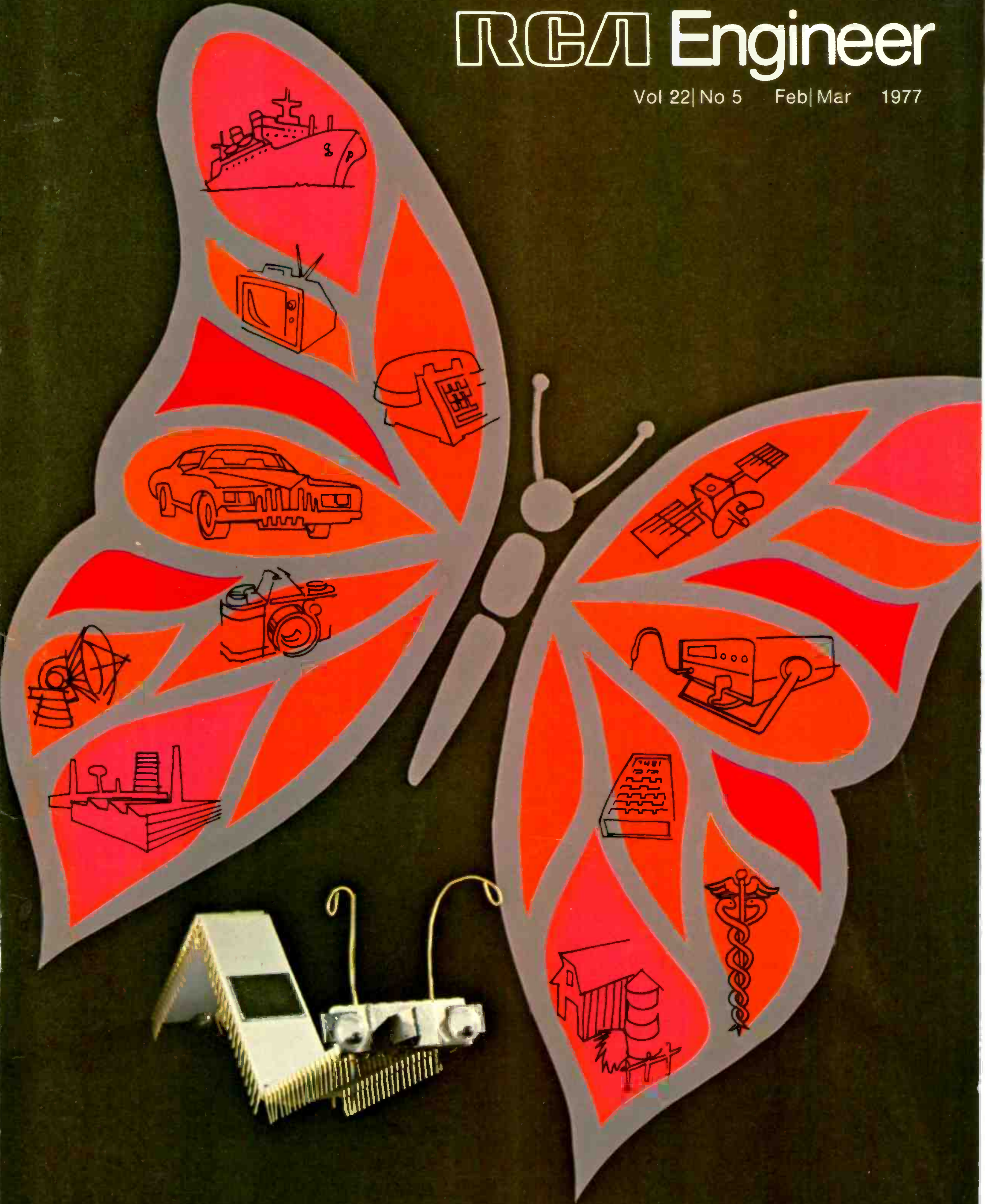
RCA Engineer

Vol 22 | No 5 Feb | Mar 1977



Microprocessors—design in transformation

RCA Engineer

A technical journal published by
RCA Research and Engineering
Bldg. 204-2
Cherry Hill, N.J. 08101
Tel. PY-4254 (609-779-4254)
Indexed annually in the Apr/May issue.

- To disseminate to RCA engineers technical information of professional value
- To publish in an appropriate manner important technical developments at RCA, and the role of the engineer
- To serve as a medium of interchange of technical information between various groups at RCA
- To create a community of engineering interest within the company by stressing the interrelated nature of all technical contributions
- To help publicize engineering achievements in a manner that will promote the interests and reputation of RCA in the engineering field
- To provide a convenient means by which the RCA engineer may review his professional work before associates and engineering management
- To announce outstanding and unusual achievements of RCA engineers in a manner most likely to enhance their prestige and professional status.

RCA Engineer
Vol 22 | No 5 Feb | Mar 1977
Microprocessors—design in transformation

Not much to look at, but microprocessors are transforming the world of electronics; the ideas are still blossoming. Clockwise from the bottom left, in the butterfly's wings, the application areas are: manufacturing, radar, photography, transportation, communication, home entertainment, marine and navigation, spacecraft, test equipment, calculators, medicine, and agriculture.

Concept: Ed Burke, Missile and Surface Radar, Moorestown, N.J. **Photography:** Tom Cook, RCA Laboratories, Princeton, N.J.

the microprocessor

technological change
economic opportunity

Today, the microprocessor is causing a major technological change in the electronics industry. We are now in the early phase of this change, which will eventually produce major market dislocations and significant leadership changes in various segments of the electronics market. The microprocessor's impact on the industry can be compared in magnitude to the conversion from tube circuits to solid state. At such a major point of inflection, opportunity is greatest for those who recognize the change and capitalize on it.

The RCA microprocessor program represents a success story; it was started in the RCA Laboratories in 1971, nurtured by the Solid State Technology Center through 1973 and 1974, and transferred to the Solid State Divison in early 1975. It has now expanded to a full line of large-scale integrated circuits, complete with support hardware and software. The Solid State Division is solidly behind the microprocessor program, with strong continuing support from the Laboratories and the Solid State Technology Center.

The significance of this product to RCA, however, does not lie with the sales of the microprocessor circuits themselves. The real potential is in new electronic equipment products that creative RCA engineers can develop using this product line—new products fully exploiting the advantages of LSI economics coupled with computer architecture and software flexibility. This is the area where there is excitement and opportunity for both management and engineering within RCA to do something innovative in end products, to apply new design techniques, and to take a fresh unencumbered look at solutions to problems and needs.

In short, it is a time for RCA to use the cost-effectiveness and flexibility of microprocessors to seize the opportunity that is occurring in the industry today.

Philip R. Thomas
Division Vice President
Solid State MOS Integrated Circuits
Solid State Division
Somerville, N.J.

How the microprocessor is affecting all of us

Software specialists are saying, "I told you so," as specialized hardware is replaced by standard microprocessors and custom software. Some industry forecasts indicate an eventual shift to an 80/20 software/hardware mix.

37

Designers and manufacturers of ICs have a lot of work ahead, because the microprocessor chip is only part of the microcomputer picture. What happened in the minicomputer business will probably happen with microcomputers—memories, I/O interfaces, and peripherals will represent the bulk of the sales—and that means an extensive design and production effort.

16, 71, 76

Systems, equipment, logic, and circuit designers are finding that their specialties are no longer separate parts of the design process. In microprocessor-based products, the work is best done by the same person or a closely cooperating group.

16, 28, 50, 68

Manufacturing, process-control, and testing engineers have virtually boundless opportunities ahead. Microprocessors are increasing the capability of controllers and inspection systems; at the same time they are making custom units economically justifiable.

50, 58, 64

Reliability and field engineers are finding that microprocessors are being used in an increasing number of the systems and equipment they work with. Service philosophy may change drastically—the "signature analysis" method of identifying troublespots represents an entirely new service philosophy that should be evaluated.

50

Technical managers are in the midst of vast market changes. The microprocessor can usually do more and do it more reliably—and often with cost and schedule advantages. This is a definite plus, but improvements are happening so fast that today's products are obsolete tomorrow.

4, 76

No matter what kind of engineer you are, microprocessors will affect your work, if they haven't already. **Get involved now.**

7, 12, 15

RCA Engineer

Vol 22| No. 5 Feb| Mar 1977

overview

introduction to microprocessors

COSMAC

on the job/off the job

microprocessor applications

advanced microprocessor technology

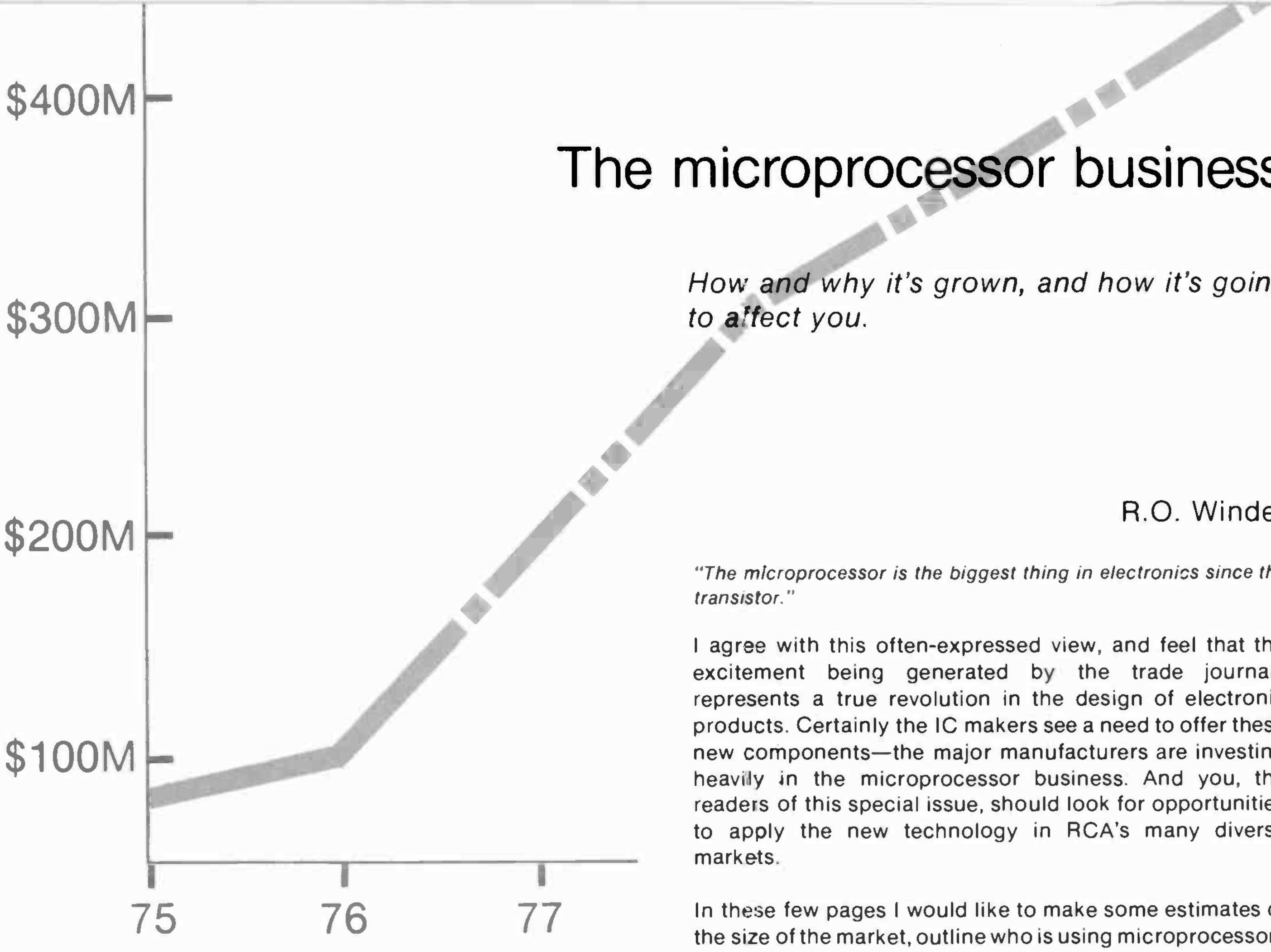
special—short course

departments

The microprocessor business

How and why it's grown, and how it's going to affect you.

R.O. Winder



Robert Winder is Director of Memory and Microprocessor Products in RCA's Solid State Division. His work with microprocessors began in 1971 at the RCA Laboratories, continued in the Solid State Technical Center in 1973, and moved to SSD in 1975. Dr. Winder received the David Sarnoff Award for Outstanding Technical Achievement in 1976 for his work with RCA's microprocessor products.

Contact him at:
Memory and Microprocessor Products
Solid State Division
Somerville, N.J.
Ext. 6005

"The microprocessor is the biggest thing in electronics since the transistor."

I agree with this often-expressed view, and feel that the excitement being generated by the trade journals represents a true revolution in the design of electronic products. Certainly the IC makers see a need to offer these new components—the major manufacturers are investing heavily in the microprocessor business. And you, the readers of this special issue, should look for opportunities to apply the new technology in RCA's many diverse markets.

In these few pages I would like to make some estimates of the size of the market, outline who is using microprocessors and why, and discuss the manner in which "micromakers"—the manufacturers of microprocessors—must support their customers (i.e., you). I'll use the RCA Solid State Division microprocessor product line as an example throughout—not only because I'm familiar with it, but because Solid State Division is competing directly and vigorously with Intel and the others in 8-bit microprocessors, a fact that many of you may not realize.

What the products are

Semiconductor memory circuits were the first general-purpose large-scale integrated (LSI) circuits. They, together with the more specific calculator, watch, and tv-game circuits, plus numerous custom chips, take advantage of LSI technology to provide greater function, miniaturization, and reliability, at lower cost. Microprocessors, which contain the CPU (Central Processing Unit) of a computer on one (or a few) ICs, may be thought of as complements to memories, so when Intel introduced them, it meant that LSI could be used for the entire "main frame" of a computer—that part of a computer common to all applications. Even though customers had to provide SSI and MSI logic for interface to the outside world, the resulting systems had great advantages in flexibility, miniaturization, and cost—the revolution was underway.

Solid State Division's offering in standard memory parts includes CMOS, CMOS/SOS, and NMOS RAMs (random-

Reprint RE-22-5-1
Final manuscript received October 18, 1976.

Table I
Rapid market expansion for the microprocessor families will be even more rapid in the next few years. (Source: *Electronics*, Jan 6, 1977.)

(millions of dollars)	*1975*	*1976*	*1977*	*1980*
Integrated circuits, total	1086.2	1512.7	1927.3	3106
Standard logic families, total	422.1	581.0	728.2	922
Microprocessor families, total	70.9	108.2	188.2	438
CPUs, total	24.9	38.5	63.9	166
MOS	23.1	35.6	58.9	150
4-bit	13.7	16.6	22.8	26
8-bit	5.8	13.0	25.0	70
16-bit	1.6	3.0	7.6	50
1-chip controllers	2.0	3.0	3.5	4
Bipolar, total	1.8	2.9	5.0	16
Bit slice	1.8	2.5	4.0	8
Full CPU	—	0.4	1.0	8
ROMs	14.5	16.7	32.0	80
RAMs	8.0	15.0	30.5	79
I/O interface chips	10.0	21.0	39.3	80
Peripheral chips	13.5	17.0	22.5	33

access memories) in 32x8, 256x4, 1024x1, 4096x1, and 1024x4 configurations, and CMOS ROMs (read-only memories) in 512x8 and (soon) 1024x8 configurations. Solid State Division does not presently offer a PROM (programmable read-only memory); their customers buy competitors' PROMs if their volume (under 50 or so) is too low for hard-wired ROMs.

Solid State Division's microprocessor business is built around the CDP1802 microprocessor, a CMOS implementation of Joe Weisbecker's COSMAC architecture, which is radically different from the Intel and Motorola architectures. COSMAC was specifically developed to minimize logic complexity, allow very compact programs, and interface efficiently with the outside world. This lower complexity permits us to manufacture the CDP1802 in CMOS at a cost comparable to NMOS and PMOS competition with its more complicated logic. And Solid State Division is able to compensate for its late start in this business by capitalizing on the well-known electrical benefits of CMOS technology—low and flexible power requirements, unexcelled noise immunity, and tolerance to wide temperature extremes.

The four-chip computer

As the market has grown, other kinds of product have become increasingly important—the circuits that interface the microprocessor to the outside world. These I/O (Input/Output) circuits consist of a mixture of custom and standard circuits. A typical custom I/O circuit would contain all the interfacing logic for the microprocessor, including A/D inputs, real-time clocks, and output-frequency generators. A customer would then have a 4-chip computer: 1-chip CPU, 1-chip RAM, 1-chip ROM, and 1-chip I/O circuit.

Such systems will sell for less than $20 in 1978 in automotive volumes. However, not everyone buys in automotive volumes. Most customers will be using "I/O ports" to pass 8-bit data to and from the CPU, UARTs (universal asynchronous receiver-transmitters) to pass serial data, MDUs (multiply-divide units) to do fast arithmetic when required, and a collection of "connective tissue" parts to facilitate memory and I/O expansion. Solid State Division presently offers a simple byte I/O port, and the other parts listed are in various stages of development.

The one-chip computer?

Finally, I will point out that the virtues of LSI suggest pulling the various pieces of a microcomputer into common ICs—CPU and memory together, or ROM and RAM together, or even CPU, ROM, RAM, and I/O together. These products lose flexibility, but gain in cost advantages. As indicated in the market-summary table, they will gain in importance as LSI technology matures.

What they're used for

The market-forecast table lists the estimated size of the total digital IC market. In my opinion, the percentage growth of microprocessor products within this market represents the combination of two important trends: 1) microprocessors are displacing other digital markets, and 2) they are creating their own.

First, an increasingly higher number of digital systems that might have been implemented using SSI and MSI will instead be done in LSI—using microprocessor technology. In other words, what might have been an *ad hoc* collection of event-recognition circuits, counters, shift registers, miscellaneous imbedded RAMs, A/D converters, and messy control logic will become organized into an efficient computer structure instead.

Why this trend? Because the microprocessor design process is better organized, more modular, more likely to implement the desired function, and generally is better understood. The various I/O interfaces to a computer are separate and work through a common bus structure and control scheme with the CPU. This makes deletions, modifications, and I/O additions easier. Control algorithms are implemented in subroutine-organized software—again, easier to modify. The payoff is in less engineering to reach a better product that is more easily maintained and enhanced.

The second trend swelling the market for microprocessors is the lower cost of LSI logic, which makes new products possible. For example, the automobile engine controllers that have been mechanical up to now—distributors,·carburetors, etc.—will be electronic in the future. Computers will become commonplace in the home, beginning as tv-game add-ons. And LSI technology will accelerate the use of electronics in business and industry with new low-cost microcomputer-based products.

What will be left to SSI, MSI, and non-computer-like logic?

