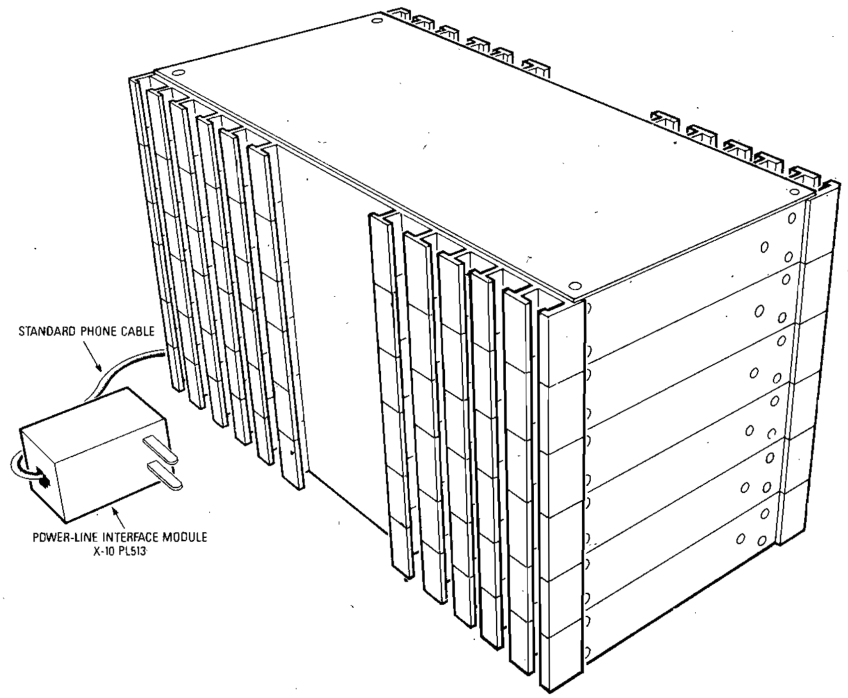


FIG. 1—THE X-10 PL513 POWERLINE INTERFACE module has an adapter that plugs directly into the wall and communicates with its corresponding receiver modules right through you home's AC wiring.

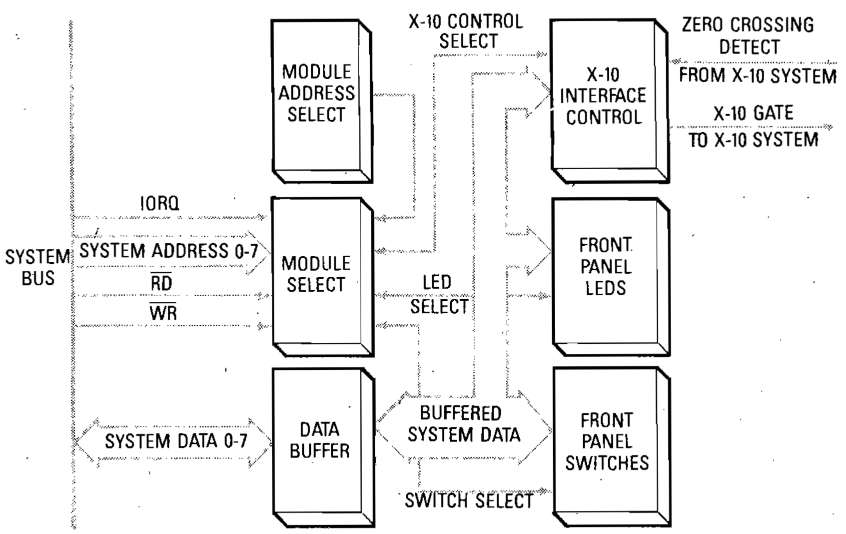


FIG. 2—THE BLOCK DIAGRAM OF THE X-10 MODULE. It uses the same type of I/O-port addressing as does other REACTS modules.

conceivably be done in one day or less. As you think of new uses for the X-10 interface, you need only purchase the required X-10 modules, and write the software (which may be the addition of only a few program lines or a short subroutine).

The X-10 interface module can communicate with up to 256 of the X-10's remote modules and can provide on/off control as well as light dimming and brightening. But while the X-10 system greatly expands the REACTS system's capabilities, it does not provide any type of feedback to the central unit, in this case, the REACTS system.

Feedback is needed in such applications as controlling your home's climate, sensing the opening and closing of windows or doors, or switching lights on and off as darkness falls or as dawn arrives. It's also needed to verify that a command is carried out. However, the eight-bit input portion of the REACTS octal I/O module is one means of adding feedback, and in future articles we will be discussing additional feedback and monitoring techniques.

