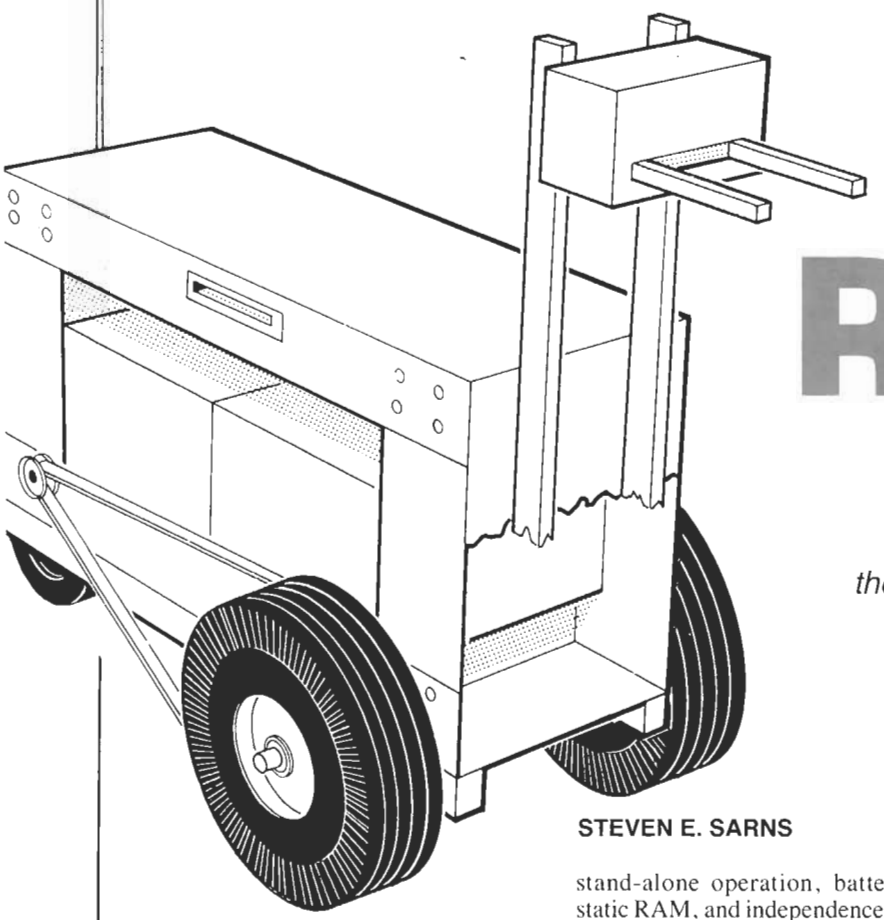
BUILD THIS



R-E ROBOT

A detailed look at the Robotic Personal Computer—the brains inside the R-E Robot.

STEVEN E. SARNS

Part 2 OUR LAST ARTICLE introduced you to the R-E Robot, and to our project to provide a sophisticated, yet low-cost, alternative to the expensive home robot. This month, we will look a little more closely at the *R*obotic *P*ersonal *C*omputer (RPC). The development of the RPC represented quite a challenge, but produced an exciting single-board personal computer that can be interfaced with a wide range of off-the-shelf or custom peripherals.

Design criteria

Most of our work to this point had used the SBC88 (a single-board controller project described in the April through June, 1984 issues of **Radio-Electronics**). However, we realized that that single-board controller with its on-board inputs and outputs was too limited to use for our robot project. That applied both to the controller's input/output capability and to the programming environment that it provided. We knew what we needed—the productive programming environment of a personal computer combined with the economy and ruggedness of a single-board controller. Typical single-board computer features that add to that ruggedness include application code in ROM, stand-alone operation, battery-backed static RAM, and independence from troublesome disk drives once the application programming is completed. However, we needed the PC bus for possible future memory and input/output expansion of the robot. With so many ideas and possibilities revolving around our robot, we did not want the hardware to limit us.

After a careful search, we found that a single-board computer that met all of our requirements was not available. Our task then was to design such a computer, and to do it on a spartan budget.

The RPC

The *R*obotic *P*ersonal *C*omputer (RPC) was designed to bridge the gap between the single-board controllers and the personal computer. Both types of machines have advantages and disadvantages. The controller is the classic solution to problems for which microprocessor control is required. However, the productive programming environment and the rapidly declining price of the personal computer are now tempting many manufacturers to incorporate complete personal computers into their products.