First, some I/O interfaces will never be offered as off-the-shelf products—they will have to be built up with the standard logic building blocks. Second, some systems are so simple that the "overhead" of a computer structure is too expensive. Third, there is a clear performance limit to what microcomputers can do. They are basically sequential devices. For a given stage of a technology, *ad hoc* logic—the "messy" printed circuit board full of standard logic circuits—can do many functions in parallel that will over-task a microprocessor trying to perform the same functions sequentially using that same technology. Although multiple microprocessors can be used, cost goes up. And fourth, for very-high-volume applications, certain functions will be done in custom LSI in an *ad hoc* manner (generally because they require high-speed parallel information processing). But the central core of digital systems will need a reasonable amount of "intelligence" and can be operated by one sequential processor, so that the microcomputer approach will be the best, whether it is done with custom microcomputers, with assemblies of standard microprocessor components, or with a single-chip microcomputer.

How they're supported

The micro-makers recognized early that their customers needed more help in using microprocessors than they needed with SSI, MSI, or memories. Few of the early producers had the benefits that RCA has: an in-house system capability, and even a few computer buffs left over from the good old days. Early support systems were primitive and were redesigned repeatedly. Applications-engineering help, always in short supply as the business grew, took some time to mature in quality. Early documentation was atrocious, and training seminars nonexistent.

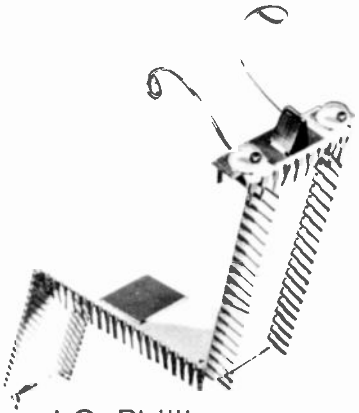
RCA Solid State Division customer support is typical of that now offered by micro-makers. Support systems we sell include:

- the Microtutor, a minimum complete computer used for learning the basics of microprocessors;
- the Evaluation Kit, (see Block, p. 22) a PC board and package of parts sufficient to put together a small computer for electrical experimentation and prototyping (add only a power supply and terminal);
- the COSMAC Software Development Package, (see Solomon, p. 37) a FORTRAN program that provides an assembler, simulator, and interactive debugger (available as card deck or tape, or on any of several time-shared systems, including CMS in Cherry Hill);
- the COSMAC Development System, a complete, expandable rack-mountable microcomputer (just add terminal), with empty slots to support prototyping or small production volumes, software for assembly and editing, a floppy-disc option for fast and convenient software development, and a higher-level-language operating-system option for the most efficient design and debugging of microcomputer hardware and software;
- and (early in 1977) the RCA version of Intel's "in-circuit emulator," a microcomputer that monitors and controls the activity of prototype or production microcomputer products—in short, the microprocessor engineer's oscilloscope.

Data sheets, manuals, application notes, seminars, and application engineering consultation—by phone and by direct visits—are an essential part of the support offered by all micro-makers (see Meyer, p. 43). Your access to application engineers will be a function of the supplier you work with and your importance to them. I'd like to say, on that subject, that you RCA customers, no matter how small the volume you need, are extremely important to us in Solid State Division. This is not only because of the obvious image impact, but because we believe we have the best products available, and that it is in RCA's best interest that you use them!

A final, especially interesting area of support the micro-makers offer is custom engineering for fee. This work ranges from implementing ROM patterns, through the design of custom I/O circuits, and into complete system design and software implementation. The Solid State Division, using RCA's unusual system-design capabilities, has done very well with this kind of service, getting prospective customers into production quickly with products likely to be succesful.

The Advanced Technology Laboratory and the the Solid State Technology Center's Design Automation group provide an essential set of tools for these jobs—APAR (Automatic Placement and Routing) for fast turnaround custom I/O chips, and a ROM design system that generates IC masks and test programs automatically. I strongly recommend that RCA engineers working in microprocessors acquaint themselves with these capabilities—it costs suprisingly little to generate ROMs and custom ICs quickly. The SSTC will be happy to work with you in putting its arsenal of tools to work.



Getting involved with microprocessors

Start with this article if you think that every computer is bigger than a breadbox.

J.C. Phillips

If the lessons of history and present trends mean anthing, the electronics world is going through a digital revolution—and the microprocessor is a primary catalyst. Digital technology—and specifically microprocessor technology—has grown so fast that many of you may still be groping for a place to start.

Start here. Those of you who have little or no knowledge of microprocessor technology should be able to pick up some of the buzzwords from this article—at least in sufficient depth to comprehend the message of this special issue, which focuses on the importance and the versatility of the microprocessor.

What is a microprocessor?

A microprocessor is the central processing unit of a microcomputer, and a microcomputer is simply a computer constructed from a handful of integrated circuits. Thus, any discussion of microprocessors and microcomputers can start with some general remarks about computers.

All computers perform five rather human-like functions: input, storage, control, processing, and output; Table I outlines these functions for humans and computers, including microcomputers. Although Table I is general, it places the control and processing functions in proper context with the input, storage, and output functions.

All microcomputers are based on these building blocks, with information exchanged among the blocks via a group of wires (one for each binary digit) called a *bus*; see Fig. 1.

How information is handled

The basic unit of information in any computer is the *binary digit*, or *bit*, capable of only two stable states, called "one" and "zero." All computers (including microcomputers) operate on groups of bits called *words*. The word is the primary group of bits that the computer manipulates (reads in, stores, reads out, etc.) in a single step. *Word length* for any given computer is usually determined by the number of bits that can be stored in a specific memory location;

Table I
All computers perform these five functions. These are known as *Harvard*-class machines. If, in addition to these properties, the machine stores instructions in the same form as data in memory—each equally accessible to the calculating section of the computer—then instructions may be treated as data, and the machine can modify its instructions. Such a machine is called a *von Neumann* or *Princeton*-class computer. Microcomputers are available in Harvard and von Neumann classes.

	Human		*Computer (microcomputer)*	
Function	*Device*	*Action*	*Device*	*Action*
Input	Five senses (maybe six)	Receive sensory data and convert it to impulses for brain	Keyboards, magnetic tape, paper tape, punched cards, switch closures A/D converter	Receive data as binary "ones" and "zeros"
Storage	Brain	Store facts in brain for recall and reaction.	Tapes, discs, drums, cores, solid-state circuits	Place data (1s & 0s) in specific locations for later use.
Processing	Brain	Interpret data and instructions stored in memory.	Central Processor Unit (microprocesor)	Gather information from appropriate locations in memory (storage) and interpret results.
Control	Brain	Select appropriate action based on data and instructions.	Central Processor Unit (microprocessor)	Select an alternative course of action on the basis of computed results.
Output	Voice, actions, gestures, etc.	React	Printer, keyboard, switch closures, CRT, D/A converters, alphanumeric displays	Deliver results for user.

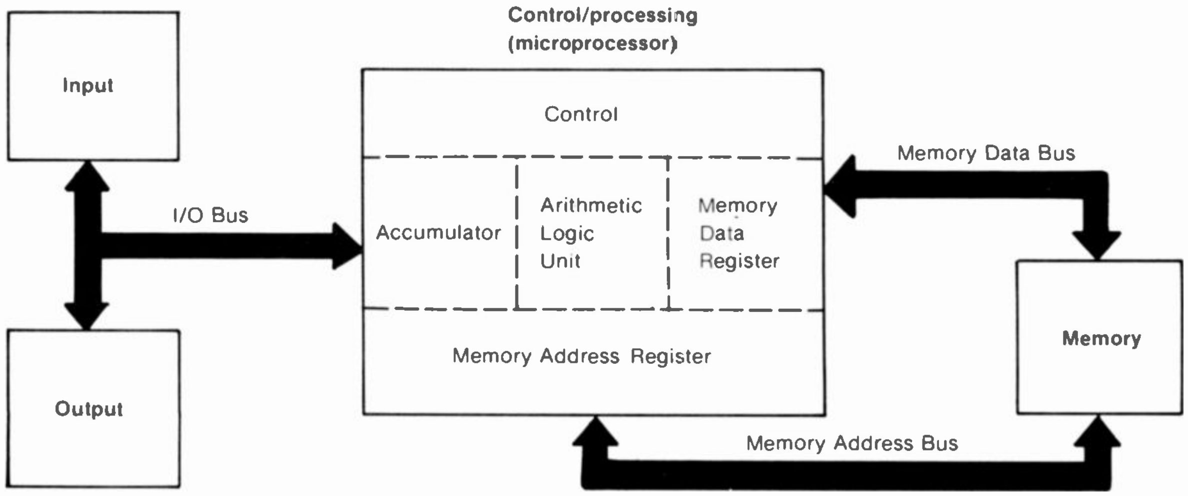


Fig. 1
Microcomputer architecture is based on four basic building blocks: input, output, memory, and microprocessor. The blocks are interconnected by buses—groups of lines [one for each *b*inary dig*it* (bit) to be transferred].

common word lengths are 4, 8, 12, or 16 bits. In general, the longer the word length, the greater the precision (number of significant digits). Also, the longer the word length, the richer the *instruction set* and the more varied the *addressing modes*.

An *instruction set* is the set of general-purpose instructions available with a given computer; generally, different computers have different instruction sets. *Addressing modes* are the ways that a given computer will specify the memory location containing the word to be operated upon. These modes are important factors in programming (and hence, in computer efficiency).

Two types of words are used in every computer: *data words* and *instruction words*. Data words contain the information that the computer processes; instruction words cause the computer to execute a particular operation. For example, an instruction may move data, do arithmetic and logic functions, control input or output (I/O) devices, or decide which instruction to execute next.

Byte is another commonly used term that is sometimes confused with *word*. A byte is any sequence (regardless of machine architecture) of *n* bits operated on as a unit. The most frequent size is 8-bits. A half-byte (4-bits) is called a *nibble*.

The basic task is to fetch and execute instructions

All of the operations of a computer are synchronized by a clock, and usually several clock pulses are required to accomplish the tasks specified by a single instruction. The execution of that single instruction is called an *instruction cycle*. For the RCA COSMAC microprocessor, a typical instruction cycle requires 16 or 24 clock pulses; a machine cycle requires eight clock pulses (see Fig. 2). Each instruction cycle typically consists of one or more *machine cycles*; the machine cycle is the basic microprocesor operation cycle. It usually consists of one of the following operations:

Fetch: The microprocessor uses an *address* (a number identifying a specific memory location) to read an instruction word from memory for use by the microprocessor; or

Execute: The instruction that has been fetched is decoded and the requested operation is performed.

There are other types of machine cycles (e.g., some types of interrupts), but most either fetch or execute.

Since a microprocessor must have sufficient hardware to fetch and execute instructions, it must contain:

—a *program counter*, which specifies the address of the next instruction to be fetched and executed. Normally, the counter is incremented automatically each time an instruction is fetched.

—an *instruction register*, which is a fast-access circuit used to store bits or words.

—an *arithmetic logic unit* (ALU), which executes such binary transformations as add, subtract, shift, AND, OR.

—a *control section*, which provides a way to address memory and to perform *branch instructions* (alter the sequence in which instructions are fetched based on the results of ALU operations).

Memory stores data and instructions

Instructions and data must be stored in *memory* so that they can be recalled at the appropriate time for the microcomputer to perform its function. Each instruction or data word is assigned a unique *address* that the microprocessor refers to when fetching or storing information. Memory is broadly divided into two classes: read-only memory (ROM); and read/write or random-access memory (RAM).

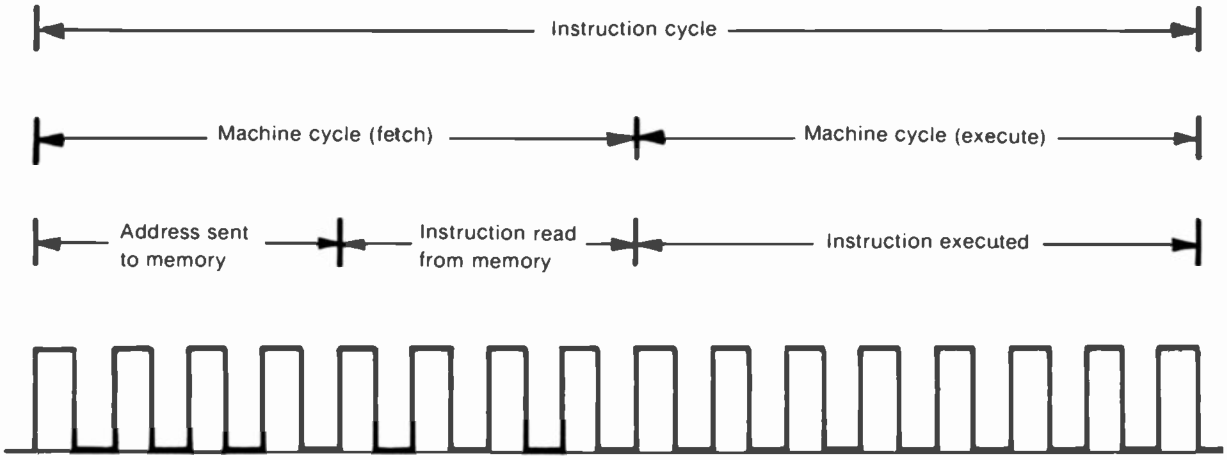


Fig. 2
Instruction cycle for COSMAC consists of one or more machine cycles; a machine cycle requires 8 clock pulses.

ROM (fixed memory) is used to store program steps and constant data values. It is difficult and time-consuming (and often impossible) to write into ROM; therefore, it is used in situations where the memory values do not change, e.g., in dedicated systems. Since this is a primary application of microcomputers, most of these machines use ROM for program storage. Some microcomputers use only ROM for storing program steps.

RAM (random-access memory) has both read and write capability. RAM is used to store data that changes during the operation of the system, e.g., results of calculations, or programs that are changed frequently.

Although most of the memory in a microcomputer is on separate chips, the microprocessor itself contains several fast-access storage units, called *registers*, that are used for temporary storage of data and instructions. The common registers are:

- *Memory address register* (MAR), which holds the address of the memory location being accessed.
- *Memory data register* (MDR), which accepts and temporarily stores the word coming from, or going to, memory (during read and write operation, respectively).
- *Accumulator* (AC), which holds the result of an arithmetical or logical operation to be 'used by the microprocessor. For example, the AC can provide an input to the arithmetic logic unit—ALU.
- *Program counter* (PC), which is automatically incremented to identify the next instruction to be fetched or executed.
- *Stack pointer*, which identifies (addresses) the last element in a stack. The stack is an array in RAM, separate from the microprocessor, which allows words or addresses to be accessed in a last-in, first-out fashion. A LIFO array is sometimes referred to as a *pushdown* array. Typically, a stack is arranged similar to a stack of papers, with each new input placed on top of the pile.
- *Scratch-pad memory*, which consists of general-purpose registers (RAM) that temporarily store data and addresses. The number and flexibility of such registers varies greatly among microprocessors.
- *Instruction register* (IR), which contains the instruction that has been fetched from memory and is being decoded and executed. This register usually has some external method of control (e.g., reset pushbutton or toggle switches) since the instruction register is typically used to direct the microcomputer to the location of the first program step.
- *Status register* (SR), which consists of one or more flip-flops (*flags*), provides information that is crucial to many arithmetic operations (for example, overflow from operations, zeros in the accumulator, sign of a number in the accumulator). This information is often used to decide what program step comes next.

Software —instructions that tell the microcomputer exactly what to do.

All operations that the microcomputer performs must be broken down into a series of individual tasks called *instructions*. The details and number of these instructions vary greatly depending on the microcomputer. Theoretically, any program may be written using the instruction set of any microcomputer. The length of the program and the time of the execution, however, may differ greatly with different instruction sets.

The instructions used may be divided into five functional types: transfer of data, control, subroutine linking, operation, and input/output. Such instructions may reference data from memory or from microprocessor registers, or may simply control the operation of the machine. Those that reference memory require an extra memory cycle to obtain the data and, therefore, may require more time to execute. In COSMAC, all instruction executes are identical.

A sequence of these detailed instructions needed for the computer to accomplish a specific task is called a *program*. Since the memory can store only 1s and 0s, the instructions must be encoded in binary. A program in this form is in *machine language*.

This is the lowest-level language in which programs can be written. Using it is tedious in that the value of every bit in every instruction of the program must be specified (e.g., by a string of binary digits for every word in memory).

Symbolic language allows instructions and storage locations to be represented by alphanumeric symbols. These symbols are chosen to make memorization easy and are therefore called *mnemonics*. (For example, in the COSMAC instruction set, OUT 1 represents *output 1*, IDL stands for *idle*, and SHLC means *shift left with carry*.) A program written in symbolic language must be translated to machine language before it can be stored and executed. The translation is performed by an *assembler* program. Most statements written in symbolic (or assembly) language translate to only one instruction in machine language.

To make programming easier, *higher-level languages* have been developed in which statements more clearly resemble English and mathematics, mostly for use on large-scale computers. Each is developed with a particular class of problem in mind. Some of the more popular languages are Fortran, ALGOL, APL, BASIC, COBOL, and PL/1. In these higher-level languages, one statement may correspond to many machine-language instructions. The conversion from a higher-level langauge to assembly code or to machine code is performed by a program called a *compiler*. (In a sense, a compiler is an assembler for higher-level languages.) Still another possibility is the popular BASIC, which is an *interpretive* language; i.e., the translation into machine code is accomplished at run time.

In preparing a program, it is frequently necessary to make numerous corrections and changes—for example, to correct a symbol or to insert one or more instructions into the program. This task is facilitated by the use of a special program called an *editor*. The editor allows the program to be stored into RAM and made available on an instruction-by-instruction basis for the purpose of making alterations easy.

Once written, a program is stored in one of the two types of memory. If the program is stored in ROM, it is always present in the machine and may be executed by branching to the address of the first instruction. This is the normal procedure in dedicated microcomputers. If the program is to be stored in RAM, which is volatile, then the program is usually loaded from keyboard, tape reader, or disc. To facilitate this loading, a special program called a *loader* is necessary and is often made resident in ROM. The loader is frequently combined with the capability to *read* or *write* directly into locations in memory so it can perform various housekeeping and debugging chores. In this case it is called a *monitor*.

Once a program is loaded, it is started by jumping to the first address in the program. Since most microprocessors are in dedicated systems, the program is usually stored in ROM. During the development process, however, it is very advantageous to store the program in some type of memory that can be modified easily. Most microprocessor manufacturers provide support systems (development systems) to aid in software development. (See the paper by Solomon in this issue.)

Advantages—lower cost, more flexibility, and new functions are just a few.

As demonstrated by several papers throughout this issue, the microprocessor offers significant advantages over both hardwired discrete logic and custom LSI (large-scale integrated) logic. The key advantage, of course, is that a given microprocessor chip can be used for a multitude of applications by simply changing the support hardware and the software. Thus, the cost of rather complex equipment is quite low because the microprocessor can be produced in large production volumes. In his excellent "Overview of microprocessor applications," in the June 1976 *Proceedings of the IEEE*, A.J. Nichols identified seven primary advantages of microprocessor-based designs, compared with discrete-logic designs.

1) The manufacturing costs of the product are lower. Typical microprocessor-based designs cost 20% to 60% of their TTL discrete-logic equivalents.

2) The time and cost for the original development is lower. An accomplished design team can cut the design time by about two thirds. As support tools improve, the design cycle will continue to decrease.