Applications

The X-10 kit that is available from the source listed in the Parts List comes with an X-10 interface software driver. Using that driver, writing application programs for the X-10 system is easy. The driver operates in one of two modes. In the first mode you simply assign the string variable X-10$ to a string which contains the module's house code, its key code (or unit code), and the type of function to be performed (on, off, brighten, dim,

PARTS LIST

All resistors are ¼-watt, 5%, unless otherwise noted.

R1, R4—10,000 ohms × 9, SIP resistor pack
R2—10,000 ohms
R3—470 ohms × 9, SIP resistor pack

Capacitors

C1–C12—0.47 µF, 25 volts, ceramic disc
C13—47 µF, 10 volts, electrolytic

Semiconductors

IC1, IC11—74HC573 tri-state octal D-type latch
IC2—74HC245 tri-state octal transceiver
IC3, IC4—74HC161 synchronous binary counter
IC5–IC7—74HC163 synchronous binary counter
IC8, IC9—Programmable Array Logic IC's (custom parts)
IC10—74HC14 inverting Schmitt trigger
IC12—74HC688 8-bit magnitude comparator
LED1–LED8—Green right angle PC-mount LED

Other components

1 14-pin IC socket
5 16-pin IC sockets
5 20-pin IC sockets
1 40-pin IC socket
S1–S8—SPST toggle switch
S9—6-segment DIP switch
SO1, SO2—60-pin male and female bus-connector set
SO3—RJ14 6-pin telephone jack

Miscellaneous: PC board, hardware, solder, etc.

Note: The following items are available from DataBlocks, Inc, 579 Snowhill Road, Glenwood, GA 30428, or call (800) 652-1336; in Georgia call (912) 568-7101. A kit containing all parts and a PC board: $159.00. A PC board is available for $39.00, and the two PAL's (IC5 and IC6) are $69.00. Please add $5.00 shipping for any order that is less than $37.00, and $10.00 for any order that is more than $37.00.