It takes a major commitment of money and talent to design a single-board computer for dedicated control applications. Such a project requires a development system because the completed single-board computer has no operator interface. Typically, development systems cost between $5000 and $30,000 and require engineers to design the hardware and programmers to develop the software in assembly code. Once completed, however, the manufacturer has an economical system that is custom-tailored to his application. All software is stored in ROM, the most secure and least expensive form of data storage.

The personal computer offers an alternative to the foregoing for situations where disk-based operating systems and applications programs are acceptable. Their use nearly eliminates the need for the services of design engineers, because all that may be required in the way of hardware design is a peripheral board, and then only when a standard off-the-shelf product cannot be located.

There are some disadvantages to using a personal computer in a dedicated control application. The major one is that they store their code on disk. Many dedicated control applications require that the computer exist in harsh environments where disk drives cannot function reliably. Another problem is the operator interface. The personal-computer interface has been optimized for the programmer or data-entry person. However, many control applications require highly customized operator interfaces, such as LCD or touch-

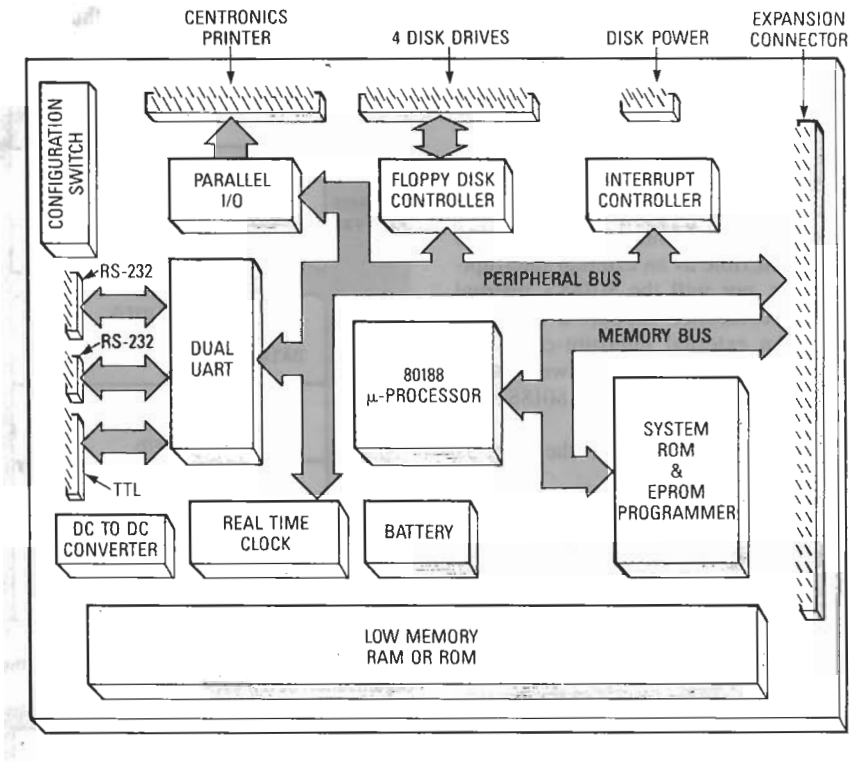


FIG. 1—BLOCK DIAGRAM OF THE RPC. All major functions of the computer are shown here.

panel displays. Finally, and most important, the end product is much more expensive than necessary.

With all that in mind, we set out to design our RPC, a single-board personal computer. Our design requirements dictated that the result would be a unique product. The major requirements were:

- The system must be its own development system (like a personal computer).
- The system must be tailored (both in hardware and in software) to execute ROM-based programs.
- The system must have a readily accessible expansion bus.
- The system must be economical at the board level, containing only those components needed to implement the design goals (like a single-board computer).

Once the design specifications were clear, we drew up some block diagrams. The trick was to achieve a balance between all the goodies that could be tied into the system, while preserving economy of board space and component cost. Before long, the design shown in Fig. 1 began to take shape. And after several months of sketching, the first schematics emerged. A complete schematic of the final version of the RPC will be shown in a future installment.