3) As a consequence, products can be brought to the market faster and in closer correlation with market needs. This can provide a significant edge in obtaining or increasing market share.

4) The microprocessor's inherent flexibility makes quick response to competitive pressures in the marketplace possible, with consequent increases in product lifetimes.

5) Greater functional capability can be provided at reasonable cost. This means better products for the same or lower prices.

6) The smaller number of components in a microprocessor system increases the reliability of the final product.

7) Should failure still occur, the microprocessor can, in some cases, perform self-diagnosis, providing substantial reductions in service charges.

In essence, the microprocessor offers the advantages of LSI without the cost penalty of designing the various mask and metalization patterns needed for custom LSI. Of course, high system complexity and significant production quantities will demand a custom design.

But there are some limitations

As stated previously, in doing its job of fetching and executing instructions by communicating with memory and with input/output devices, the microprocessor is a one-chip central processor unit for a microcomputer. However, placing all these central processor functions on a single integrated circuit imposes two very practical restraints:

1) The number of pins available for connection to the outside world—memory, I/O, power, and timing.

2) The number of functions that can be put on that chip.

The number of available pins limits the speed of information exchange.

Because the number of pins is limited, some type of serial (or multiplexed) data-transfer technique is used to carry information to and from the processor. In serial transfer, pins are traded for execution time; that is, entering 8 bits serially on a single pin takes 8 times as long as full parallel entry. Serial entry requires additional gating and storage circuits. Typically, memory interface pins are also used for I/O functions.

Functional limitations—designers are focusing on the rest of the system.

The number of functions that can be placed on a chip is limited primarily by the integrated-circuit technology used, and this limit is constantly changing (see the Clapp and Feller paper in this issue).

In general, several types of one-chip microprocessors have evolved to include the functions outlined above. Further work will probably be focused on improving the rest of the microcomputer system, with particular emphasis on input/output chips, and on multiprocessor systems (see the paper by Russo in this issue).

Implications for engineers— put your soldering iron on the shelf next to the tube tester and slide rule

In a sense, the microprocessor is bringing a breath of fresh air to the engineering lab. Probably the most tangible, albeit mundane, change brought about by the microprocessor is that the labs no longer reek of burning resin and carbon resistors. Breadboards are created in a new way—with manufacturer-supplied development systems (see the paper by Block, this issue). Design changes are implemented via software. Thus, learning about microprocessors also requires learning computer programming—the soldering gun and needlenose of the future!

RCA has made it easy for you to learn about microprocessors and computer programming (see the DiGirolamo paper in this issue). If you still don't think you should, reread Phil Thomas' inside cover message and glance at the market projections in Bob Winder's paper; also note the numerous applications cited in this issue.

Your efforts will be amply rewarded. A powerful, pocket-sized computer readily and economically available as a tool and as a design element carries significant implications for every engineer, regardless of discipline. There is a microprocessor in your future, and the future is now.

References

In anticipation of this special microprocessor issue of the *RCA Engineer*, we asked several microprocessor experts to identify the microprocessor literature that may be useful and interesting—particularly to neophytes.

Mort Lewin of Rutgers University (consultant to the Solid State Divison) called our attention to an exhaustive bibliography compiled by Ann R. Ward of Bell Laboratories. This bibliography appeared in two separate issues of the IEEE Computer Society journal *Computer* (part I in July 1974 and part II in January 1976).

Dr. Lewin then identified four introductory books with which he has had contact:

1. Hilburn, J.L. and Julick, P.M.; *Microcomputers/Microprocessors: Hardware, Software, and Applications* (Prentice-Hall; 1976).
2. Soucek, B.; *Microprocessors and Microcomputers* (Wiley; 1976).
3. Barna, A. and Porat, D.I.; *Introduction to Microcomputers and Microprocessors* (Wiley· 1976).
4. Adam Osborne and Associates, Inc.; *An Introduction to Microcomputers* (Berkeley, CA.; 1976).

Dr. Lewin particularly recommended the first and last, as did R. Smith from MSR, Moorestown, and several engineers from RCA Globcom. The Globcom engineers—John Frankle, Nick Giacketti, Bill Henn, and Tony Longo—also identified the following as useful sources:

5. Wester, J.G. and Simpson, W.D.; *Software Design for Microprocessors* (Texas Instruments, Inc.; 1976).
6. Motorola Semiconductor Products, Inc.; *Microprocessor Application Manual* (McGraw-Hill Book Co.; 1975).
7. *Proceedings of the IEEE*, Special issue on microprocessor technology (June 1976).
8. *EDN*, Special microcomputer reference issue (Nov. 20, 1976).

Specific microprocessor types are covered by the manufacturer.

Complete information on specific microprocessor types is usually available in the form of data sheets, instruction manuals, and application notes. For more information on COSMAC literature, refer to the paper by Meyer in this issue.

Applications areas papers are covered elsewhere in this issue.

Literature sources dealing with broad applications are listed with the papers in this issue that cover those application areas. For example,

Manufacturing and process control ....
Wang/Wu/Russo/Abramovich/Marcantonio
Test and control .... Marcantonio/Russo
Communications .... Lippman

Reprint RE-22-5-2
Final manuscript received February 17, 1977.

Learning to use microprocessors

G.B. DiGirolamo

You can learn microprocessor technology and gain hands-on experience in microprocessor applications through a comprehensive course developed by Corporate Engineering Education exclusively for the RCA technical staff.

Engineers are generally aware of the enormous promise of the microprocessor, but many are unprepared technically and psychologically to use this tool effectively. In May 1975, Corporate Engineering Education introduced a course called "Microprocessors for Logic Design" that has helped more than seven hundred members of RCA's technical staff at seventeen locations understand microprocessor technology and develop "hands-on" skills in using microprocessors. "Hands-on" learning is built into the course through team projects using a microcomputer, the RCA COSMAC Microtutor. Several examples of in-course team projects recently completed by engineers at Automated Systems in Burlington, Mass. are shown below.

About the course

The objective is to teach microprocessor technology.

The course is intended to teach microprocessor technology in depth and to help develop skills in designing with, and for, microprocessor systems. Those who complete the course will be able

- To understand the internal structure of a microprocessor,
- To understand microprocessor interfaces with both random-access and read-only memories as well as input/output devices,
- To use microprocessors as a powerful tool in logic design.

Alumni of this course will also be able to do microprocessor system design—configuring a multiplicity of input and output devices and memories into a microcomputer. Finally, participants will learn the state-of-the-art in available hardware and software, be able to interpret manufacturers' specifications, and use effective criteria to evaluate competitive products.

As prerequisites, you should know something about logic design and computers.

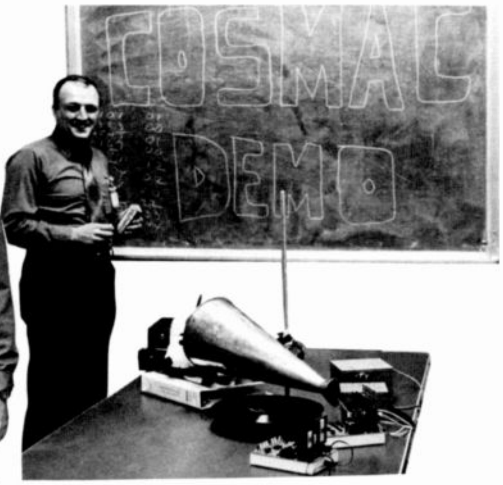
To get the most out of this microprocessor course, participants should start with knowledge and experience in logic design equivalent to CEE course C30—Modern Logic Design I. Some exposure to programming and to computer concepts is helpful, but not essential.

The success of the microprocessor course depends heavily upon the effectiveness of an on-site instructor in dealing with the students' learning problems. Therefore, the course

Reprint RE-22-5-5
Final manuscript received December 16, 1976

Jonathan Goode, Marc LeVarn, and **Tony Macadino** developed a "Random Number Game" in which a player attempts to correctly identify a random number generated by the microprocessor.

Ed Kramer, Roger Plaisted, and **Ted Kupfrian** decided to expand the limited input/output capabilities of the Microtutor by interfacing it to a video terminal as their project. They named their project "Microtutor 'UART'."



Gus Fortin, "the professor" at Aerospace Systems in Burlington, taught the microprocessor course and believes microprocessors are getting into all of our products, so he advises all designers to get aboard this new technology. As emphasized in the paper, the Associate Instructor is crucial to the success of this course.

also requires an Associate Instructor, who must have some prior microprocessor knowledge and understand computer fundamentals. Also he must know some assembly language, have written a program which has run successfully, and have logic design capability.

Optimum class size is sixteen participants.

Teams of four are formed in each class, and a Microtutor is assigned to each team.

Seventeen class sessions are suggested.

Included in the seventeen class sessions (2½ hours each) are fourteen video taped lectures, two sessions devoted to review and project demonstrations, and a session for the final examination. Homework is assigned between each session with "hands-on" team exercises between Sessions 5 and 9.

How to sign up for the course.

If you want to take this course, or any other CEE course for that matter, tell your supervisor or your local training manager. When a sufficient number of engineers (usually eight or more) express an interest, the training manager will arrange to get the course materials from Corporate Engineering Education.

Video-taped lectures, hands-on experience, and class participation are the important features.

The primary instruction medium is video-taped lectures. Each class session consists of the primary instruction supplemented by reviews of homework, class discussions, and class exercises, all led by the Associate Instructor.

A study guide, specially prepared for this course, contains printed forms of all the visuals and illustrations used on video tape. In addition, it contains several selected papers and reprints and represents a substantial presentation of

Bill Shubert and **Ron Tetrev** devised a "Microtutor Editing Keyboard" using a hex keyboard and a two-character hex display to develop a unit to easily accept and modify machine-language programs.

Steven Hadden, Larry Hill, and **Robin Hulls** programmed music in a modern-day version of "His Master's Voice." Snoopy is listening to the "William Tell Overture" as programmed on the COSMAC Mictotutor appropriately titled "Microtutor Music Master."

perspective

Where are all the others?

As mentioned in the article, about 700 RCA engineers have completed the CEE course, "Microprocessors for Logic Design," since its release in May 1975. This interest is indeed gratifying and actually not suprising in view of the excitement and importance of microprocessor technology. But 700 represents only a small fraction of RCA's engineers. How about you? Why haven't you signed up for the course?

Afraid it's too hot to handle?
Well it is tough. Requires a lot of homework. But think about the other tough things you've done. Why not accept it as a technical challenge that will be stimulating? Our very competent Associate Instructors will help you.

Figure you don't need it?
Come on now! How long can you really pass yourself off as a knowledgeable engineer without being conversant with one of the most significant technologies ever developed? How can you know you won't find an application for the technology when you really don't understand it? And isn't it just a matter of time? Why be last?

Maybe you'll learn it some other way?
Sure! It'll come as if by magic. You'll talk a little with this fellow and with that fellow, you'll read a few articles, and. . . . Sound familiar? Don't count on it. This technology is more complex than that. Oh, you can learn some of the jargon, but let's see you write a program—even a simple one.

You're too busy?
Right! You've got all those commitments competing for your time. Don't we all! I bet every one of the 700 who've completed the course was just as busy. This is a matter of priorities. Think a little about the importance of microprocessors.

Don't have the prerequisites?
OK, that's legitimate. Why not get a group together and take our C30 course "Modern Logic Design." Then you'll have the prerequisite.

You'll wait for a better course?
There are all sorts of microprocessor courses available. Practically everyone in engineering education has one. But you'll have to go where it is and that won't be where you work. Also, it won't be tailored to RCA as this one is. And we doubt if you can find one as good as ours.

It's high time you think some about all this. Don't allow youself to be obsolete in microprocessor technology. Now's the time to get in. It's going to become more difficult as time passes.

—W.J. Underwood

Bill Underwood is Director of Engineering Professional Programs and responsible for the Continuing Engineering Education function.

Contact him at: **Corporate Engineering, RCA Bldg. 204-2, Cherry Hill, N.J., Ext. PY-4383** microprocessor technology. A syllabus of the course is given in Table I.

Extensive use is made of the RCA COSMAC Microtutor, which is a microcomputer designed by RCA Laboratories. As mentioned previously, this small device is used by class participants for "hands-on" experience. Participants write programs in assembly language and run them on the "Tutor." This activity has resulted in several interesting projects.

Future courses will deal with single-chip microprocessors and new applications.

Future course development, involving the COSMAC single-chip processor that replaces the two-chip version, is now being considered. As in the present course, the single-chip course would cover the hardware and software aspects of the microprocessor as well as the practical aspects. This includes the "hands-on" use of a Microtutor as an integral part of the course. More recent information on new applications of microprocessors and microcomputers will be included in the applied segment of the course.

The present course, C55, responded to a need and was successful because it was timely. This course brought a newly unfolding technology to the engineering work force. However, microprocessors differ from other technologies in that they put unprecedented power in the hands of the individual design engineer. The microprocessor course made this design tool understandable and useful.

Acknowledgment

Credit for the development and preparation of the microprocessor course is due Dr. Robert O. Winder, Solid State Divison, Somerville; Dr. Anthony Robbi, Solid State Technology Center, Somerville; Dr. Paul Russo and Michael Lippman, RCA Laboratories, Princeton; Richard Smith, Missile and Surface Radar, Moorestown; and Dr. Morton Lewin, formerly with RCA Laboratories and now with Rutgers University. The author acknowledges with gratitude their contributions in making the microprocessor course possible.

Gerry DiGirolamo is the course development administrator for the microprocessor course described in this paper. He has been working in Corporate Engineering Education since 1965. In the photo, he is directing a video-taping session.

Contact him at:
Continuing Engineering Education
RCA Bldg. 204-2
Cherry Hill, NJ
Ext. PY-5141

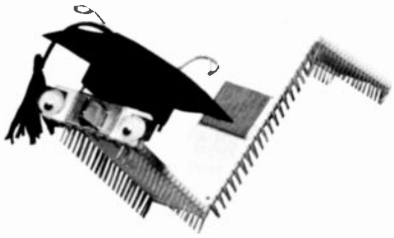


Table I
Microprocessors for logic design, course outline. Sessions 16 and 17 are for review and final examination, respectively.

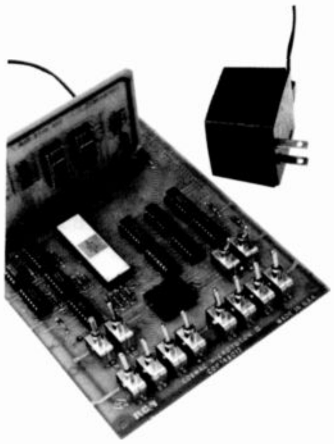
Session	Description
1 — *overview*	What a microcomputer is, and what a microprocessor is; why and when to use a microprocessor.
2 — *basic processor operations*	Binary numbers. Hexadecimal notation and the concept of a "byte." Random-access memory and its addressing. Read-write vs. read-only memories. Instruction format of COSMAC.
3 — *COSMAC architecture I*	COSMAC instruction actions, including the concept of the conditional branch. The input/output instruction.
4 — *COSMAC architecture II*	COSMAC binary arithmetic instructions. COSMAC arithmetic-logic-unit.
5 — *architecture and assembly language*	COSMAC architecture review. COSMAC instruction set. Assembler directives.
6 — *stacks and subroutines*	Subroutines. Nesting. Stacks. Interrupt service routine.
7 — *software techniques and system development*	Assembly-level programming. Trade-offs between assembly level and higher level languages. Time-sharing software development systems. Stand-alone microprocessor development systems.
8 — *input/output structures*	One-level I/O structure. Two-level I/O concept. Microprocessor-based system design. Operation of the DMA channel. External flags (EF1-EF4) to identify and decode interrupts. Interrupt priority.
9 — *single-chip COSMAC microprocessor*	Differences between the single-chip COSMAC and its two-chip predecessor. The 1800 family of support chips.
10 — *midterm review and test*	Review of sessions 1 through 9. Demonstration of team projects.
11 — *comparing microprocessors*	LSI-oriented technologies. Intel 8080 and the Motorola 6800. Comparisons with RCA COSMAC system.
12 — *bit-slice processor*	Bit-slice microprocessor. Monolithic Memories (MMI) 5701/6701. Application to coordinate conversion.
13 — *RCA FRED system*	FRED, an experimental model of a minimum-cost home/school computer. Use of an unmodified tv set as a display. The power and flexibility of the dot-matrix display philosophy.
14 — *microprocessor-based data communications system*	Narrowband dedicated leased-channel store-and-forward data communications system. The TTY and RS-232 communications interface. Floppy discs and floppy disc drives.
15 — *microprocessor-based data communications systems and multiprocessor structures*	Difference between bits/second and baud rate. Floppy-disc seek, and latency, access, and data transfer times. Worst-case and average disc thruput rates. Disc thruput rates, related to data communication system thruputs. The general network and master-slave organization. Two-microprocessor implementation of leased channel system.

Getting your hands on COSMAC hardware

Eventually, you're going to have to stop reading about microprocessors and start using them. Here are the COSMAC options available for when that time comes.

W.D. Lauffer

If you have no previous computer knowledge, Microtutor II is the place for you to start.



An improvement on the popular Microtutor I, it includes everything you need to write and run machine-language programs. You can learn to program games, educational exercises, and controllers quickly—the *Microtutor Instruction Manual*'s lively style is an excellent example of making technical education painless.

Specifically, Microtutor II includes the CDP1802 microprocessor, 8-bit toggle-switch inputs, a crystal clock, a two-digit hex display, power supply, and a 256-byte RAM—all contained on two printed-circuit cards. There are also sockets for memory expansion and external options. For example, built-in operating system memory addressing is possible with only one more 256-byte RAM card.

Or, start from scratch with the COSMAC "ELF."

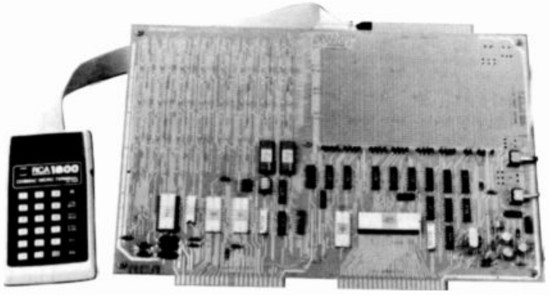
The true experimenter will want to start with the individual chips and components and build up a microcomputer system. If this is the route you'd like to take, consider the COSMAC "ELF." Joe Weisbecker has written a series of articles on its construction, programming, and expansion in *Popular Electronics*. The issues are August '76, September '76, and March '77; a future article will cover video interfacing.

COSMAC VIP is designed specifically as a home computer.

Sometime this year, VIP will appear on the market. A low-cost home computer aimed at video graphics and games, it uses a video monitor, audio cassette recorder, and a 16-position key pad as I/O devices—keeping total system cost down. As designer Joe Weisbecker says, "It's not fair to ask the limited-budget, unsophisticated beginner to buy a system for several hundred dollars but then require an additional $1000 for a terminal and enough RAM to make it useful." These I/O devices and a hex interpretive language make is possible for users to write and load their own games and programs simply and inexpensively.

Compare COSMAC with the competition.