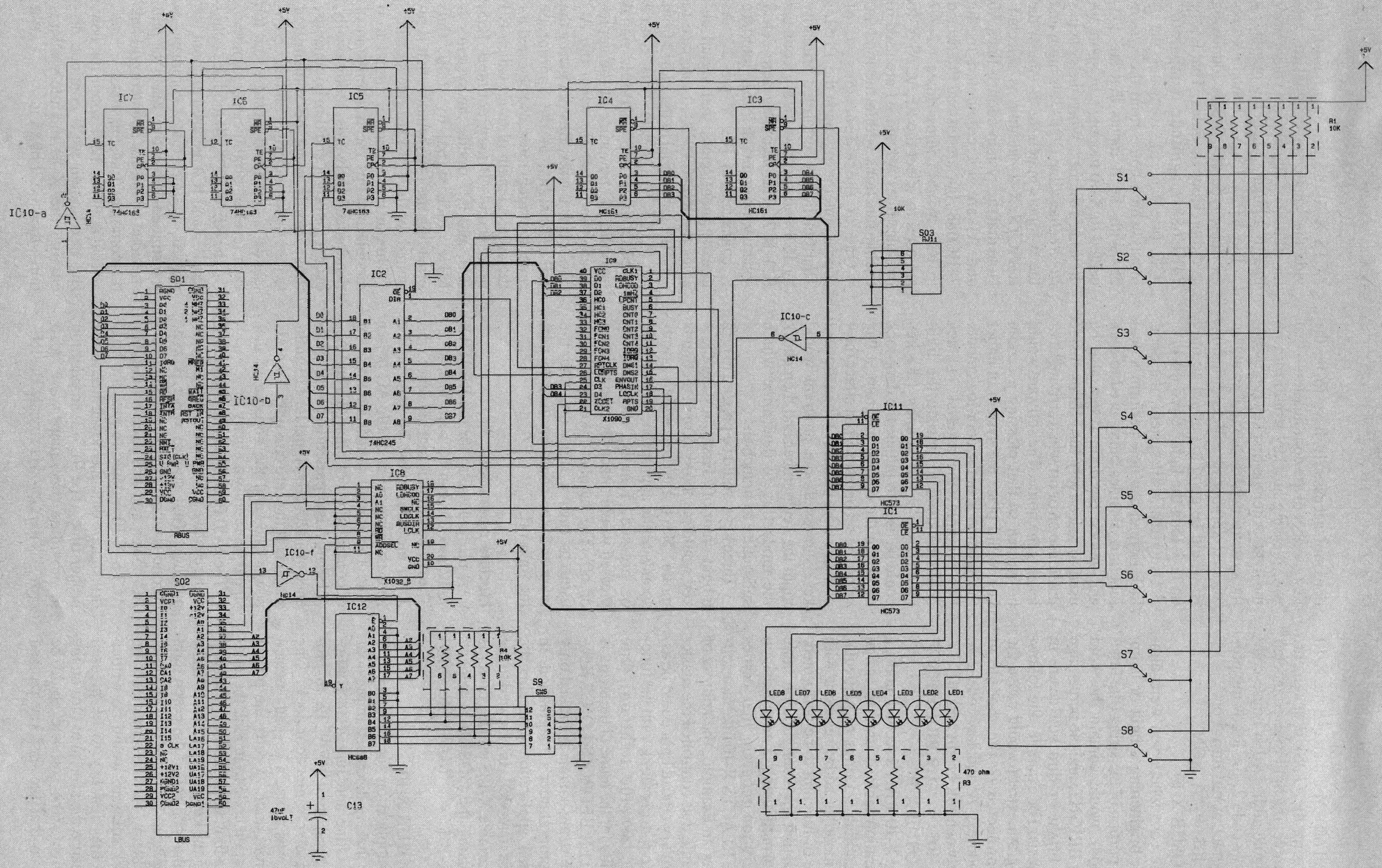


FIG. 3—COMPLETE SCHEMATIC FOR THE X-10 MODULE. The switches can be programmed so that they can manually control things that are connected to the REACTS system. The LED's can be programmed to indicate the status of various system functions.

all on, all off). You then call the X-10 driver using the GOSUB statement. For example, if a light were connected to a module addressed at house-code A and key-code 2, the following program extract would turn it on:

```
10 X-10$ = "A,2,ON"
20 GOSUB X-10
```

In the case where you were dimming or brightening a light or lamp, the last portion of the string (the function code), would contain the word DIM or BRIGHT, and the percentage of full bright or dim you desire in parenthesis. For example, the following program extract would cause the light connected to the module at house-address A and key-code 2 to burn between full bright and off:

```
10 X-10$ = "A,2,BRIGHT(50)"
20 GOSUB X-10
```

In the X-10 driver's second mode, a configuration table is used to assign user-identifiable strings to particular remote modules' addresses along with the type of function to be performed. Constructing the configuration table is performed using a menu-driven configuration program that comes with the REACTS X-10 interface software. As an example, the address of the module that controlled the master bedroom's overhead light could be assigned to a string along with the function to turn the light on. One possibility for the name of the string is "MASTER BEDROOM ON." However, the string's name is limited only by the programmer's imagination. In the actual application program, the string, as in mode 1, is assigned to the string variable X-10$ and the driver is called using the GOSUB statement:

```
10 X-10$ = "MASTER BEDROOM ON"
20 GOSUB X-10
```

One application for the octal I/O module that we discussed in previous articles (June and July 1988 **Radio-Electronics**) used REACTS to provide an electronic door lock. One method was to use a keypad, located at the front door, to enter the proper combination code to unlock the solenoid deadbolt. One problem with that is seeing the keypad in the dark. If the switch that controlled the porch light were replaced with an X-10 module, that problem could be remedied. The door-lock software would be written so that the first key to be pressed would cause the system to turn on the porch light. The software could be written so that the first key pressed does not necessarily have to be the first number of the combination—that way, pressing any key would turn on the light. After the first key is pressed, the system could then start looking for the first number of the combination. (It would be good to put a time limit on how long the light stayed on.)

Using the X-10 system, the door-lock keypad can now perform a dual function. Besides being used to lock and unlock the door, it can also be used to selectively turn lights inside the house on or off when arriving at or leaving the home. The software would be written so that pressing a certain key would place the system in either the light mode or the door-lock mode. For example, the asterisk (*) key could be used to initiate the light mode, while the pounds (#) key could be used to initiate the door-lock mode. In the control program, a flag would be set or reset when either the * or # key were pressed. When the next key is pressed, the program would input the key number and then check the mode flag to determine which subroutine to branch to.

Three keys would be pressed to turn a light on or off. First, pressing the * key would put the system in the lights mode. Next, the key which corresponds to the appropriate light would be pressed. Finally, the last key pressed would either turn the light on or off. The number-1 key could be used to turn a light on, and the number-2 key could be used to turn the light off.