Designing the hardware

Once we knew the overall design of the RPC, we were faced the task of selecting the components. The first step was choosing the microprocessor. Given the success of the IBM *PC*, our first thought, naturally, was to use the 8088. That microprocessor is supported by the latest generation of software packages available. However, the 8088 requires a large number of support IC's. The newer 80188 is a highly integrated version of the 8088. Much of the peripheral support circuitry required by the 8088 has been included on the 80188 chip (see Fig. 2). That high degree of integration also made it easier to meet our goal of a physically small circuit board.

Although dynamic RAM is less expensive than static RAM, there are several advantages in using static RAM for the RPC. During the applications-software development process, RAM can be replaced with programmed EPROM's as the program is developed. An EPROM programmer is included on the board. That EPROM programmer also programs EEPROM's, which is handy when you consider that some applications software will probably need to be developed in the field. An on-board battery keeps the RAM powered for several months. One of the disadvantages of static RAM is the package size. Provision for special two-high RAM-stacking sockets is included. Those sockets accommodate up to 128K of RAM/EPROM memory.

Upper memory, normally the domain of the *B*asic *I*nput *O*utput *S*ystem (BIOS) and system monitor, if present, has received special attention. An entire 64K segment is allocated to 4 byte-wide sockets. The highest 16K contains the BIOS and power-on reset vector. Two more sockets contain 32K of high-level language and applications code. The fourth socket is either the EPROM programmer or contains applications code. A complete memory map is shown in Fig. 3.

The system must support a disk drive during program development for program storage and retrieval. (The disk drive, however, must not be required in the final target system. System software must be ROM based and tailored to ROM-based applications programs.) The disk drive is invaluable when large amounts of data must be stored. Disk-drive support there-

Can you imagine what a robot we could build with a staff of 250,000 (the entire readership of **Radio-Electronics**)? One key to the success of the **R-E** Robot is the collective development capability of that readership. In an effort to encourage the exchange of programs, sources of parts, hardware enhancements, and any other items of general interest, **Radio-Electronics,** Stock Drive Products, and Vesta Technology are each offering special support.

Radio-Electronics will open a special section of its new remote bulletin board system (RE-BBS) to builders of the **R-E** robot. You can reach the bulletin board by calling 516-293-2283.

Stock Drive Products (55 S. Denton Ave., New Hyde Park, NY 11040 516-328-0200) has agreed to supply a kit of parts for the drive sub-system, including two 10-inch pulleys and two 2-inch pulleys. Part number 2Z6-RL11862 is available for $32.00.

To simplify the mechanical aspects of building a robot, Vesta will sell, for a limited time, an aluminum chassis (resembling the one in Fig. 1) at cost, approximately $45. The fully-populated RPC will be available for $294, including 16K of RAM and the FORTH operating system. The Board-1 PC board is available as a bare board for $41, or fully assembled for $289. All source code for testing the robot and implementing RCL is available on a 5.25-inch disk for $2.00. All Vesta products are covered by a 15-day return policy. MasterCard or Visa accepted; no purchase orders or terms available. Please add $8.00 for shipping and handling for the computer board. Vesta Technology, Inc., 7100 W. 44th Avenue, Suite 101, Wheatridge, CO 80033, 303-422-8088.

Additional sources for various parts and sub-systems will be listed in future installments of this article. **R-E**

fore is included on the board. The Western Digital WD1770 single-IC floppy-disk controller supports both 5.25" and 3.5" drives. The preferred drive is the latest Citizen 3.5" model; it offers two important advantages: 1" overall height and 5-volt DC operation (no other supply voltages are required).

A terminal was selected as the programmer interface for program development. The variety of terminals available ranges from a computer system emulating a terminal to a battery-operated handheld unit. The terminal can be removed if it is not required by the application. Of course, the RPC must have a UART to drive the terminal. The Signetics SCN2681 was selected. That dual UART is supplied in a 28-pin package, an important consideration when trying to minimize board size. The second UART channel has selectable RS-232 or direct TTL I/O. That allows direct connection to inexpensive board-level modems or other serial data-interface units. RS-232 voltage-levels are generated on-board with a small DC-DC converter.

Most control and data-acquisition applications require time-of-day information, a function also included on the board. The National MM58274 real-time clock IC is supported by the same battery used by the RAM, which provides several weeks of battery operation.

Printer support during development is required. A single octal driver, together with a few leftover gates is all that is required to implement a parallel-printer port.

Interrupts in control applications are a key feature that must be supported. The internal interrupt-controller of the 80188 is not as flexible as an external interrupt-controller, nor will the 80188's internal interrupt-controller support a PC-DOS BIOS. An external interrupt-controller must be used in the system, so we selected the Intel 8259A because the 80188 supports that device.