The COSMAC Evaluation Kit was designed to let you see just how RCA's 1800-series components work. The kit includes a printed-circuit board, a manual, and all the parts you'll need to evaluate, and design various systems around, the CDP1802 CPU, the CDP1832 ROM, the CDP1824 RAM, and the CDP1852 byte I/O port.



By adding a teletypewriter, hand-held Microterminal, or other terminal, you can use the built-in ROM utility routines to write programs and exercise the system. The board has room for memory and I/O expansion. For a detailed description, see Dennis Block's article on the Evaluation Kit is this issue.

Do system design with the COSMAC Development System.

When you get down to making prototypes, you'll need to breadboard and debug hardware/software configurations on the way toward the final design. This is where the COSMAC Development System (CDS) fits. Its 19-inch rack-mountable chassis provides a power supply, front-panel controls, printed-circuit backplane, a set of small printed-circuit boards, and 22 empty card slots. The basic set of cards can support up to 65,536 bytes of memory.

The CDS can be used with a number of different terminals ranging from paper tape to floppy disc; speed will depend on the investment you decide to make.

CDS software includes the COSMAC resident assembler and resident editor. Larry Solomon's paper in this issue should tell you how to go about using them while developing your COSMAC system.

Reprint RE-22-5-13

The CDP1802: a powerful 8-bit CMOS microprocessor

A.W. Young

Learn the advantages of COSMAC and its supporting family of chips; then find out what's involved in putting a system together.

The RCA CDP1802 microprocessor is a powerful and efficient single-chip 8-bit CPU that forms the heart of a general-purpose microcomputer. It is easy to understand, easy to use, and low in cost. COSMAC architecture is based on the efficient use of short-word-length instructions within an interface restricted to 40 pins. Instead of using a minicomputer instruction set forced into 8-bit bytes with multiple-byte instructions, the COSMAC architecture has been developed around single-byte instructions.

Reprint RE-22-5-12
Final manuscript received November 29, 1976.

Alex Young, as Manager of Memory/Microprocessor Design Engineering, is responsible for the design of the RCA1800 microprocessor family and its support systems. Previously, he had been Manager of COS/MOS Engineering and had designed over 20 CMOS circuits as a member of the Custom Circuit Design Group.

Contact him at:
Memory/Microprocessor
Design Engineering
Solid State Division
Somerville, N.J.
Ext. 6468

The applications addressed by the CDP1802 include low-end minicomputer replacement, general-purpose controllers, emulation of hardwired logic, and totally new areas made possible by the advent of low-cost, stored-program, general-purpose data processing.

The CDP1802 is fabricated with a self-aligned silicon-gate CMOS technology, giving these results:

1) *Full temperature range* (−55°C to +125°C) for ceramic-packaged devices.

2) *Low power dissipation*—generally an order of magnitude or more less than other 8-bit microprocessors at maximum processor speed. Specifically, 50 mW at 6.4 MHz and 10 V; less for lower voltages and slower clock rates.

3) *Single noncritical wide power-supply range*. The CDP1802 will operate from 3 to 12 V over its full temperature range. Since the low-current single-voltage supply does not require sophisticated regulation, power-supply design is straightforward and simple.

The CDP1802 has captured these CMOS technology features to the extent that it "looks like" another CD4000-series CMOS device. The COSMAC design, however, provides additional CMOS technology advantages, including:

1) *High noise immunity*. Inputs have been specifically designed to switch at 50% of supply voltage. To account for normal processing tolerances and the effects of temperature and voltage range, the design point translates to a guarantee of 30% of supply voltage.

2) *Static operation*. The CDP1802 is completely static. No minimum clock frequency is required and no complex multiphase clock generators are necessary. An on-chip oscillator amplifier is provided and requires a single feedback resistor and crystal connection to operate the clock. Alternately, an external single-phase RC oscillator could be used as the system clock.

3) *High speed*. The CDP1802 will operate at 10 volts with instruction times

Table I
Features of the CDP1802 microprocessor.

Static silicon-gate CMOS technology—CD4000-series compatible

Instruction times of 2.5 or 3.75 µs at 10 V

Compatible with CDP1801 software

Full temperature range (−55°C to +125°C)

30% noise immunity

Single supply voltage

Wide operating voltage range: 3-12 V

No minimum clock frequency

Low power—50 mW at maximum 10-V instruction time

TTL-compatible

Single-phase clock

Optional on-chip crystal-controlled oscillator

Simple control of reset, start, pause, and load modes

8-bit parallel organization with bidirectional data bus

Works with any combination of the standard RAM and ROM

Memory addressing of 65,536 bytes

91 instructions

On-chip DMA

Program interrrupt

Programmed I/O

Four input flags

Programmed output port (1-bit)

16 × 16 register matrix for use as multiple program counters, data pointers, or data registers

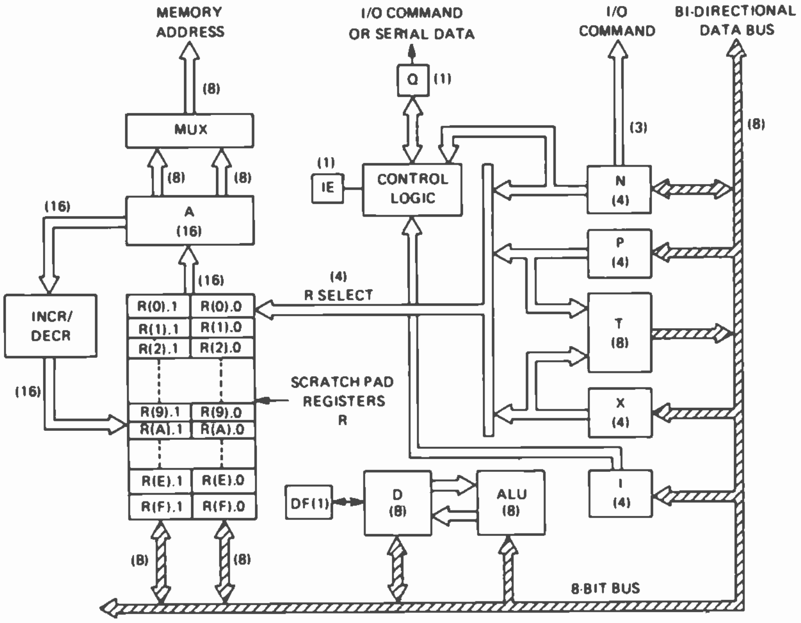


Fig. 1
COSMAC architecture is based on sixteen on-chip 16-bit scratchpad registers. Three 4-bit pointers (N, P, and X) can designate any of the 16 general-purpose registers as a program counter, memory address register, data source, or data destination.

of 2.5 or 3.75 μs. All but long-branch and long-skip instructions execute in 2.5 μs.

High output drive, low input current, low quiescent power, and extra static protection are also provided by design. The combination of COSMAC architecture and CMOS technology make the CDP1802 a high-performance cost-effective device for a broad range of applications. Specific features of the CDP1802 are summarized in Table I.

COSMAC architecture

The COSMAC architecture has been carefully developed for efficient operation in an 8-bit environment. Fig. 1 shows the internal processor architecture.

The basic architecture provides 16 on-chip 16-bit scratchpad registers.

These registers (R) can be used to point to data in memory, point at programs, or store data (two bytes per register). An additional 8-bit data register (D) is used to buffer data transfers between an R register and the data bus and also functions as a conventional accumulator.

Any of the 16 general-purpose registers can be designated as a program counter, a memory address register, a data source, or a data destination by means of one of three 4-bit pointers.

The register labeled P holds the 4-bit program-counter pointer. One instruction loads P with any 4-bit number, so the program counter can be changed by simply changing the number in P.

A second 4-bit register, N, stores a variable pointer carried along with the instruction. For example, a load instruction designates one of the 16 registers as an address register. The bits in the instruction doing the pointing are stored in N to designate the appropriate register.

A third 4-bit register, X, stores a pointer used to designate an address register during input/ouptut and some ALU instruction. Like P, it can also be loaded with any number by a single instruction.

Using four-bit pointers to indirectly specify a 16-bit address is a key feature of the COSMAC architecture.

This feature results in compact code because operand addresses can be efficiently carried by a short 8-bit instruction. By providing a generous quantity (16) of general-purpose registers, the programmer can set up many pointers to data tables and utility routines and so access them quickly and efficiently.

The notation R(P), R(N) or R(X) refers to the general-purpose register designated by the 4-bit code in P, N, or X, respectively. M(R(P)), M(R(N)), or M(R(X)) refers to the memory byte pointed to by R(P), R(N), or R(X), respectively. Using hex notation, R(3) designates the general-purpose register corresponding to binary code 0011. R(3).0 refers to the least-significant byte of R(3), and R(3).1 refers to the most-significant byte of R(3).

In addition to the general-purpose register array, data register, and register pointers, the COSMAC architecture provides a conventional ALU that performs arithmetic and logic operations between operands stored in D and M(R(X)) or M(R(P)), with the result stored in D. An overflow bit, DF, can be used for conditional branching.

Instruction cycles are subdivided into fetch and execute cycles, also referred to as machine cycles. When instructions are fetched from M(R(P)) during the instruction-fetch cycle, the four most-significant bits are placed in register I and designate an instruction op-code. The four least-significant bits are placed in register N and specify R(N) as a data destination (as in D→RN.0—put low R(N)) or as a memory address register (as in load D with M(R)N)). The four-bit N field is also used to extend the op-code, as in ALU and branch instructions.

The COSMAC I/O interface has been given particular prominence in the architecture design.

This interface can be summarized as follows:

1) *Four input flags* ($\overline{EF1}$-$\overline{EF4}$), which can be tested by conditional branch instructions.

2) *A serial output* (Q), which can be set and reset under program control and tested by conditional branch instructions.

3) *Programmed I/O data transfer*, which uses the N bits as a device-select code and transfers data between the selected device and memory.

4) *A maskable interrupt mechanism*, which can be activated by the $\overline{\text{INTERRUPT}}$ input. When this interrupt occurs, the old values of P and X are automatically saved in temporary register T and new values are jammed into P and X, effecting a 1-cycle switch to a new program counter for interrupt servicing.

5) *A DMA channel*, which can be activated by $\overline{\text{DMA IN}}$ or $\overline{\text{DMA OUT}}$

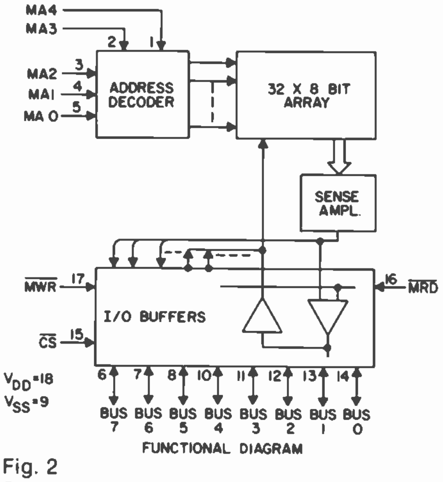


Fig. 2
Static 32-byte CMOS RAM, the CDP1824, was developed specifically for microprocessor systems requiring a minimal amount of writable storage.

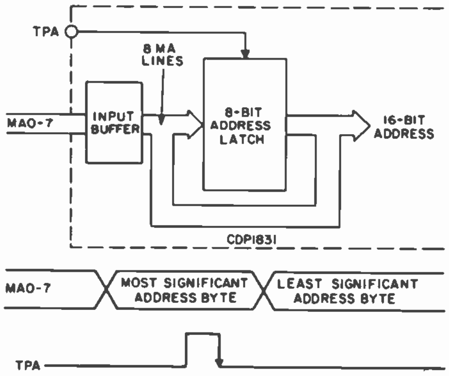


Fig. 4
Mask-programmable COS/MOS ROM, the CDP1831, uses address-multiplexing method shown here.

and uses R(0) as the DMA pointer. Each DMA request causes one machine cycle to be "stolen," appropriate memory address and control signals to be generated, and the pointer incremented.

6) *Timing synchronizing signals* (TPA and TPB), which assist in data transfers and general system timing.

The CPU performs instructions in 2.5 or 3.75 microseconds.

General operation of the CPU is controlled by the inputs $\overline{\text{WAIT}}$ and $\overline{\text{CLEAR}}$, giving four possible operating modes:

LOAD—Holds the CPU in the idle state and allows an I/O device to load memory using R(0) as the memory address register.

RESET—"Resets" the CPU by forcing X, P, Q, and R(0) to 0 and enables interrupts. After RESET, the CPU starts execution of instructions from memory location 0000 with R(0) as the program counter.

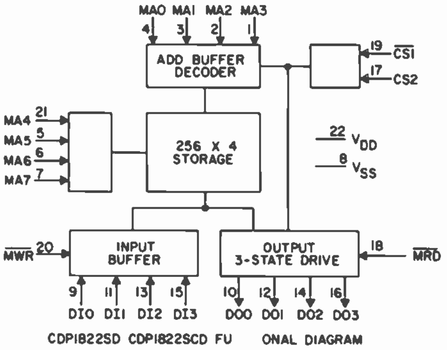


Fig. 3
CDP1822 RAM is designed for the COSMAC CDP1802 microprocessor, but is also functionally compatible with the standard 2101 type.

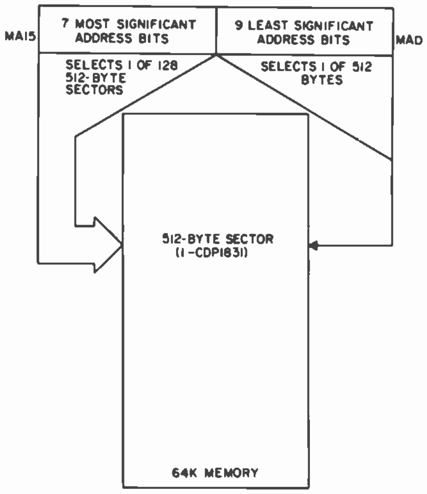

7 MOST SIGNIFICANT ADDRESS BITS
9 LEAST SIGNIFICANT ADDRESS BITS
MA15
MA0
SELECTS 1 OF 128 512-BYTE SECTORS
SELECTS 1 OF 512 BYTES
512-BYTE SECTOR (1-CDP1831)
64K MEMORY

Fig. 5
Internal allocation of the 16 address lines in the CDP1831 ROM.

PAUSE—Stops the internal CPU timing generator on the first negative high-to-low transistion of the clock. The on-chip oscillator, if used, continues to run but is ignored.

RUN—Normal operation where the CPU fetches and executes instructions.

While in the RUN mode, the CPU can be in one of four states: instruction fetch; instruction execute; DMA; and interrupt.

All instructions require a fetch and one or two execute cycles. Since each cycle consists of 8 clock periods, instructions require 16 to 24 clock periods, which translates to 2.5 or 3.75 μs at 6.4 MHz.

Support circuits

A full family of 1800-series support circuits, including memory and I/O, complement the CDP1802 microprocessor.

The RAM support circuits begin with the CDP1824 32-byte static CMOS RAM.

This device, in an 18-pin package, was developed specifically for microprocessor systems requiring a minimal amount of writable storage. The CDP1824 provides more than adequate working space and stack storage for a wide range of control applications. It is directly compatible with the CDP1802 microprocessor at maximum processor speed and requires no additional interface components. The CDP1824 is fabricated with the same silicon-gate CMOS technology as the CDP1802 and therefore has its same wide-temperature-range, low-power, and single-supply features.

Fig. 2 is a functional diagram of the CDP1824. The five address inputs, MA0-MA4, are buffered and decoded to uniquely select 1 of 32 bytes. The eight input-output data lines interface directly with the microprocessor data bus. Operation is determined by the state of the $\overline{\text{MWR}}$ and $\overline{\text{MRD}}$ inputs. For $\overline{\text{MRD}}$ = 0, the output drivers are enabled and the data stored in the addressed byte will be put on the data bus. For $\overline{\text{MWR}}$ = 0, the contents of the data bus will be stored at the addressed location. A chip-select input $\overline{\text{CS}}$ is provided to enable the memory.

The CDP1822 is a 256 x 4 static RAM.

It is functionally and pin-compatible with the 2101 industry type and is also pin- and performance-compatible with CDP1802. Fig. 3 is a functional diagram of the CDP1822. The eight address inputs, MA0-MA7, are buffered and decoded to uniquely select 1 of 256 four-bit words. The four data outputs are driven from the three-state drivers that are enabled by MRD when the device is selected. Normally, when used with the CDP1802, the four data-input and four data-output terminals are connected together and to the CDP1802 bidirectional data bus.

The CDP1822 requires no clock or precharge signals for proper operation and interfaces directly to the CDP1802 without additional components.

The CDP1831 is a static 4096-bit mask-programmable COS/MOS read-only memory organized as 512 8-bit words.

It interfaces directly with the CDP1802 without additional components. The CDP1831 responds to a 16-bit address multiplexed on 8 address lines (MA0-MA7), 8 bits at a time. Fig. 4 shows how the address lines are multiplexed and latched to form the 16-bit address. The most significant address byte is latched internally by either a positive or negative user-defined level of TPA. For compatibility with the CDP1802, this level is negative, as indicated in Fig. 4.

Fig. 5 shows the internal allocation of the 16 address lines. The seven most significant address lines select one of 128 512-byte sectors of the 64-kilobyte microprocessor memory space. The sector address is equivalent to a ROM-select and is user-programmable for a specific CDP1831.

The seven-bit ROM-select code is "anded" with the three chip selects (CS1, CS2, and $\overline{\mathrm{MRD}}$) to enable a given ROM. This signal is provided on an output pin (CEO) of the CDP1831 to indicate when the ROM is enabled.

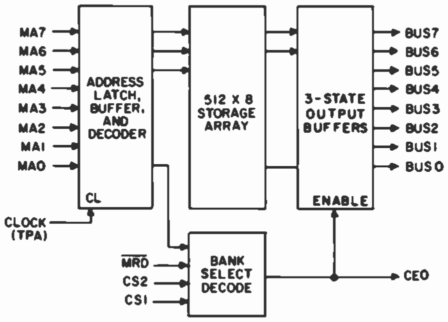
The nine least-significant address lines are decoded to select one of 512 bytes in the ROM (sector). In summary, the CDP1831 responds to a 16-bit address; this address specifies one of 128 memory sectors, which is fixed for a given ROM, and one of 512 contiguous bytes within the specified ROM.

There are three important features of the CD1831 that result from this addressing structure. First, it is directly compatible with the CDP1802 multiplexed address bus. Second, it is possible to make up a ROM system of from 512 to 64k bytes that is directly compatible with the CDP1802 and requires no address decoding. Third, when the ROM is selected, the CEO output goes high and can be used directly to disable a RAM system.

Reading data out of the CDP1831 follows a standard procedure. The device is selected by clocking the proper sector-address into the address latch. The tri-state output buffers are enabled by the proper combintation of the three chip-selects, CS1, CS2, and MRD. The chip-select enable polarities are user mask-programmable. Valid data will appear at the output within one access time (t_{AA}) from the last address change. Fig. 6 is a functional diagram of the CDP1831.

The CDP1832, like the CDP1831, is a static 4096-bit mask-programmable COS/MOS read-only memory organized as 512 8-bit words.