Module operation

A block diagram for the REACTS X-10 module can be seen in Fig. 2. The control signals that are sent to the X-10 modules must be sent at the zero-crossing point of the AC power line. The control signals are made up of combinations of binary 1's and Ø's. A binary 1 is represented by the presence of a 120-kHz signal for 120 milliseconds at the zero crossing of the AC power line, and the absence of that signal represents a binary Ø.

The X-10 PL513 interface outputs a square wave to the REACTS X-10 module. That square wave is synchronous with the zero-crossing of the AC power, and it is used by the REACTS X-10 module to gate the control signals onto the power line at the proper time. The PL513, in addition to providing the square wave, also provides the 120-kHz signal. The REACTS X-10 module only has to provide the square-wave signal (to gate the 120-kHz signal onto the power line) if a binary 1 is being output.

The REACTS X-10 module uses the same type of I/O-port addressing as did previous modules. That is, the 74HC688 magnitude comparator, IC12 (see Fig. 3), compares the A inputs which come from the system bus with the B inputs that are connected to X-10 module's I/O-port addressing switches. When the system's *I/O ReQ*uest line (IORQ) is high, and the A inputs match the B inputs, the REACTS X-10 module is selected. Note that the system-address lines A0 and A1 are not connected to the 74HC688, but are instead connected to IC8. While address lines A2–A7 are used to select the REACTS X-10 module, lines A0 and A1 are used to enable functions within the module.

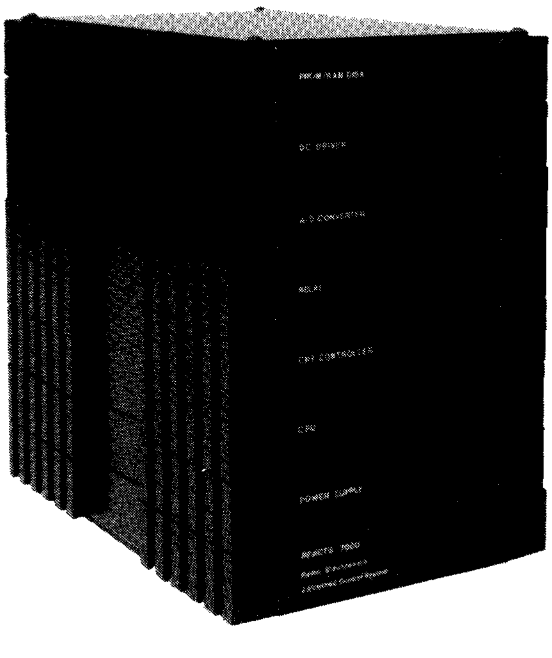
One of those functions is, of course, interfacing with the PL513. That function is performed primarily by IC5 and the five binary counters IC3–IC7. The incoming zero-detect square wave from the PL513 interface is connected to IC9 at pin 22. As already mentioned, that signal enables the REACTS X-10 interface to send the gate signal to the PL513 from pin 16 of IC9. The gate signal is used to place the PL513's 120-kHz signal onto the AC powerline at the appropriate time; that is, when the AC power is at 0 volts. Besides gating the 120-kHz signal at the appropriate time, the signal must also be gated for the correct length of time, which is 1 millisecond. Binary counters IC5, IC6, and IC7 divide the system's 1-MHz clock signal by 1000, thus providing the 1-ms window that is required for the 120-kHz gate signal.

Outputting a control signal to a module requires the transmission of two thirteen-bit words. Additionally, except for bright and dim commands, those words must be transmitted twice. As for the bright and dim commands, a command must be sent for each level of brightness or dimness. On the REACTS X-10 module, counters IC3 and IC4 are used to count the number of times that a command is transmitted. When they reach the prespecified count, the counters will halt the command transmissions.

That's all we have room for now. But next month we'll continue discussing the X-10 module's operation—then we'll build it. **R-E**

Build REACTS: THE RADIO-ELECTRONICS ADVANCED CONTROL SYSTEM

This month we show you how to add X-10 compatibility to REACTS for easy home automation.



JIM BYBEE

LAST MONTH WE BEGAN TO DIScuss how the REACTS X-10 module works, however, as is often the case, we ran out of room and were forced to end our discussion. So, this month we continue our explanation on the X-10 module's operation. We'll also talk about the software that makes our system "tick," and then show you how to build the module so that it can become a working part of your REACTS system.