A complete I/O map of the RPC is shown in Fig. 4.

The external bus connector considerably affects the flexibility of the system. The popularity and flexibility of the PC bus led to the selection of that standard. One problem with that bus is that peripheral boards are plugged-in in such a way that they mount at a 90° angle to the motherboard. Such an arrangement posed significant packaging problems. To elimi-

continued on page 69

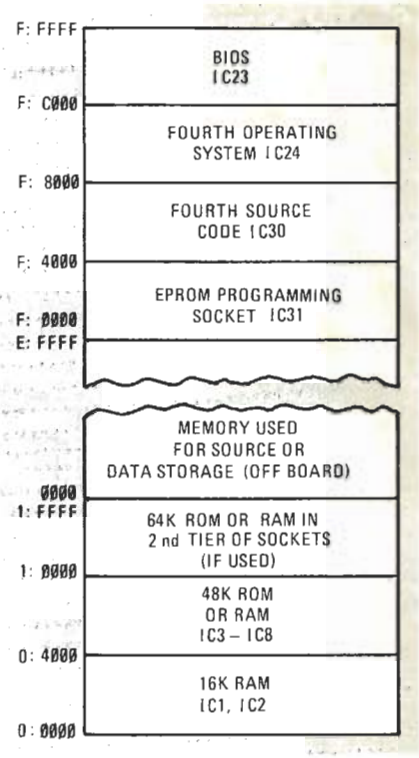


FIG. 3—MEMORY MAP OF THE RPC. Note the organization of the upper 64K of memory.

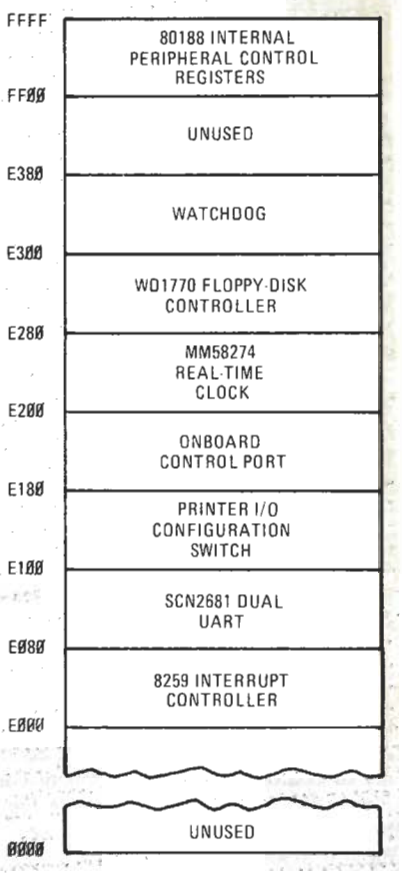


FIG. 4—THE RPC'S I/O SYSTEM organization is detailed here.