It is conventionally organized, requiring 9 address lines and a chip-select ($\overline{\mathrm{CS}}$) to read one of 512 eight-bit words into a bidirectional data bus. Data is available within one access time (t_{AA}) from the last address change. The chip select control ($\overline{\mathrm{CS}}$) functions as an output-disable. The CDP1832 is pin-compatible with the 512×8-bit erasable 2704 memory and can be directly inserted into a 2704 socket without any PC board changes.



CDP1831 FUNCTIONAL DIAGRAM

Fig. 6

Reading data out of the CDP1831 follows standard procedure to produce valid data within one access time from the last address change.

In addition to the CDP1831 and CDP1832, the following 8192-bit mask-programmable CMOS ROMs, organized as 1024 8-bit words, are in development: the CDP1833, with address latch; and the CDP1834, without address latch.

The 1800 family offers, in addition to RAM and ROM circuits, memory support circuits to simplify the design of memory systems.

The CDP1858 address latch/decoder provides all the control required to interface the CDP1802 microprocessor with up to 4 kilobytes of RAM built from 256×4 RAM circuits.

The CDP1859 address latch/decoder is similar in function to the CDP1858 except it is designed to interface the CDP1802 with up to 4 kilobytes of RAM configured from 1024×1 RAM circuits.

The CDP1852 byte I/O is directly compatible with the CDP1802 for use as an 8-bit address latch.

The CDP1856 memory-bus buffer/separator can function as a bus buffer for larger memory systems and as a bus separator for split-bus memory devices. It is directly compatible with the CDP1802.

I/O circuits

A number of general I/O support circuits are available and more that extend the CDP1802 I/O structure are in development. However, for a minimal system, the CDP1802 accommodates serial and parallel data transfer directly. In addition, the CDP1802 is compatible with the industry-standard CD4000 series COS/MOS family. The result is the ability to design any conceivable I/O interface with compatible devices. An outline of the RCA 1800 series of I/O circuits follows.

The CDP1852 is an 8-bit input/output port that simplifies parallel data transfers between the CDP1802 and external devices.

The CDP1852 (Fig. 7), when programmed as an input port (mode=0) or an output port (mode=1), will interface directly with the CDP1802 microprocessor without additional components. When used as an input port, data is strobed into the 8-bit register by a high (1) level on the clock line. The high-to-low transition of the clock latches the data in the register and sets the SR service-request output to 0. The three-state output drivers are enabled by CS1·CS2=1, and the high-to-low transition of CS1·CS2 resets $\overline{\mathrm{SR}}$=1.

When the CDP1852 is used as an output port, data is strobed into the 8-bit register when CS1·CS·CLOCK=1. The data is available at the outputs at all times because the three-state output drivers are always enabled when the mode input=1. The service request pulse is generated at the termination of $\overline{\mathrm{CS1}}$·CS2=1 and is present (high level) until the next high-to-low clock transition.

A CLEAR control is provided for resetting the port's register and $\overline{\mathrm{SR}}$ (input mode) or SR (output mode) signal. It is important to note that the polarities of service request ($\overline{\mathrm{SR}}$/SR) and chip-select 1 (CS1/$\overline{\mathrm{CS1}}$) change when the polarity of the mode input signal is changed.

The CDP1857 is an I/O bus buffer and bus separator.

It can be used for buffering the data bus, separating the bus data, and as an input or output port where data-latching is not required. The CDP1857 is a silicon-gate CMOS device in a 16-lead package.

In addition to the above devices the following circuits are in development for later availability:

CDP1851 Programmable I/O
CDP1853 N-bit decoder
CDP1854 UART (Universal Asynchronous Receiver-Transmitter)

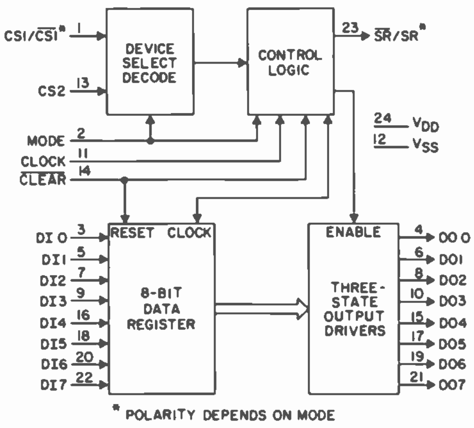


Fig. 7
Input/output port (CDP1852) performs parallel data transfers between the CDP1802 and external device directly.

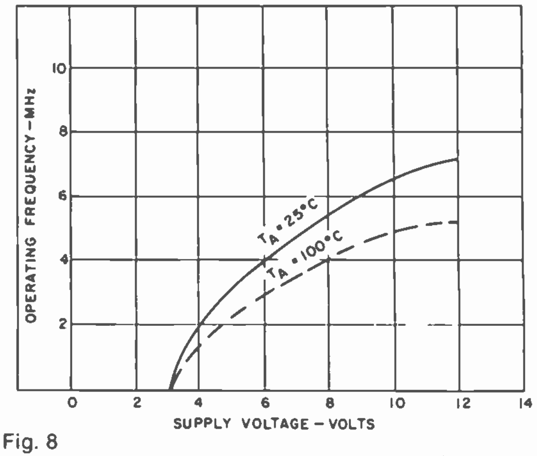


Fig. 8
Maximum clock frequency of CDP1802 is a function of supply voltage and temperature.

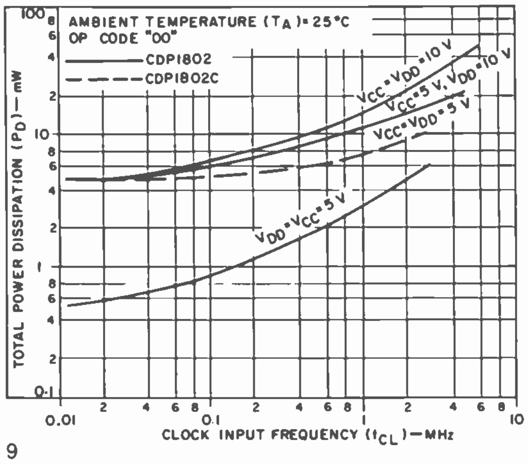


Fig. 9
Power dissipation for CDP1802 varies with input frequency and temperature.

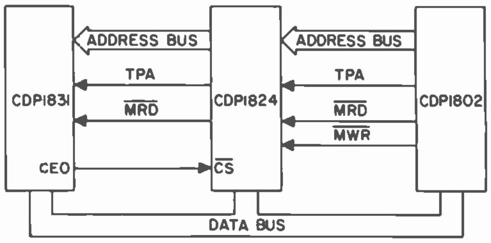


Fig. 10
Minimal COSMAC microcomputer system has only three chips.

Putting a system together

Designing a microcomputer based on the CDP1802 is a straightforward procedure. Generally, any microcomputer consists of a CPU, a variable amount of memory consisting of RAM and ROM, and an I/O interface. The design procedure consists of determining general system requirements or constraints, sizing and allocating RAM and ROM, including address space, and I/O design. Once the general microcomputer design has been mastered, the designer can focus more and more attention on the I/O, including the interaction of I/O hardware and software to minimize cost.

The first step in designing a microcomputer is to determine general requirements.

Specifically, operating speed, supply voltage, operating temperature range, and power dissipation must be established.

For a system requiring maximum performance, operating speed must first be established. Operating speed (or clock frequency) will then fix the instruction time. For the CDP1802, all instructions require 16 clock periods, except long branches and long skips, which require 24 periods. Instruction time determines processor throughput and eventually I/O response time and resolution.

The CDP1802 maximum clock frequency is a function of supply voltage. Therefore, once the clock rate is established, the minimum operating voltage can be determined from Fig. 8. For systems that are not throughput limited, Fig. 8 can also determine a convenient clock rate and minimum supply voltage.

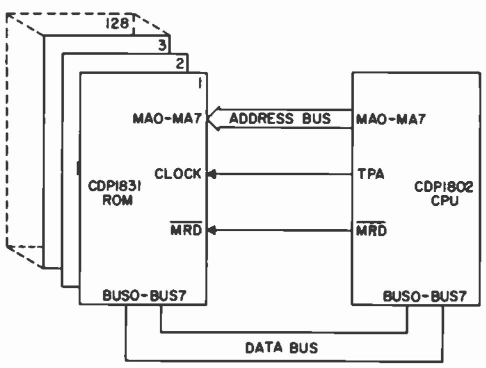
Next, the operating temperature range must be established. The processor clock rate is temperature-dependent and must be derated by 0.35%/°C. This dependency must be factored into the overall clock-rate/supply-voltage requirement. Fig. 9 shows power dissipation as a function of input frequency at 25°C. To account for temperature variations, a convenient approximation is to add 20 μW/°C. This factor is the increased quiescent dissipation, which is temperature dependent.

This example illustrates how to go about microcomputer design.

Design a microcomputer meeting the following system requirements:

1) Add two eight-bit bytes in 8 μs
2) Operation from 25 to 100°C
3) Supply voltage must not exceed 7 V

Because the processor has an ADD instruction that adds M(R(X)) to D, one instruction will satisfy the 8-μs add requirement. Clock frequency is $1/(8 \times 10^{-6} \div 16) = 1/500 \times 10^9 = 2$ MHz. Therefore, the microprocessor must operate at 2 MHz at 100°C. To determine the minimum supply voltage, derate the 25°C curve in Fig. 8 by (100°C − 25°C) (.35%/°C) = 26%. At 2 MHz, the supply voltage from the derated curve is 4.8 V. As a result, the CPU voltage can be selected between 4.8 and 7 V, depending on the desired margin and power limitations. Assuming a 5-V supply is chosen, the power dissipation can be read from Fig. 9 as 5 mW for the CDP1802D and 9 mW for the CDP1802CD, both at 25°C. For 100°C, these values are increased by (100°C − 25°C) (20 μW/°C), or 1.5 mW.



ALL CDP1831 INTERFACE CONNECTIONS ARE COMMON TO ALL ROM'S.

Fig. 11
ROM expands easily from 512 bytes to 65,536 without address decoding.

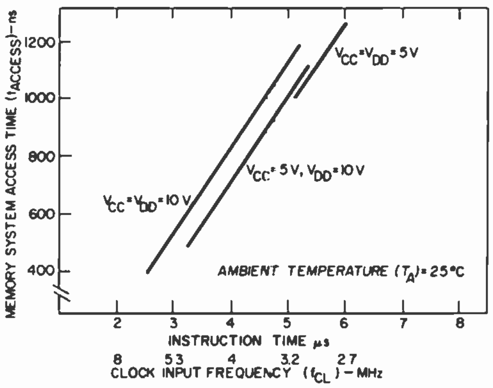


Fig. 12
Memory-system access time must be compatible with the microprocessor. Required access times for specific instruction times and clock frequencies are shown here.

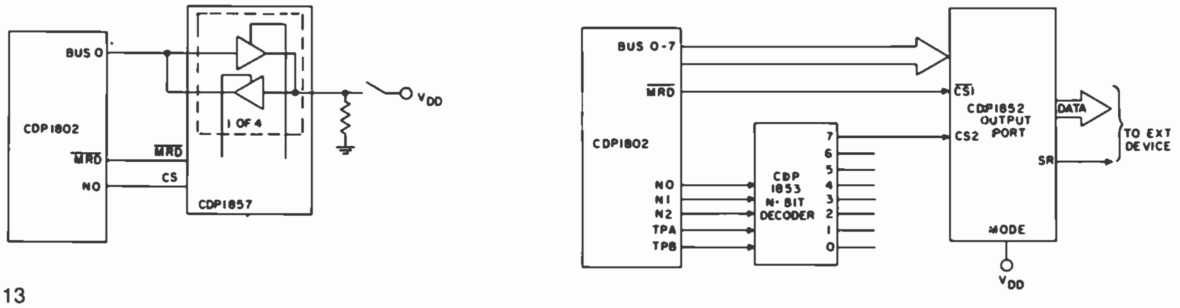


Fig. 13
Programmed I/O data transfer is possible with the CDP1802. Three ports can be selected directly, 14 with the addition of an N-bit decoder, and the number becomes virtually limitless with a two-level I/O convention. Diagrams show simple binary input port (left) and decoded parallel output port.

The designer should try to minimize variable-storage RAM and maximize invariable-storage ROM for the most cost-effective design.

The design of COSMAC microcomputers starts with the minimal system shown in Fig. 10. The memory system consists of 32 bytes of RAM (CDP1824) and 512 bytes of ROM (CDP1831). These devices are directly compatible with the CDP1802 and require no additional devices to interface with the processor. The address space for the system is determined by the user when he programs the CDP1831. The seven-bit user-specified sector address (seven most significant bits of the 16-bit address multiplexed to the CDP1831 8 bits at a time) locates the CDP1831 in one of 128 512-byte memory sectors. When the proper sector address is recognized by the CDP1831, the CEO output goes high and disables the RAM. The RAM, which is enabled when the ROM is not selected, occupies all 256-byte memory pages except the two occupied by the ROM.

Building onto the minimum system is also straightforward. Fig. 11 shows how the ROM can be expanded from 512 bytes to 65,536. No address decoding is required and the system is directly compatible with the CDP1802. RAM can be added as before and enabled with the "OR-ed" CEO outputs of the CDP1831 ROMs.

The CDP1824, CDP1822, and CDP1831 memories are compatible with the CDP1802 at maximum processor speed. However, for large memory systems, the memory-system access time may not be speed-compatible with the CDP1802. To assist in determining memory-system speed requirements for the CDP1802, Fig. 12 shows the required memory-system access time for a specific instruction time or clock frequency.

The I/O section of a microcomputer is the least structured area of the system.

However, the I/O interface for the CDP1802 has been designed for flexibility, ease of use, and low cost. The simplest way to bring data into the CPU is via the four external flags, EF1-EF4. These inputs can accommodate independent serial data links, a terminal for example, or can be used together to input 4-bit data characters. The actual transfer of data is under program control. Branch instructions test the flag inputs and branch on the appropriate condition. Loops can be constructed to continually test the flag lines and manipulate the data as required.

Complementing the four input flags is a single-bit output port, Q, which can be set, reset, and tested by branch instructions. Under software control, Q can be used as a pulse-width-modulated, variable-frequency, or serial-data output port.

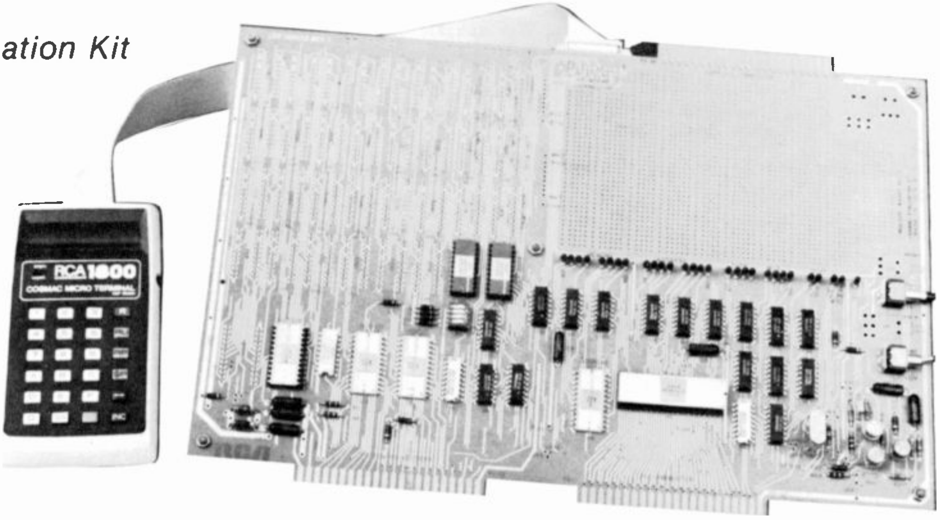
The CDP1802 also offers a programmed I/O data-transfer mechanism.

The processor I/O instructions are of the 6N format, where N is a variable input or output device-select code. The three least-significant bits of the binary N-code are available at dedicated processor pins for device selection. The CDP1802 can therefore select up to three I/O ports without additional components. This number increases to 14 by using the CDP1853 N-bit decoder; by adopting a two-level I/O convention, the number of external devices becomes virtually limitless. Examples are shown in Fig. 13. It should be noted that the N lines can be used as outputs in some systems.

The I/O design can become increasingly more sophisticated by using the built-in DMA and interrupt mechanisms in conjunction with the previously described I/O interfaces. State-code outputs are provided to indicate when the processor is in the DMA or interrupt states.

A low-cost hardware prototyping system for microprocessor design

The Microterminal/Evaluation Kit combination is a convenient low-cost design tool for CDP1802-based microcomputer systems.



D. Block

Dennis Block became one of the early members of the RCA COS/MOS activity with his transfer to the Solid State Division in 1968. His work there included design of both standard and custom circuits and applications of the CD4000 family. Presently he is working in the microprocessor area with responsibility for applications assistance on standard products, support development and documentation, and related marketing activities.

Contact him at:
Microprocessor Applications Engineering
Solid State Division
Somerville, N.J.
Ext. 6046

Reprint RE-22-5-14
Final manuscript received December 18, 1976.

A hardware prototyping system is a "must" in microprocessor system development. It is the equivalent of the breadboard in traditional logic design and has the same purpose—testing and verifying a design before it is committed to production.

RCA is supporting the CDP1802 microprocessor with a wide variety of prototyping systems for both hardware and software development. This paper describes a new low-cost microprocessor system and portable terminal designed for users of RCA 1800-series components or anyone with limited resources to invest. The system is also suitable for the hobbyist.

The system consists of the COSMAC Evaluation Kit, which is a microcomputer system based on the CDP1802 microprocessor, and the Microterminal for I/O communications. A user-supplied 5-V, 1-A power supply turns this system into a complete microcomputer with substantial capability and considerable flexibility. Although each of the two units can be used separately, together they open the door to expanded areas of microcomputer applications.

The COSMAC Evaluation Kit

An effective hardware-prototyp:ng system should meet several criteria. First, it should be readily expandable, both in memory and in I/O capability. It should also provide a means of program entry and verification, plus debugging features such as single-step and CPU register readout.

The Evaluation Kit meets these requirements. A kit of components for building a microcomputer based on the CDP1802 COSMAC microprocessor, it contains a PC board, CPU, byte input and output ports, terminal interfacing, a ROM-based utility program, and a RAM for user program storage. The RAM memory supplied is 256 bytes, but the PC board is prewired with memory addresses and chip-select signals so that up to 4 kilobytes of RAM can be accommodated by adding more CDP1822 memories. A full 16 bits of memory addressing is provided. Also, a battery backup option is made feasible by the use of CMOS memories.

A 6" × 4" area of the board is free for user-added I/O devices. ICs of various pin counts can be inserted into prepared positions in this area and jumpered to an uncommitted 44-pin connector on the board. Two other 44-pin connectors provide access to all CPU and system signals.