The X-10 module includes eight status LED's and eight SPST input switches. The switches, through software, can be used to turn lights and/or appliances, that are connected to X-10 modules, on or off. Likewise, the status of those appliances (whether they are on or off) can be determined by the LED's. The LED's and switches are located at one of the REACTS I/O ports. The switches are read by inputting from the port and the LED's are turned on or off by outputting the correct bit combination to the port. Since the switches and LED's are independent from the X-10 functions of the module, they are not necessarily confined for use with only the X-10 module. For example, the switches could be used to activate or deactivate the relays on the octal I/O module. The following example program illustrates that. In it we will assume that the X-10's SPST switches are addressed to I/O port address 20, and the octal I/O relays at I/O port 55:

```
10 SWITCHES = INP(20)
20 OUT 55, SWITCHES
30 GOTO 10
```

That program would continuously input the binary value of the switches and then output that value to the relays on the octal I/O module. Each switch occupies one bit of the byte read from I/O port 20. If the switch is on, the corresponding bit will be a 1, and when the byte is sent to the octal I/O module (line 20), the corresponding relay will be activated.

Software for the X-10 module

We have already mentioned how easy it is to write programs for the REACTS X-10 module using the available driver software. Using that software, only two program lines are needed to send a command to an X-10 remote module, and one of them is always the same—for example: GOSUB X-10. The other program line can be written in one of two ways, depending on the program mode. In mode 1, the program line is made up of a string that contains the

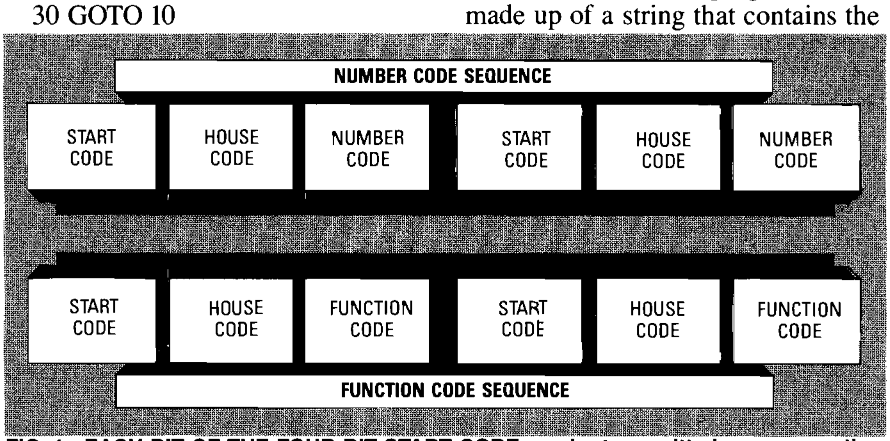


FIG. 1—EACH BIT OF THE FOUR-BIT START CODE can be transmitted on consecutive zero crossings of the AC power line. The other nine bits of each thirteen-bit word must be sent out in true form first, then in its inverse form. Each thirteen-bit word must be sent to the module at least twice.