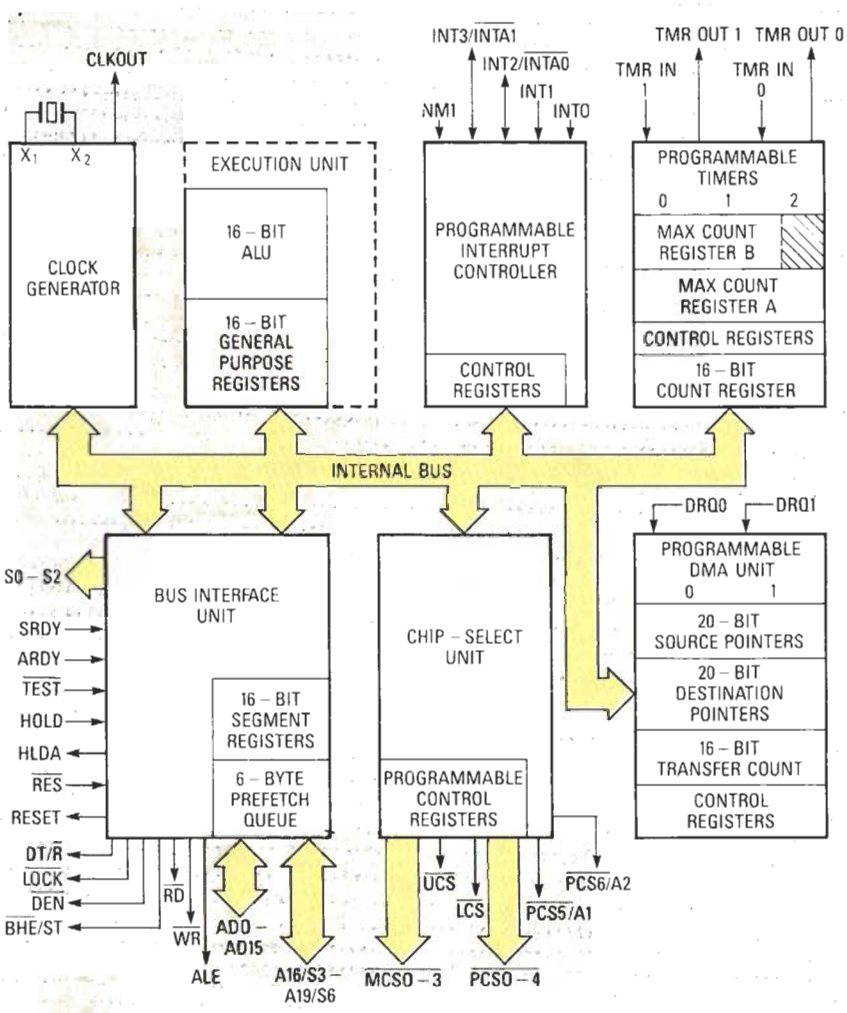


FIG. 2—THE 80188 MICROPROCESSOR was selected for our project because of its compatibility with the 8088 and its high degree of integration.

R-E ROBOT

continued from page 44

nate those problems, the bus connectors were modified to afford a more streamlined package. The fully buffered bus is daisy-chained to peripheral boards through mass-termination headers. Other buses can be supported as well, and simple adapters convert the bus connector to STD, iSBX or S100 standards. Peripherals designed for the VME, Q-BUS, and Multi-bus standards can be accommodated as well.

Designing the software

The key design requirements of the BIOS are twofold: First it must support the on-board functions of the terminal, auxiliary RS-232 communications, real-time clock, disk storage, and printer. Second, it must be able to boot MS-DOS applications software to capitalize upon the wealth of development tools available for the *PC*. The BIOS for the system is contained in a separate ROM and performs the same functions as the PC BIOS wherever possible. Due to the selection of a terminal as the operator I/O device, many graphics functions are not possible. Present day terminals fully support cursor-address functions, so that limitation is not as overwhelming as might be imagined. The BIOS performs admirably for applications-program development.

Several constraints must be met by the system software. The system must support minimal applications with an on-board stand-alone language in ROM, much as the PC does with Cassette BASIC. The logical choice is BASIC, because if the application is minimal, BASIC will probably suffice. Stand-alone BASIC will require that the terminal interface have the ability to read and write to disk and support all of the on-board functions. We wrote an integer BASIC interpreter to support the board and we stored it in a 16K ROM. We wrote the BASIC interpreter on a *PC* using standard BIOS calls to communicate to the operator. Using a *PC* as a development tool allowed direct transport of the BASIC when the hardware development of the circuit board was complete. We then installed that interpreter in the system during development and used it to test the remainder of the system.

Due to the popularity of Forth in control applications and the ease of installation of the Forth package, a version of Forth was also altered and installed in ROM on the system. The advantages of Forth were discussed in Part 1 of this article (**Radio-Electronics**, December 1986). Its disadvantage is the obscure nature of the language. Once your application is coded in Forth, it will be yours forever and you will probably not find another programmer to maintain it.

The actual development of high-level applications code in either of the two languages is very simple. Attach the terminal to the RS-232 port and select the EPROM containing the chosen high-level language. Enter the applications program into RAM and test it. Copying the program into EPROM is then simply a matter of executing the PROGRAM statement in either language. Alternatively, the program can be left in RAM. The battery will support the RAM during transport and field installation. That feature is required for programs written in some disk-based languages that are not designed for ROM.

Developing applications software under PC-DOS is also easy. The system behaves exactly like a PC, allowing development of application software in a wide assortment of languages. At this point, we have experimented with C and Pascal, and find that they run acceptably. Utilities in ROM allow easy use of the on-board EPROM programmer, so any .COM file can be easily stored in ROM.

Next month we will highlight the construction of the robot's base unit. **R-E**