Several options exist for terminal interfacing. Both a conventional 20-mA current loop and an EIA RS232 interface at 110 to 1200 baud are available, which, along with a ROM utility program, provide

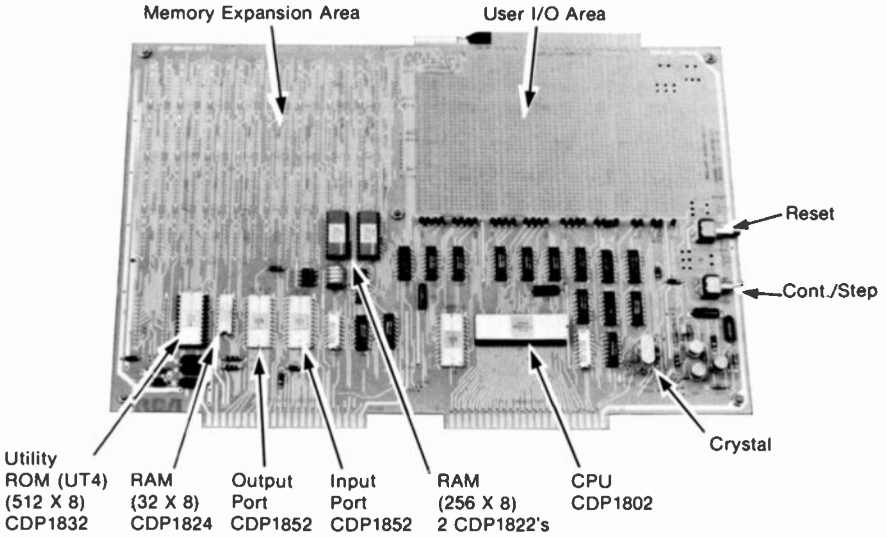


Fig. 1
Assembled Evaluation Kit PC board shows the large amount of space available for user I/O options and memory expansion.

automatic interfacing to ASCII serial terminals. A third option is the RCA Microterminal, which was designed to perform commonly required functions such as a memory inspection and modification, as well as numerous other functions to be described later.

Fig. 1 shows an assembled Evaluation Kit board; more detailed descriptions of the CDP18S020 system components follow.

The heart of the microcomputer system is the COSMAC CDP1802 microprocessor.

The CDP1802 is a CMOS LSI microprocessor that offers the advantages of CMOS technology, including full military temperature range, wide 3- to 12-V operating range, low power consumption, excellent noise immunity and fully static operation. It also has the unique COSMAC architecture, an efficient structure of general-purpose and control registers designed for a compact instruction format and flexible I/O interface.

The Evaluation Kit has been designed to assist the user in understanding the basic CDP1802 architecture as well as in applying the I/O control and data signals provided by the microprocessor. One feature of the Evaluation Kit is direct access to all CPU terminals; this means that the user can monitor CPU operation or provide an input for a CDP1802 remote emulation function.

A detailed description of the CPU architecture and instruction set is provided in the *User Manual for the RCA* CDP1802 *COSMAC Microprocessor*, MPM-201A, supplied with the Evaluation Kit. Electrical specifications are given in the CDP1802 data sheet.

The CPU clock is generated by an on-chip crystal-controlled oscillator.

There are eight clock pulses per machine cycle and two or three machine cycles per instruction. A 2-MHz crystal is supplied; however, the user can design systems with much faster instruction times. For example, a 6.4-MHz clock frequency at 10 V results in instruction times of 2.5 and 3.75 μs, and other instruction times can be obtained as a function of supply voltage and crystal frequency. There is no minimum frequency requirement because the CDP1802 is an entirely static device. Clock frequency changes can be made by replacing the provided crystal with one of the desired frequency. Alternatively, an external clock can be brought in via the CPU connector.

The control section initiates program execution and facilitates hardware and software debugging.

It has four controls with associated logic: RESET; RUN U (run utility); RUN P (run program); and CONTINUOUS/STEP.

These four controls generate signals that are connected to the $\overline{\text{CLEAR}}$ and $\overline{\text{WAIT}}$ pins of the CDP1802. These two pins in turn control the four modes of operation for the microprocessor: LOAD, RESET, PAUSE, and RUN.

The Evaluation Kit control logic has been designed to permit operation of the system in two general modes: CONTINUOUS—normal run; and STEP—single cycle. By switching to the STEP mode and continuously pushing RUN P the user can sequence his program one machine cycle at a time and watch the system action on the LED display.

Discrete LED displays provide a visual indication of CPU operations so that program bugs can be spotted.

Displays are provided on the memory address bus, data bus, and CPU status signals. They can be used with the Microterminal to display the contents of 15 of the CPU's general-purpose registers, as discussed later. The LEDs go ON to indicate a logic 1 on the line being monitored and are OFF for a logic 0 or floating data-bus condition, because the data bus is provided with pull-down resistors.

The CDP1802 state code lines, monitored by LEDs, uniquely define the four microprocessor machine states: FETCH, EXECUTE, DMA and INTERRUPT. This information is valuable in debugging operations.

The Evalution Kit is provided with an extensive, expandable memory system.

The memory consists of:

- 256-byte RAM prewired for expansion to 4096 bytes;
- 512-byte utility-program ROM;
- 32-byte utility-program RAM; and
- an optional 512-byte prewired ROM location.

The memory system interfaces with the CDP1802 microprocessor via address-latch, decode, and read/write control logic. Figs. 2 and 3 show the memory system in block-diagram and memory-map form.

The CDP1802 microprocessor itself is capable of directly addressing 64 kilobytes

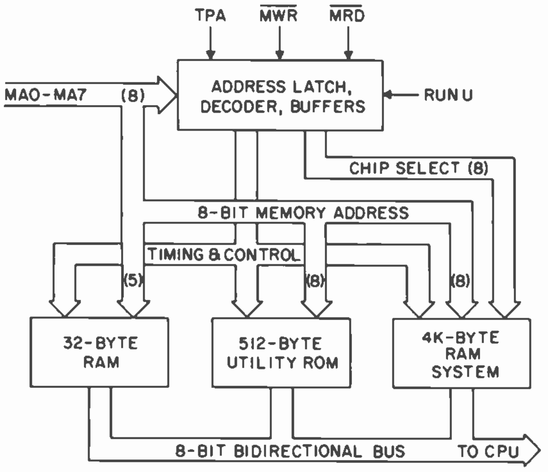
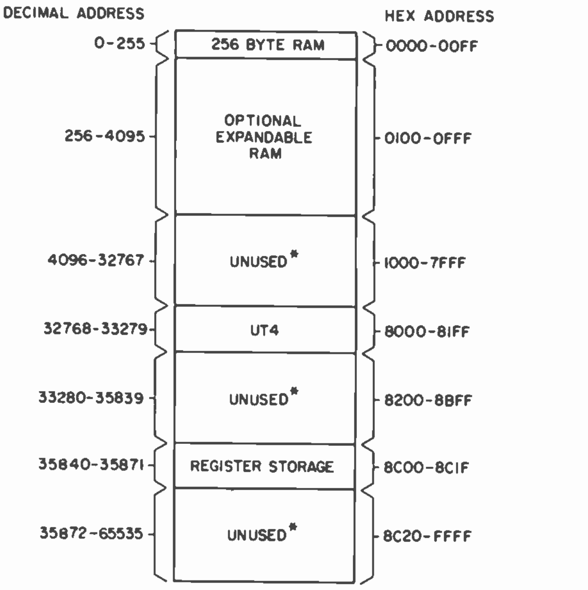


Fig. 2
Memory system for the Evaluation Kit is expandable to 4K of RAM and 512 bytes of prewired ROM.



* THESE AREAS REQUIRE ADDITIONAL ADDRESS DECODING BEFORE USE.

Fig. 3
Memory map for Evaluation Kit. UT4 is a utility progıam in ROM.

of memory in any combination of RAM or ROM without restriction. The 16-bit address is multiplexed 8 bits at a time on the 8 memory address lines from the CDP1802. The CPU generates a timing pulse, TPA, during which the most-significant memory-address bits are present on the 8 address lines. TPA is used to latch the address bits required to form the full memory address. One-half clock period after the termination of TPA, the 8 least-significant address bits are multiplexed onto the address bus and remain valid for the remainder of the machine cycle.

The Evaluation Kit uses a CDP1852 I/O chip to latch the most significant byte with TPA. The remainder of the memory-system control logic is used to decode the 16-bit address for memory-chip selection.

The Evaluation Kit is provided with a 256-byte CMOS RAM consisting of two CDP1822 256×4 RAMs. The RAM system has been prewired for expansion to 4096 bytes by simple insertion of additional CDP1822s in designated locations. The RAM system is assigned the first 4096 locations of the 64-kilobyte system as indicated in the memory map of Fig. 3.

The Evaluation Kit also has a "write protected" RAM in 1-kilobyte segments. The memory write control ($\overline{MWR}$) is connected to the RAM system through four DIP switches. When the switches are closed, the RAM system operates normally for both read and write operations. When a switch is open, $\overline{MWR}$ is inhibited from controlling the specified 1-kilobyte RAM segment and forces the RAM to a "read-only" mode.

The ROM system consists of a CDP1832 512-byte ROM factory-programmed with the utility program UT4. The CDP1832, a static CMOS device, is addressed by nine address lines and puts data on the bidirectional data bus within one access time of the last address change when selected. The Evaluation Kit design locates the utility program in the upper half of memory. UT4 allows the user, via teletypewriter or similar conventional terminal, to inspect and modify memory and to start program execution at a given location. It also handles terminal interfacing, including automatic speed adjustment. When started, the utility program will save 13-1/2 of the CPU's registers in its associated RAM, where they can be subsequently read out to determine CPU status just before UT4 was entered.

A CDP1824 32×8 CMOS RAM provides a register-save feature and scratchpad area for the Utility Program. It is selected by "ANDing" MA15 (8000) with a decoded 0C00 address state.

The I/O section has been designed for flexibility, accessibility, and easy experimentation.

The Input/Output section of the Evaluation Kit consists of an 8-bit parallel-output port, an 8-bit parallel-input port, and a 1-of-8 decoder for selecting I/O devices. The output port is designed to respond to CDP1802-programmed output-data transfers. The microprocessor I/O instructions are all of the 6N format, where N is a variable-input or -output device-select code. For output instructions, N is 1-7. The binary N code is available on CPU pins 17 through 19 for device selection.

$\overline{MRD}$ is used to indicate whether the operation is an input or output; $\overline{MRD}$=0 for output instructions and $\overline{MRD}$=1 for input instructions. Therefore, gating $\overline{MRD}$ with the N-lines uniquely defines device selection and direction.

Fig. 4 shows the circuit configuration for the output port. The three CDP1802 N lines are connected to the 1-of-8 N-bit decoder. Output "5" (pin 6) of the decoder is connected to the CDP1852 chip-select input CS2 and used to select the port. The CDP1802 $\overline{MRD}$ (memory read) output is connected to $\overline{CS1}$ of the output port and used as the second select input. Output data transfers occur between memory and the output device. Data from the output port is available at the system connector, as indicated in Fig. 4. It should be noted that the N-bit decoder is not necessary for three or less I/O devices. For these cases, an N-line can be connected directly to the I/O-port chip-select input.

The input port, which operates in a manner similar to the output port, is designed to respond to CDP1802-programmed-input-data transfers. Input instructions are of the format 6N, where N is 9-F. In analyzing the seven N codes, it can be seen that the three least significant N bits range from 1-7, as in the output instructions.

The circuit configuration for the input port is shown in Fig. 5. Output "6" (pin 7) from the N-bit decoder is connected to the CDP1852 chip-select input CS2 and used to select the port; $\overline{\text{MRD}}$ is connected to the CS1 input and used as a second (direction) select input.

Operation is as follows. Data is loaded into the port by an external device clock. The data inputs and clock signal are available on system connector pins as indicated. The negative CLOCK transition sets the service request flip-flop (SR) to 0. The $\overline{\text{SR}}$ signal can be connected to either $\overline{\text{EF3}}$ or $\overline{\text{INTERRUPT}}$ of the CDP1802, allowing either an interrupt service routine or a software loop to respond to the service request for an input data transfer. The CPU, by executing a 6E instruction, will enable the input port onto the data bus and load memory and CPU register D with the latched data. At the completion of the 6E instruction execution, the SR signal will be reset to 1.

The TTY reader and printer interface electronics can be configured for either 20-mA current loop or EIA RS232C compatibility. All TTY connections are available at the system connector pins.

The user I/O area is a special feature of the Evaluation Kit.

This area is intended for constructing user-defined memory and I/O systems. It will accommodate any combination of 8-, 14-, 16-, 18-, 22-, 28-, or 40-pin dual-in-line packages. For example, the area can be used to assemble a system consisting of eighteen 16-pin packages or twelve 24-pin packages. To assist in assembling custom memory and I/O systems, convenient access to the CPU data bus (BUS0-7), memory address bus (MA0-7), and TPA, TPB, and $\overline{\text{MRD}}$ signals is provided on the left side of the user I/O area. The area also has an uncommitted user-I/O connector for interfacing with external systems, eighteen LED locations, and two switch locations.

The user I/O area plays an important part in customizing the Evaluation Kit to a specific application. Used together with the prewired memory and control functions, it enables the user to quickly construct and debug prototype microprocessor systems.

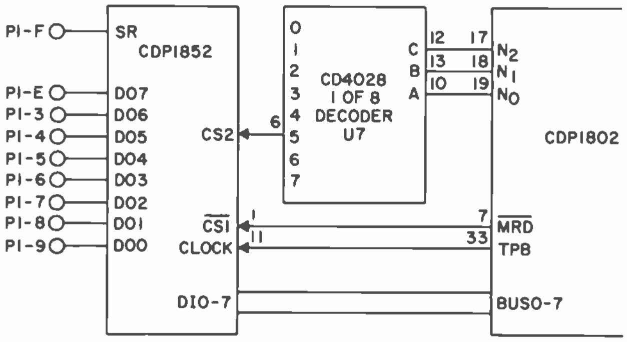
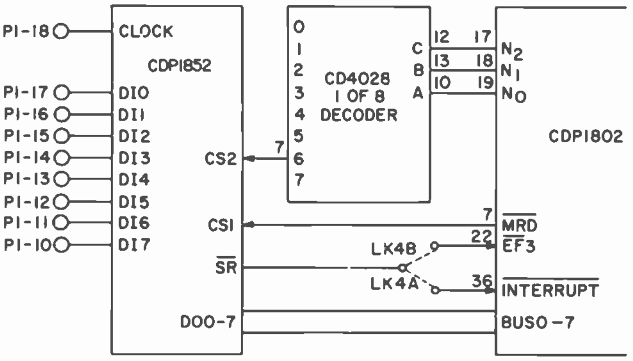


Fig. 4
Output-port circuit uses N-bit decoder where there are more than three I/O devices. Otherwise, N-line is connected directly to the I/O-port chip-select input.



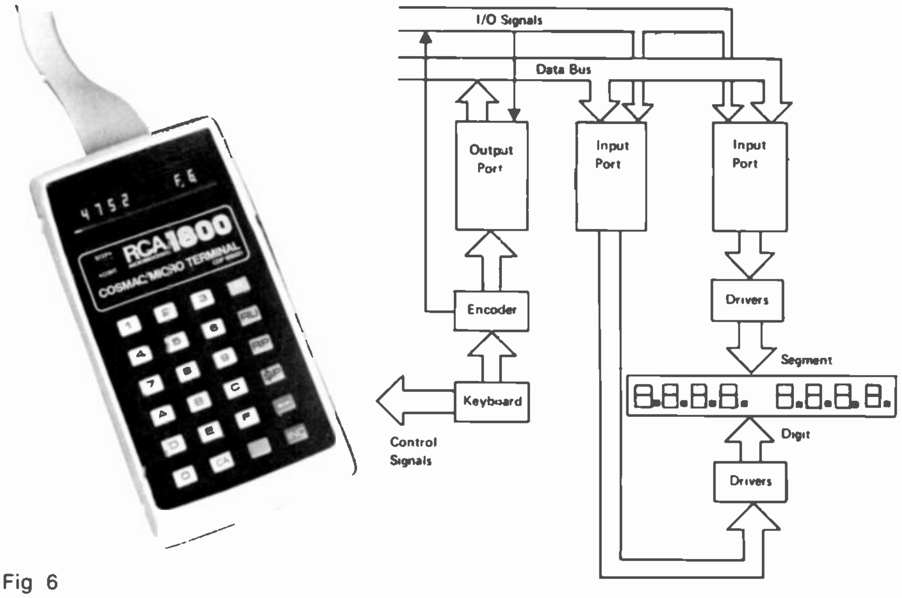


Fig 6
Hand-held Microterminal is a low-cost alternative to the teletypewriter data terminal. It controls the microprocessor and displays registers and memory locations, making it an excellent debugging tool.

The Microterminal

The Microterminal (Fig. 6) is a small, portable, low-cost data terminal especially suitable for use with the COSMAC Evaluation Kit; it can also be used with a comparable user-designed system. The Microterminal consists of a keyboard and display unit, with its connecting cable and mating connector, and a ROM containing the utility program to run the terminal and various subroutines that user programs can access. The Microterminal as shown is divided into three functional sections: control, display, and keyboard.

Table I
Read memory location 801F.

Enter	*Press*	*Display*		*Comment*
	R	—		Reset the CPU.
	RU	0.0.0.0.	XX	Start UT5.
8		0.0.0.8.	XX	Enter desired address, 801F.
0		0.0.8.0.	XX	
1		0.8.0.1.	XX	
F		8.0.1.F.	F8	Contents of 801F now displayed.
To read continguous addresses, press the INC button.				
	INC	8.0.2.0.	35	Contents of 8020 is 35.
	INC	8.0.2.1.	A6	Contents of 8021 is A6.
	etc.			

X denotes an unspecified hex character.

Table II
Change the data byte at location 20 to 00. (Assume an initial arbitrary state with UT5 already running.)

Enter	*Press*	*Display*		*Comment*
		XXXX	X.X.	Arbitrary initial conditions.
	CA	0000	X.X.	Clear address.
	↔	0.0.0.0.	XX	Go to address-entry mode.
2		0.0.0.2.	XX	Begin address entry.
0		0.0.2.0.	X.X.	Address established.
	↔	0020	X.X.	Go to data-entry mode.
0		0020	X.0.	Begin data entry.
0		0020	0.0.	Data established, but not written.
	INC	0021	X.X.	Data has now been written to memory.

The control section can reset logic, run programs, and increment addresses.

The function keys are:

R	Reset: resets the logic of the Microterminal and the microprocessor system.
RU	Run Utility: starts execution of the Utility Program (UT5) at location 8000.
RP	Run Program: starts program execution at location 0000.
CONT	Continuous/Step: enables continuous or single-step operation of the microprocessor system.
↔	Entry Mode Control: toggles the mode between address-entry and data-entry.
INC	Increment Address: increments the address shown in the display. In the data-entry mode, it also has the data byte shown written into the address shown.
$P	Start Addressed Program: starts program execution at the location shown in the address display.
CA	Clear Address: Clears (resets) the address display to 0000.

The R, RU, RP, and CONT/STEP controls perform the same functions as the corresponding switches on the Evaluation Kit.

The keyboard section of the Mictoterminal contains 16 keys that enter hexadecimal numbers into the address or data field.