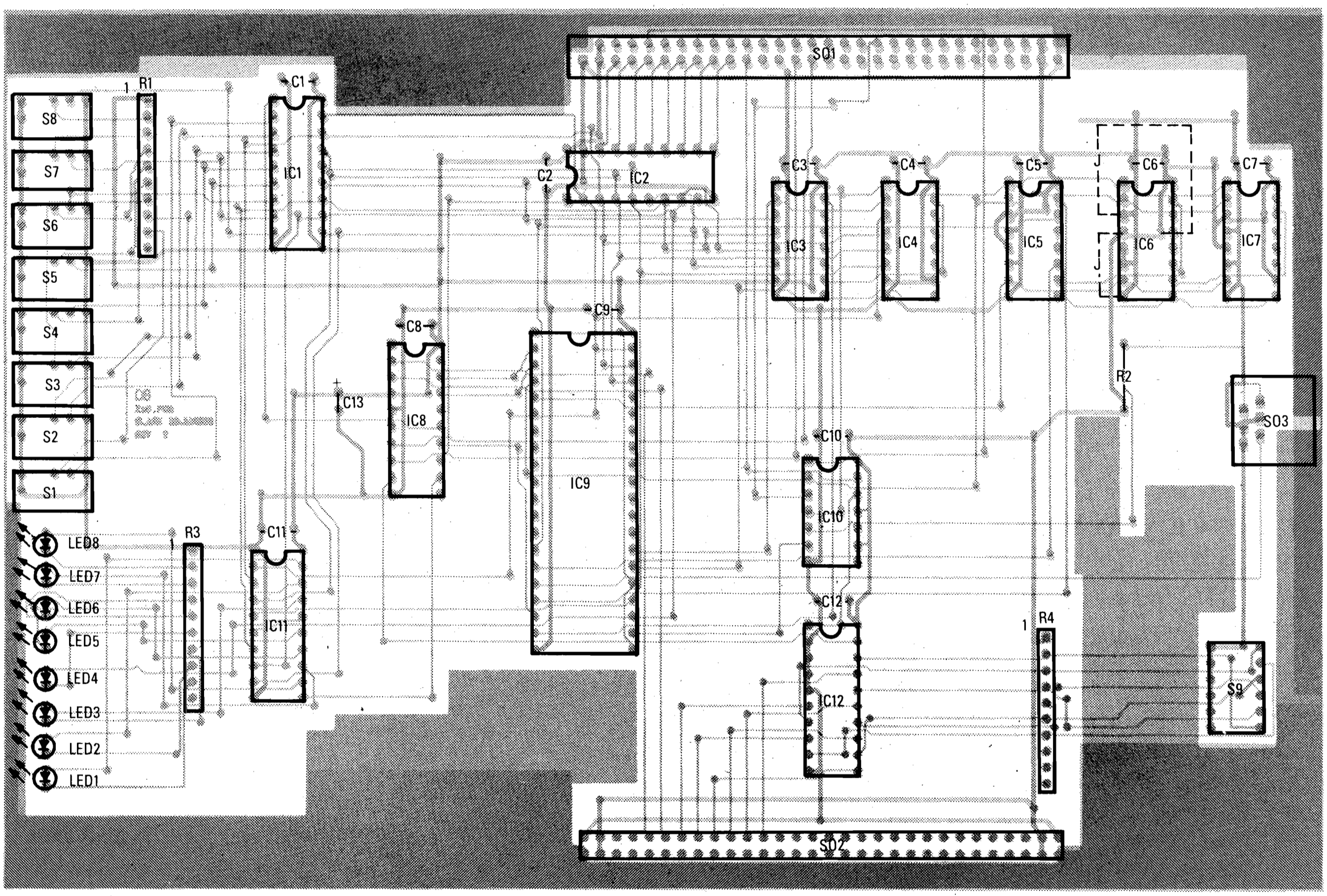


FIG. 2—PARTS-PLACEMENT DIAGRAM. Follow this when installing all parts on the X-10 board, and don't forget to use sockets for the IC's.

module's house and key codes, and the function to be performed—for example: A,2,OFF. In mode 2, the program line contains a user-defined string that represents that information—for example: MASTER BEDROOM ON.

Outputting a control signal to a single X-10 module requires the sending of two thirteen-bit data words. The first thirteen-bit word (the number code) selects the number of the module to be controlled, and the second word (the function code) selects the type of function (on, off, dim, brighten) to be performed. The first four bits of both thirteen-bit words is a start code (1110) that is always the same. The next four bits represent the house code (A through P), and the final five select either the number of the module being selected (0 through 16) or the number of the function to be performed (on, off, dim, or bright). With the all-on and all-off functions (where all of the modules are affected), only the function code is sent.

Each bit of the four-bit start code can be transmitted on consecutive zero crossings of the AC power line. However, the other nine bits of each thirteen-bit word must be sent out in true form first, then in its inverse form on the very next zero crossing (see Fig. 1). Additionally, each thirteen-bit word must be sent to the module at least twice.

As you can see, the actual transmitting of the codes to the X-10 remote modules is somewhat involved. However, the REACTS X-10 software handles all the details of transmitting to the X-10 remote modules, allowing you to concentrate on the main objectives of your control program.

Construction

Very little in the way of assembly is required for the X-10 module. You can purchase a complete kit of parts, or just a PC board with or without the PAL's. You'll also need a 4-conductor telephone cord with RJ14 plugs at both ends, and an X-10 PL513 Power Line Interface module. You can buy a PC board or else you can make one from the foil patterns in PC Service, and follow Fig. 2 for correct placement of all of the components. Just solder in all of the parts being sure to use sockets for the IC's. Then just press the appropriate IC's into their respective sockets. Figure 3 shows you the completed board.