The entry-mode toggle switch controls the destination of entered data. The hexadecimal keys and some of the control keys are encoded and sent to the microprocessor system on a bidirectional data bus; logic scans and signals keyboard activity to the microprocessor. The actual scanning, debounce, and decoding algorithms are performed by software routines in the utility program.

The display section consists of an eight-digit seven-segment LED display, display drivers, and refresh logic.

The display shows a four-digit address field on the left and a two-digit data field on the right; the two fields are separated by blank positions. In other subroutine-oriented display modes, all eight digits are available. Decimal points adjacent to either the address or data digits indicate whether the display is in the address-entry or data-entry mode.

A separate 5-V supply terminal is provided for the LEDs and their drivers. The rest of the logic is supplied from V_{CC}, which may range from 5 to 12 V.

The Microterminal interface is controlled by utility routines in its associated ROM.

These routines, collectively known as UT5, handle keyboard-scanning, debouncing and decoding, display-code conversion and multiplexing, standard modes of terminal operation, and display-function subroutines addressable by a user program. This ROM replaces the UT4 Utility ROM supplied with the Evaluation Kit for interfacing to teletypewiters or similar terminals.

There are three basic operations that can be performed with the Microterminal: memory read, memory write, and CPU register readout.

When the Microterminal is in the address-entry mode, the contents of memory at the location shown in the address field are displayed as a two-digit hexadecimal byte in the data field. When the utility program is started, it puts the terminal in the address-entry mode, denoted by lighted decimal points in the address field. It is then only necessary to enter the desired address by pressing the hexadecimal keys. Numbers are shifted in from right to left. See Table I.

When the Microterminal is in the data-entry mode, the byte in the data field is written to the address field location only when the INC button is pushed. The data-entry mode is signified by lighted decimal points in the data field. Hexadecimal numbers are entered in the data field from right to left. See Table II. To skip a location while entering data, simply increment past the unaltered locations.

Registers R1 through RF can be read out at the point where a user program is halted.

When this feature is used with a "planted" IDLE instruction, it provides the means for implementing an elemetary breakpoint for debugging purposes. Assume a breakpoint is required at location XXXX of the user program so that CPU registers and certain memory locations can be examined there. First an IDLE instruction would be written into location XXXX before starting the program. The user program will idle when it reaches that location, and registers can then be sequentially displayed in the address LEDs by repeatedly pressing RU while in the STEP mode.

Microterminal utility programs contain extra subroutines as a convenience for the user.

The Microterminal utility program, UT5, contains two main sections: a routine called KEYUT, which takes care of terminal interfacing, and a group of user-oriented subroutines.

KEYUT performs two main tasks: periodically refreshing the display; and scanning for keyboard inputs and executing the required actions.

KEYUT spends most of its time in a loop of refreshing the display and waiting for a key depression. When a key is pressed, the program decodes it and performs the required action. KEYUT waits for the key to be released (refreshing the display in the meantime) before returning to the main loop.

As a convenience to the user, the ENTRY subroutine initializes the CPU registers necessary for the COSMAC standard call and return technique. This technique allows multiple levels of subroutine nesting by using a stack in RAM. The call and return subroutines themselves are included on the ROM (UT5).

Thus, a long branch to ENTRY as the first three bytes of a user program saves writing the initialization code otherwise required to establish the standard call and return technique.

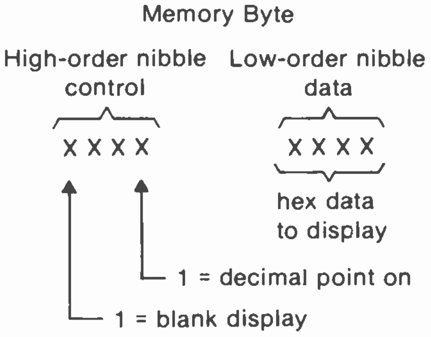
The COUNT subroutine is an independent program that displays memory sequentially starting from location 0000, incrementing the address approximately once per second. This subroutine provides an automatic read-back of a user program previously entered.

Two subroutines are provided to output data to the display via a user program. The subroutine REGDIS sends the contents of registers RA and RB to the display. Each register is displayed as a four-digit hexadecimal number with RA appearing at the left of the display.

Table III
Display registers RA and RB; increment RA.

Enter	*Press*	*Display*	*Comment*
	R		
	RU	0.0.0.0. X X	Start UT5.
	⟷	0000 X.X.	Go to data-entry mode.
C0		0000 C.0.	Set up a long
	INC	0001 X.X.	branch to ENTRY.
81		0001 8.1	
	INC	0002 X.X.	
08		0002 0.8.	
	INC	0003 X.X.	Go to location 0005,
	INC	0004 X.X.	the return point
	INC	0005 X.X.	from ENTRY.
D4		0005 D.4.	Call to REGDIS.
	INC	0006 X.X.	
81		0006 8.1	
	INC	0007 X.X.	
A6		0007 A.6.	
	INC	0008 X.X.	
1A		0008 1.A.	Increment RA.
	INC	0009 X.X.	
30		0009 3.0.	Loop back to 0005.
	INC	000A X.X.	
05		000A 0.5.	
	INC	000B X.X.	Program is now loaded. To start it running, press
	R	—	
	RP	$\underbrace{XXXX}_{RA}\underbrace{XXXX}_{RB}$	

The other subroutine, called LEDD, is more general-purpose. It allows user control of all eight digits of display, plus their decimal points. LEDD reads out eight consecutive bytes of memory, starting at the location pointed to by RF, interpreting the bits at each location as a control and data character as follows:



Unused display positions are specified as blanks (8X) in the appropriate memory positions. LEDD leaves the data pointer RF at its initial value when it exits.

Table III demonstrates the use of REGDIS, one of the UT5 subroutines.

Conclusions

The Microterminal-Evaluation Kit combination offers a low-cost approach to microprocessor design. For designers of high-volume systems, it represents an attractive prototyping tool, and it may be the end product for the hobbyist or low-volume user. The Microterminal's portability suggests numerous other independent uses for it, such as a field service tool or as a terminal for games or small test equipment, remote data entry, teaching devices, etc. The Evaluation Kit can also be used independently or with a more sophisticated terminal such as a CRT to perform as a compact yet powerful single-board computer. But together these two components are a synergistic marriage of the best features of the microprocessor revolution.

Acknowledgments

The author wishes to acknowledge his debt to many, but particularly to A.W. Young, C.T. Wu, and R.J. Sedlak for their major contributions to the development of the systems and equipment described in this article.

Architectural trade-offs in designing microcomputer systems

P.M. Russo

Should it be ROM or PROM, one-level or two-level I/O, assembly language or a higher-level language, single-processor or multiprocessor, master-slave or master-master, ...?

Paul Russo joined RCA Laboratories in 1970; since then he has done research in computer architecture, program behavior, system performance evaluation, microprocessors, and microprocessor applications. He was involved, from the outset with the development of the COSMAC microprocessor and the original FRED system. Dr. Russo has taught several microprocessor courses and was the recipient of a Laboratories Outstanding Achievement Award in 1974.

Contact him at:
Systems Research Laboratory
RCA Laboratories
Princeton, N.J.
Ext. 3231

Reprint RE-22-5-17
Final manuscript received October 25, 1976.

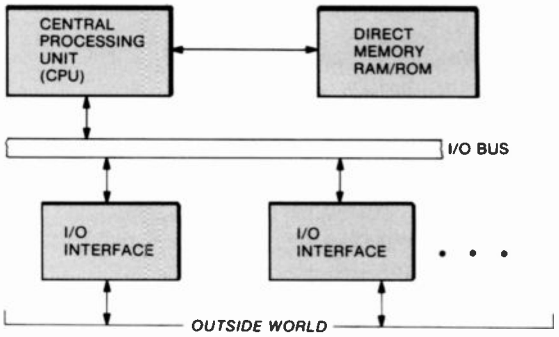


Fig. 1
Any computer system has these basic subsystems.

Microcomputer design resembles standard computer design in that it involves a number of choices and trade-offs. So, before one can address the general topic of microcomputer system design, the various subsystems associated with any computer system must be identified. Fig. 1 illustrates the three basic elements that make up any single-processor system:

> The *CPU* performs all the ALU and central control functions associated with the computer system.
>
> The *memory* contains both the program and the current data on which the CPU is operating.
>
> The *I/O interfaces* are the key elements of any computer system, since they represent the only communication channels between the internal computer functions and the outside world (the I/O devices).

Any computer system, be it a large data processing center or an automobile ignition controller, is made up of the above three subsystems. However, differing applications dictate different I/O devices, different levels of performance, and different cost requirements. These, in turn, result in vastly different CPU, memory and I/O interface requirements, different I/O architectural trade-offs, and possibly even entirely different device technologies.

Where the difficult trade-offs lie

This paper briefly examines the various architectural considerations that go into the design of microprocessor-based systems. The topics discussed include the basic system design steps; one-, two-, and multi-level I/O structures; the classical hardware/software trade-offs in interface design; and low- *vs.* high-volume product hardware/software cost trade-offs. Finally, there is a brief overview of multi-microprocessor structures, one of the new frontiers in microprocessor system architectures.

In general, selecting the appropriate device technology, CPU and memory organizations, and I/O devices is less difficult than making the subtle hardware/software trade-offs associated with the I/O architecture. Yet the I/O architecture is usually fundamental to the success of a system design and will dramatically affect cost and performance.

Microcomputer system design

The selection of any subsystem, be it CPU, memory, or I/O, cannot be made without examining the interrelationships that exist between all of them. However, since selecting the CPU and memory subsystems is relatively simpler than specifying the I/O architecture, the latter will be given greater emphasis throughout this article.

The properties of any CPU fall into three broad categories—functional architecture, chip architecture, and technology—whose importance depends on the application.

The CPU *functional architecture* is the key to its data handling and control capability. It includes features such as:

Word size (e.g., 8 bits);
Memory address range (e.g., 64K words);
Instruction set; and
I/O facility.

The CPU *chip architecture* defines the signals available or needed at the various chip pins. Hence, it defines both the amount of external logic the CPU needs to communicate with the outside world and the amount of logic and number of voltage levels required for basic CPU operation. The chip architecture will primarily influence cost, since it directly affects the ease of interfacing with various types of memory and I/O devices and defines the complexity of external clock-generation circuitry.

The CPU *technology* defines the chip density, power dissipation, drive capability, noise immunity and, usually, machine-cycle time. These factors affect chip cost (yield, chip size), system throughput (chip speed), and system cost (power supplies, external buffers to drive I/O devices, etc.).

Obviously, it is impossible to specify the characteristics of an optimal CPU *a priori*, since most desirable attributes, even within any one CPU category, are inherently conflicting. As an example, for a given number of pins, word size can be traded against memory addressing range, but both cannot increase simultaneously without affecting the amount of external support logic required and hence system cost. Excellent articles on microprocessor selection are available in the literature.[1,2]

Memory selection, besides satisfying CPU speed requirements, is dictated by software needs (must software be changed often?) and the intended volume of the application.

In general, all microprocessor-based systems require some RAM and some type of read-only memory. The system designer has a host of memory types to choose from, each of which can usually be implemented in a variety of technologies. The following is but a partial list:

Random-access read/write memory (RAM);
Mask-programmed read-only memory (ROM);
Programmable read-only memory (PROM);*
Erasable programmable read-only memory (EPROM);**
Electrically erasable programmable read-only memory (EEPROM).

Additionally, non-volatile memories (ones that will maintain data with power off) also exist (e.g., MNOS), but these are usually not suitable as main memories because of cost and speed considerations.

In applications requiring flexibility and/or low volume, PROMs or EPROMs are often the best choice for read-only storage. For large-volume dedicated applications, where the program is frozen for relatively long time periods, ROMs usually offer the lowest-cost alternative.

The importance of the I/O architecture to the cost-effective implementation of microcomputer systems cannot be overstated.

I/O architecture is the key to success and involves the judicious trade-off between hardware and software for implementing system functions. The I/O interfaces are the only link with the outside world and hence are crucial from both human-engineering and system-capability points of view. I/O-related topics will be considered in depth below.

The COSMAC 1802 I/O structure, which is extremely elegant and powerful, will be used as the example vehicle throughout this paper. It is summarized in Appendix A and fully documented in Ref. 3.

System design steps—top-down approach

Fig. 2 shows the basic steps in the top-down approach to microcomputer system design. Note that the last three steps can be done concurrently. The following paragraphs examine each design step briefly.

*PROM usually refers to memories that can be programmed only once (e.g., fusible-link PROMs).

**Usually the entire chip is first erased via uv light exposure so it can then be written onto. The write time is generally much longer than the read time.

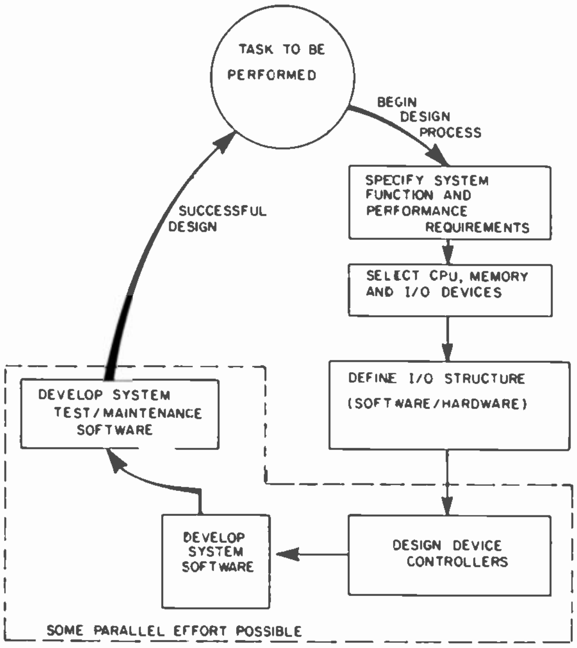

TASK TO BE PERFORMED
BEGIN DESIGN PROCESS
SPECIFY SYSTEM FUNCTION AND PERFORMANCE REQUIREMENTS
SELECT CPU, MEMORY AND I/O DEVICES
DEFINE I/O STRUCTURE (SOFTWARE/HARDWARE)
DESIGN DEVICE CONTROLLERS
DEVELOP SYSTEM SOFTWARE
DEVELOP SYSTEM TEST/MAINTENANCE SOFTWARE
SUCCESSFUL DESIGN
SOME PARALLEL EFFORT POSSIBLE

Fig. 2
Top-down approach to microprocessor system design is becoming more evident.

Specify system functions/performance.

The system designer must identify, at a high level, all the functions the system must perform. These may include monitoring and control, with various user interactions. Critical interrupt and other response times must be clearly identified along with the need for real-time operation. Finally, noise, power consumption, measurement techniques, diagnostics, and maintenance requirements must all be carefully spelled out. If effective simulation tools are available, this is the proper time to use them to their fullest.

Select CPU, memory, and I/O devices.

The system architect must then select the CPU, memory, and the I/O devices. This selection process must be made with full attention to noise, power, auto-restart, and environmental considerations. Typical I/O devices may include disc and tape units for mass storage, display devices, keyboards and switches for operator interaction, sensors and transducers for monitor/control functions, and backup power if uninterruptible operation is necessary.

Define I/O structure.

Having specified the I/O devices, the designer must then define how they will communicate with the computer system, both at the hardware and software levels. Additionally, each set of functions, for every given I/O device, must be partitioned into those suitable for software and hardware implementation.

At this stage of the design process, the system architect must indicate which device will use which CPU I/O facilities (interrupt, DMA channel, external flags, I/O instructions, etc.) and whether they will operate in a one-, two-, or multi-level I/O environment. The concept of multilevel I/O is covered in depth later.

Design device-interface hardware.

This is a relatively straightforward procedure. Care must be used to ensure that sufficient signal drive for device control is provided, and that compatible logic levels are used. Also, the logic family chosen must judiciously balance cost, power, speed, and noise immunity.

Develop system software.

Once the hardware is fully debugged, the software must be developed to control and guide the hardware into performing its appointed task. Decisions must be made whether to program at the assembly level (more programming effort, but better memory utilization and higher throughput) or to use a higher level language (less programming effort, but less efficient use of memory and lower throughput).

A third alternative is to develop a custom interpreter to support a more "English-like" language. For example, a test language can be defined to enable relatively unskilled operators to develop complex test programs. A resident interpreter would then decode these "English-like" commands at run time.

Finally, a decision must be made regarding the software structure. Should it be modular and flexible (subroutine-oriented) for ready expansion, or should the code be packed for maximum memory efficiency?

Perform system testing and provide diagnostics.

This last step is often given only lip service, yet it is crucial to the project's success. Not only must the designer ascertain that the system is properly packaged, that the power supplies are adequate, etc., but he must also specify test procedures so that repairs can be made efficiently. This may include the generation of good documentation and software diagnostics (test and maintenance programs) to help pin-point hardware failures or to ascertain proper system operation.

To summarize, the top-down approach to system design is far preferable with microcomputer systems. There is certainly a trend towards this philosophy.[4]

Choosing the I/O system

There are many ways in which microprocessors can communicate with the outside world. Not all microprocessors are capable of supporting all the following I/O techniques, but COSMAC, fortunately, can. The various ways that a CPU can interact with the I/O devices fall into two general classes—those totally under software control and those occurring asynchronously with normal processing.

Software-controlled I/O

There are three dominant categories of software-controlled I/O—programmed-mode I/O instructions, memory mapping, and serial input and output ports. Each of these techniques is briefly illustrated below. (See the Appendix for signal details.)

Programmed-mode I/O instructions are specific CPU operation codes that transfer information between peripheral devices and memory.

In COSMAC, a 61-67 instruction will move M(R(X))* onto the data bus, whereas 69-6F will move the data bus contents (supplied by the peripheral device addressed) into M(R(X)). The CPU provides enough information to the peripheral device to allow it to ascertain when valid data is on the bus (memory to I/O) or when it must place its infotmation on the bus (I/O to memory).

Referring to Fig. 3, we note that when MRD is a logical "1", the N bits identify a 61-67 instruction. Conversely, when MRD is at logical "0", the three N bits identify a

*R(X) is a 16-bit general purpose register defined by the 4-bit X register. M(R(X)) is the contents of the memory location addressed by the contents of R(X).

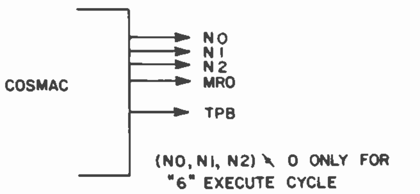


Fig. 3
Programmed-mode I/O uses MRD and three N bits to identify 61-67 instructions (memory to data bus) and 69-6F instructions (data bus to memory).

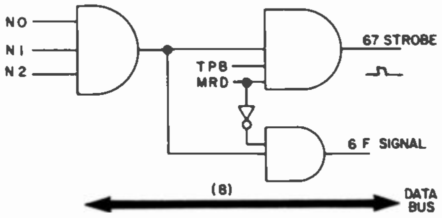


Fig. 4
Data is strobed out at TPB pulse in this example of programmed-mode I/O.