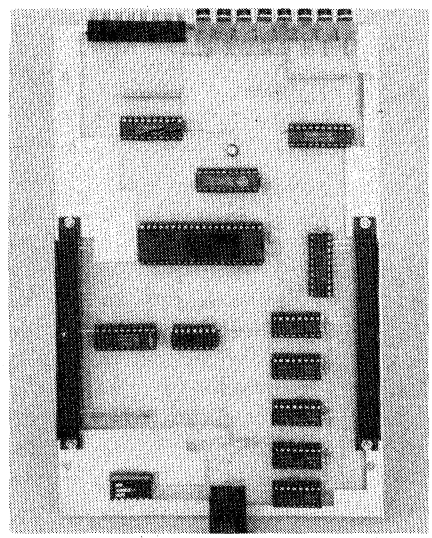
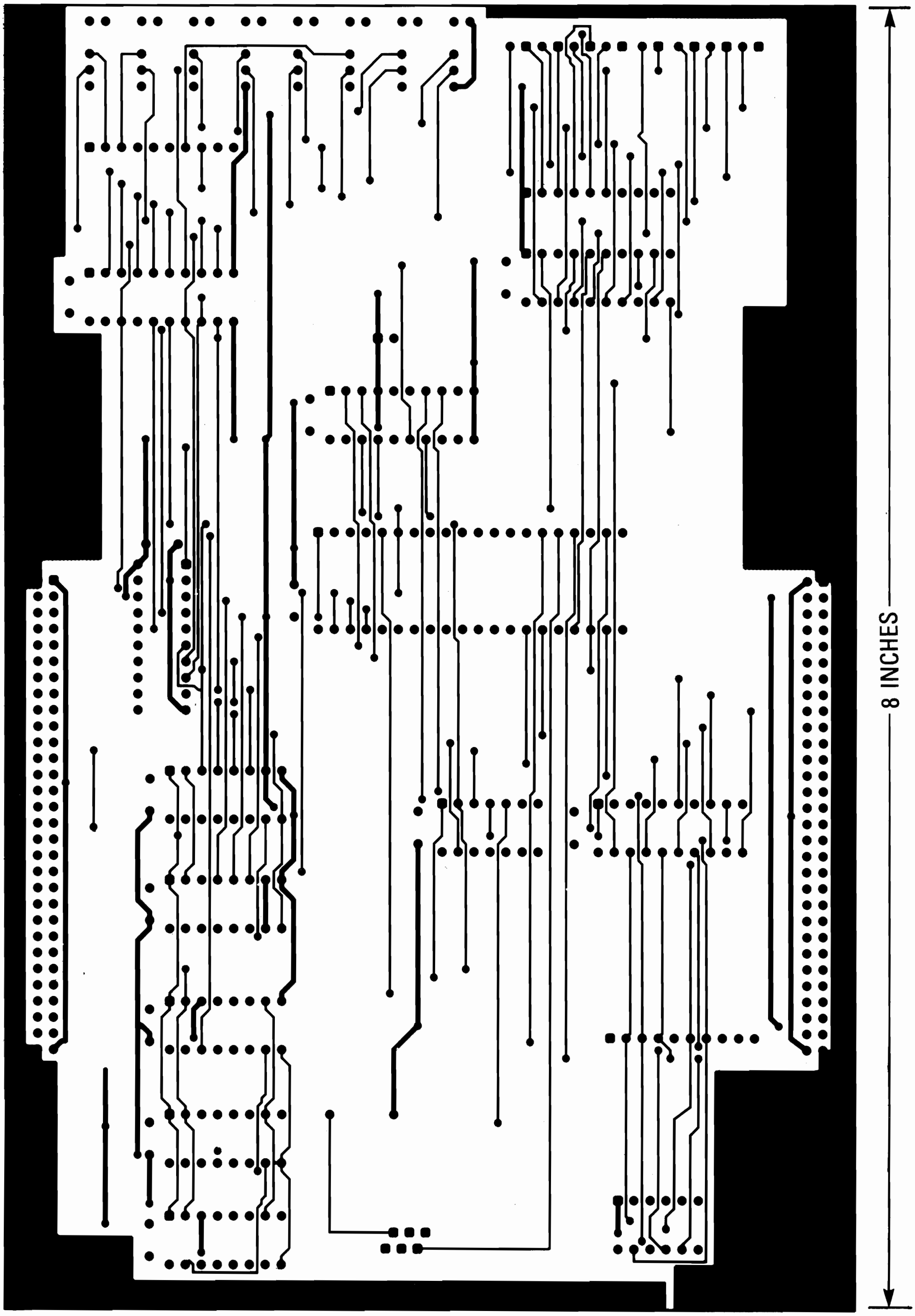


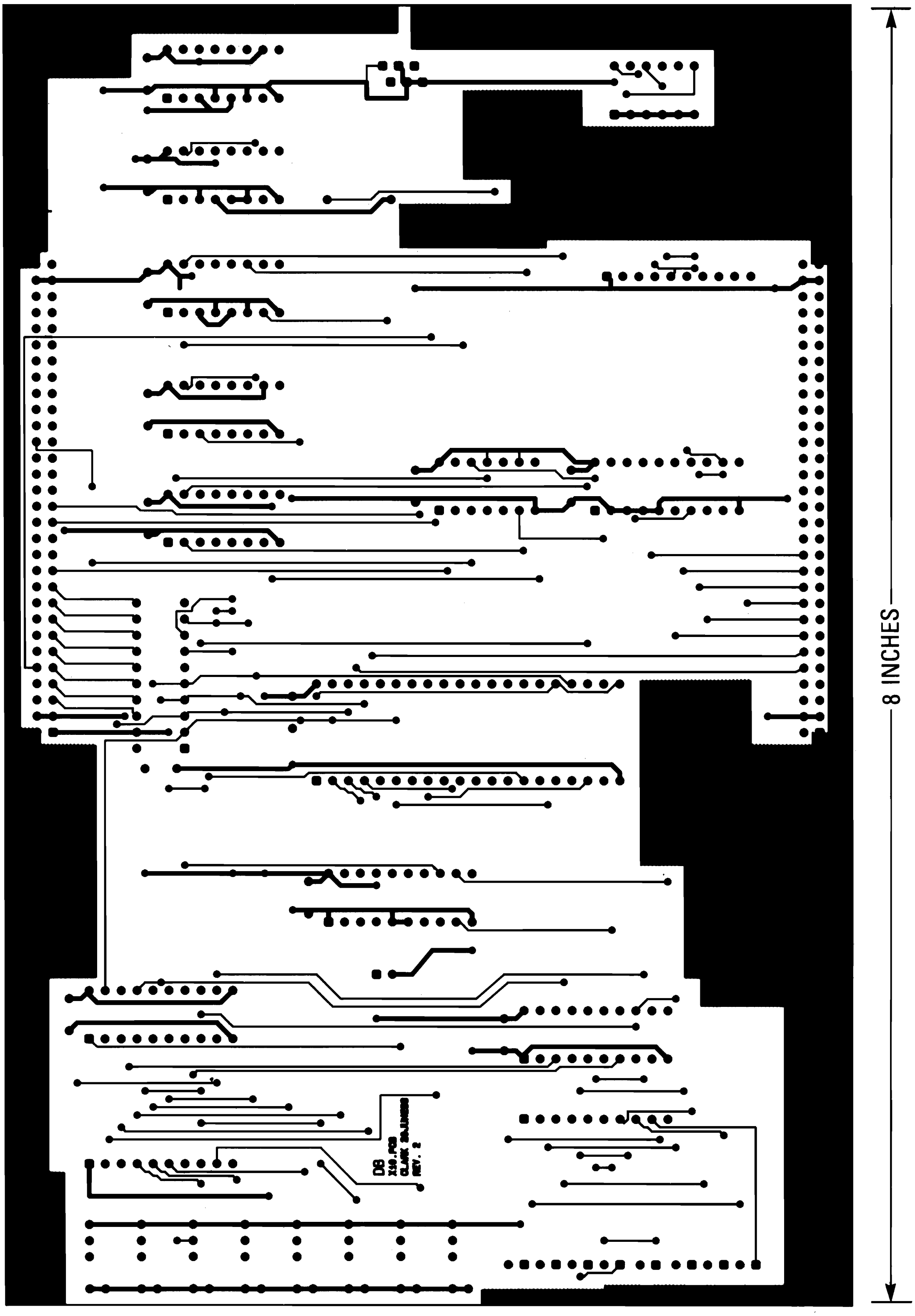
FIG.3—THE COMPLETED BOARD. Notice the eight programmable switches.

With the REACTS X-10 interface, you take a great leap toward automating your home, and you may indeed be content with the capabilities you now have. However, we will be discussing other modules in the future, including an A/D conversion block, battery backup, and more. **R-E**



HERE IS THE SOLDER SIDE of the REACTS X10 module.

PC SERVICE



HERE IS THE COMPONENT SIDE of the REACTS X10 module.