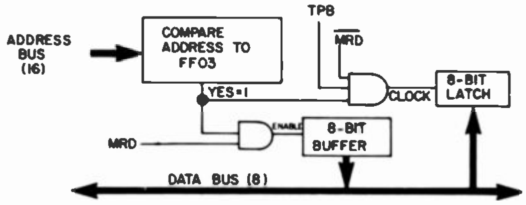


Fig. 5
Memory-mapped I/O assigns a memory address to each 8-bit I/O port (FF03 in this example). The I/O hardware then decodes the address and sends input or output data when required.

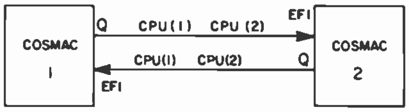


Fig. 6
Serial I/O ports are physical pins on the CPU chip whose level can be sensed by software. In COSMAC, for example, the four EF lines can represent input ports and the Q line can represent the output port.

69-6F instruction. Since the N bits are zero unless an I/O execute cycle is taking place, they can be used to indicate when and which I/O instruction is taking place. Fig. 4 illustrates this situation. Note that the data on the bus is strobed out at TPB (output instruction), whereas input data is provided to the bus during the entire execute cycle.

The memory-mapped I/O concept is straightforward.

One views any 8-bit I/O port as a memory location. Memory addresses assigned to these I/O ports must not overlap the address range of the real memory system. The I/O hardware need only decode the desired address and input or output data as required. Any memory read or write instruction can be used to access these I/O ports. Fig. 5 illustrates the approach where the 16-bit address FF03 is assigned to this I/O port. When a memory read instruction is generated, MRD=1 and the buffer contents are held on the bus during the entire machine cycle. When a memory write is generated, TPB clocks the bus data into a latch.

Many microprocessors, including COSMAC, have serial input and output ports.

Serial input ports refer to physical pins on the CPU chip whose state can be sensed via software. COSMAC's four EF lines (see Appendix A) can be viewed as four serial-input ports. They, of course, may also be used in entirely different ways, as is shown below. When used as serial input ports, information is received by the CPU as a time sequence of logic states. Software routines convert these serial bit streams into bytes compatible with normal CPU operation.

Likewise, serial output ports are physical CPU pins whose level can be set via software. Software routines can convert 8-bit bytes into serial bits streams with the transmission rate and all timing fully under software control. The Q output represents the serial output port in COSMAC.

The use of serial I/O ports usually results in a shift of the load from hardware to software. The hardware is usually very simple (and low cost) but the CPU is almost completely tied up during signal reception/transmission. Fig. 6 presents a trivial two-CPU system illustrating this method.

Asynchronous I/O

The two most common forms of asynchronous I/O are via interrupts and direct memory access (DMA) channels.

An interrupt facility may be viewed as a subroutine call mechanism under the control of external events.

When the interrupt line of a CPU is activated, the CPU branches to a specific location in memory, usually upon the completion of the current instruction execution. This location is the entry point of the interrupt service routine, which uses other forms of I/O to identify both the interrupting device and the specific cause for the interrupt. Additional I/O will then be used to service the interrupt, at the completion of which normal processing resumes where it left off.

A DMA channel provides a quasi-transparent path via the CPU between the memory and I/O subsystems.

Full control of the DMA channel is maintained by the specific I/O device having access to it. When a DMA input or output request is generated by the I/O device, the CPU will complete execution of the current instruction and then "hold its breath" for one machine cycle. During this time a byte can flow between memory and I/O devices and various counters and pointers are automatically updated to prepare the DMA channel for the next service request. Additional general information on DMA and interrupt facilities is available in Ref. 4.

COSMAC's interrupt facility, explained fully in Ref. 3, consists of a single interrupt line. When this line is activated, control automatically transfers to a subroutine pointed at by R(1), the new program counter. Hardware or software techniques can now be used to ascertain the interrupting device and the cause of the interrupt. Where speed is of the essence, hardware techniques can be used to implement both vectored and priority interrupt facilities. A vectored interrupt facility provides the ability to branch to different service routines for different interrupt causes. A priority interrupt facility may be required, in case of simultaneous interrupts, if certain devices must be serviced before others (because they might have higher data rates, for example). Software techniques for the above are usually based on polling the devices via testing appropriate EFs (see Refs. 5 and 6). Ref. 7 presents an implementation of a hardware interrupt priority-resolution circuit.

COSMAC's DMA facility, detailed in Ref. 3, provides the bulk of what is required to implement a full DMA channel. When the DMA-IN or DMA-OUT signals (see Appendix A) are activated by an I/O device, the CPU will complete the execution of the current instruction, then undergo a cycle steal (or DMA cycle), during which a byte will flow between memory and the I/O device via the data bus. The memory location is defined by R(0), which is automatically incremented during each DMA cycle. The I/O device, however, must count the number of bytes transferred and terminate the activity when the desired number of byte transfers has occurred. This counting capability must be implemented in hardware external to the CPU, and for that reason, the COSMAC DMA facility is not a full DMA channel.

The usefulness of COSMAC's DMA capability has been illustrated in two radically different applications—a home/school computer for tv display refresh[8] and a data-communications system for moving blocks of data between memory and a floppy disc subsystem.[5,6]

I/O architecture and multilevel I/O structures

The design of the microcomputer's I/O architecture is one of the most critical steps in the entire system design process, as it will drastically affect both cost and performance. Most I/O subsystems will use a mixture of some or all of the techniques discussed in the previous section. Additionally, multilevel I/O structures often greatly magnify the system's I/O capability at relatively low incremental hardware and software cost.

Every CPU offers a different I/O facility, so the cost and performance trade-offs associated with any I/O technique will be unique to that processor. Hence, we will restrict our discussion to the COSMAC CPU.

With COSMAC, there is usually little need to go with a memory-mapped I/O scheme, since the hardware cost associated with using "6" instructions is minimal. If the system requires more than 7 input or 7 output ports, multilevel I/O structures can yield any desired number of I/O ports. As always, a few exceptions exist. In certain cases, such as where there is a real-time clock and the CPU may want to read the date and time of day stored in several sequential memory locations, a memory-mapped approach may prove effective (see Fig. 7). In this example, the CPU can obtain the date or time simply by reading the contents of memory location FFF0-FFF2 as required—no other I/O is required since the timekeeping is accomplished via external logic. When using "6" instructions for I/O with devices, the EFs may prove very useful in establishing system status. For example, they can let the CPU know that data is indeed ready to be read in. A simple system making full use of COSMAC's software-controlled I/O structure is the now-famous Microtutor.[9]

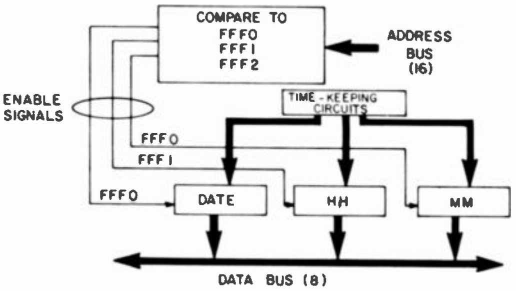


Fig. 7
Memory-mapped I/O is not normally needed with COSMAC, but this clock-reading circuit is one example of its use. Date, hours, and minutes are stored in memory locations FFF0, FFF1, and FFF2, and so are accessible by simply reading those locations.

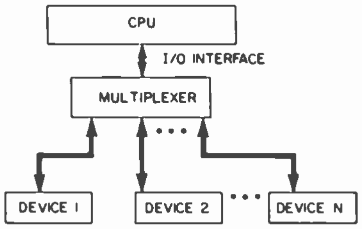


Fig. 8
Two-level I/O uses multiplexing to connect the software-controlled interface to up to 256 devices under CPU control.

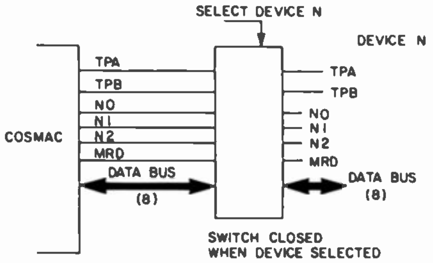


Fig. 9
Multiplexing gates only the software-controlled I/O lines.

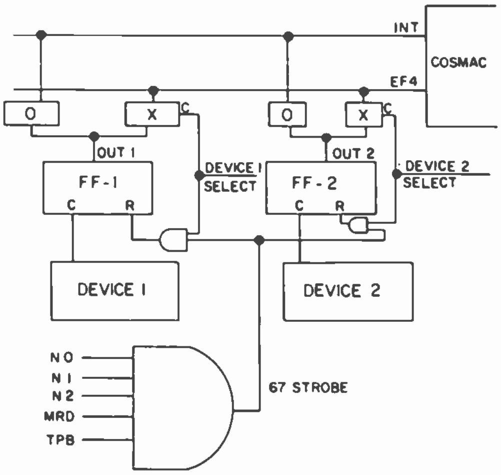


Fig. 10
Two-level I/O system determines the priority of multiple interrupts rapidly. Full explanation is in text.

As system complexity grows, the need for more and more I/O instructions and EFs becomes desirable. This need can be satisfied by the use of two- and multilevel I/O structures.* Two-level I/O will be briefly described, with the note that the same technique can be extended to more complex multilevel structures.

Two-level I/O uses multiplexing to increase the number of software-controlled devices.

The two-level I/O concept is best illustrated via Fig. 8. The multiplexer logic will connect the software-controlled I/O interface to any of 256 devices under CPU control. In this way, the CPU can, at a given point in time, use its full I/O power in communicating with any one I/O device. In effect, after the CPU points at the device it wants to communicate with, only the selected device will be connected to the CPU I/O bus. Usually this multiplexing is achieved by giving up one output instruc-

*A single-level I/O system refers to one where all the I/O instructions, external flags, etc. are dedicated to unique functions for the entire system.

tion (usually 61) to be the device SELECT instruction. When a SELECT instruction is executed, the contents of M(R(X)) enables one of 256 devices and attaches it to the I/O bus. All the other I/O devices will be cut off from the CPU until they, in turn, are selected.

Only the software-controlled I/O lines are so multiplexed—the interrupt and DMA lines are never used in this way, since this would negate their basic operational features. Thus, the signals that are gated via the SELECT switch are EF1-4, Q, TPA, TPB, N0-N2, MRD, and the data bus (see Fig. 9).

Once the two-level concept is implemented, then, except for the SELECT instruction, all the "6" instructions, EF1-4, and Q can have entirely unique meanings for each device. Hence, a 63 could imply "reset logic" for device 7 and "load byte count from data bus" for device 11. At any point in time, the system behaves as if the SELECTed device is the only I/O device in the system. More detailed information on implementing two-level I/O is available in Ref. 3.

When several interrupting devices exist in a two-level I/O system, the full flexibility in permitting a rapid priority determination of the interrupting device becomes apparent. This will be illustrated via a simple example. Consider Fig. 10, where both devices are permitted to interrupt the CPU. Device 1 has priority over device 2 when simultaneous interrupts occur. In both cases, the CPU will use the 67 instruction to acknowledge that interrupt servicing has taken place. The O blocks signify a wired-OR connection.* The X blocks signify transmission gates (when the control signal goes high, a path is established between the two other connections).

Let us investigate the operation of the circuit of Fig. 10. Suppose device 2 interrupts the CPU by setting flip-flop FF-2. Control will transfer to an interrupt routine where the CPU will first select device 1, the highest priority device,which will gate FF-1 onto the EF4 bus. The CPU will then test EF4—it will be inactive. The CPU now knows that device 1 is not the interrupting device. It will then select the next highest priority device—device 2. The selection of device 2 (61 with M(R(X)) = 02) will gate FF-2 onto the EF4 bus. The CPU will now test EF4 and establish that device 2 is indeed the interrupting device. Upon servicing device 2, it will issue a 67 instruction, which will reset FF-2. The CPU will now resume processing where it left off upon interruption.

Suppose both devices had interrupted the CPU simultaneously; then, following the above sequence of events, the CPU would first select device 1 and service it, issue a 67 instruction to reset FF-1, select device 2 and service it, and finally issue a 67 to reset FF-2. The CPU would then resume normal processing.

Choice of language and system hardware

Both the selection of microprocessor language and system hardware are important to the cost-effectiveness of a microcomputer system.

The choice of language is difficult—and is really between assembly-level or higher-level (e.g., Intel's PL/M) code.

The advantages of assembly language over machine code are so many and so obvious that they will not be discussed here. The choice between assembly and higher-level languages is, however, non-trivial and depends strongly on the application. Where a large-volume product is being designed and the cost of each additional byte of memory gets multiplied by the number of systems sold, assembly language will usually be preferable because it gives the programmer machine-level control. The extra engineering required to generate software generation can be written off over many units. On the other hand, for a quick demonstration or a low-volume product, higher-level languages may well be preferable because of their advantages of documentation and rapid code generation.

There are also other considerations. For example, a well-written assembly-level program will usually execute faster than its higher-level counterpart. In general, about all that can be said is that assembly code may be faster, requires less memory, is more expensive to develop, and more difficult to document and update. Ref. 12 provides an excellent analysis of this problem.

The selection of a product hardware package is also not easy.

For large-volume applications, a custom package (and even custom I/O chips) may be optimal. For very low volume, on the other hand, it may be desirable to build the product around the CPU manufacturer's prototyping system, e.g., the COSMAC Development System. Again, the engineering and product costs associated with a new hardware design must be weighed against the ease of implementation but high unit costs of prototyping systems.

Multi-microcomputer structures

The natural evolution of microprocessor-based system architectures brings about distributed processing, i.e., multi-microcomputer systems. In distributed intelligence systems, intelligent subsystems that are dedicated to specific tasks communicate in an optimal fashion to improve system throughput, increase realiability, and add a new dimension of flexibility.

Multiprocessor vs. multicomputer

Considerable confusion exists in the literature, since the terms "multiprocessor system" and "multicomputer system" are often used synonymously.

In general, multiprocessor systems are those that operate on a single input stream or workload. A single integrated operating system allocates hardware resources where needed. These systems are used primarily in situations where high reliability is a must. This is achieved by either assembling a fully redundant system or by allowing for rapid system reconfiguration.[10] Multicomputer systems, on the other hand, operate on several input streams, and do not have an integrated operating system.

Viewed in another light, the main function of multicomputer systems is to separate or partition the various tasks to achieve improved system throughput. Examples in larger sytsems include the separating of "number crunching," done in the main CPU, from "I/O processing" performed in various I/O processors. In multi-microcomputer systems, the primary motivation is to separate tasks that are mostly independent, i.e., ones that require

*In a wired-OR connection, when any input signal goes active, the bus signal goes active.

relatively little intercommunication. This partitioning enables the various microcomputers to be far more responsive to their dedicated tasks. In fact, each microcomputer may be controlling a process requiring, for example, rapid responses to interrupts. It would be difficult, perhaps impossible, for a single microprocessor to respond rapidly to a large number of interrupts.

Multi-microcomputer structures

In multi-microcomputer systems, each microcomputer, ideally, performs a dedicated task. Very little data is interchanged between processors relative to the total system data flow; each microcomputer can operate relatively independently of the others. The software in each subsystem is optimally tailored to the processing task for which it has responsibility. Should any one CPU become overloaded, it cannot be unburdened by any other processor; however, this should not occur with proper system design.

Multi-microcomputer systems, often called Distributed Intelligence Microcomputer Systems (DIMS), offer many advantages to the performance of the total system. The system can be modular in nature, where each subsystem is similar in hardware but customized in software. This greatly reduces maintenance and spare-parts inventory problems. Should one subsystem fail, the remainder can usually keep functioning, giving the system some "fail soft" capability, and hence, improved reliability. In DIMS, each subsystem is generally I/O-oriented, and the total system throughput, with careful design, can approach the sum of the throughputs of the individual processing elements.

Master-slave or master-master?

A myriad of possible DIMS structures exist. At one extreme lies the "master-slave" or "star" organization; at the other lies the generalized-network, or master-master, structure where every CPU has equal status. Intermediary structures and loop structures are also possible and may, in fact, be optimal for some particular applications.

Before discussing the applicability of any given structure in a microprocessor environment, it is worthwhile to examine the extreme structures. Fig. 11 illustrates a general network structure, in which any CPU can communicate with any other CPU. In this organization, all the CPUs must support compatible interprocessor interfaces and I/O instructions.

The "master-slave" organization, Fig. 12, offers many advantages to DIMS systems. In this organization, all inter-CPU communication must always go through the master. The principal advantage of this structure is that each slave CPU can have unique interprocessor I/O instructions, optimally tailored to its task. Hence, interface costs are reduced and extremely efficient use of I/O codes results.

Yet another possible structure is the ring structure illustrated in Fig. 13. Note, however, that if the information bus is also needed by the individual CPUs for their own processing, severe contention problems will ensue, with a resulting degradation in the performance of the overall system.

The master-slave organization appears to be the most suitable for the bulk of multi-microcomputer applications envisioned, i.e., ones where a main CPU controls and supervises a multitude of intelligent (microprocessor-based) subsystems, each dedicated to a specific task.

Distributed intelligence always raises the question of shared main memory.

It is clear that, as the number of processing elements increases beyond two or three, severe contention problems will arise if each CPU must access common memory for a substantial fraction of its cycles. Thus, if a common memory is used, references to it by the individual CPUs must be minimized. Hence, some local RAM is highly desirable. In fact, insofar as multimicrocomputer systems are concerned, the primary use of a common memory would be to act as a message center, where each CPU can leave messages for other CPUs and pick up messages intended for it. Such an organization is illustrated in Fig. 14. Note that if there are N processing systems, N mailboxes are required, each one having $N-1$ compartments. Basically, if the i^{th} CPU wishes to leave a message to the j^{th} CPU, it simply stores the desired information in the approprapriate compartment of the j^{th} CPU's mailbox. Any CPU can scan its mailbox to see if there are any messages for it.

The usefulness of shared memory in a master-slave organization is far more limited, since the number of paths is now equal to the number of slave processors, and hence is relatively small. A preferable approach in this case is to allow block transfers of data between the master and slave microcomputers via direct memory access (DMA) channels. The CPUs may communicate at lower data rates under program control, but data blocks will be

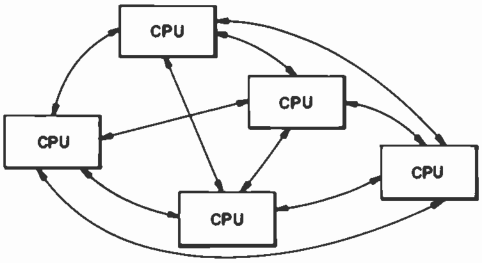


Fig. 11
Generalized network structure allows any CPU to communicate with any other CPU. Each CPU must have compatible interprocessor interfaces and I/O instructions.

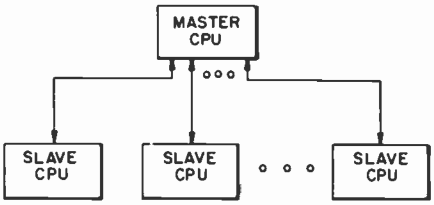


Fig. 12
Master-slave organization requires all inter-CPU communication to go through the master. In contrast to the generalized network of Fig. 11, this system allows specialized I/O for each CPU.

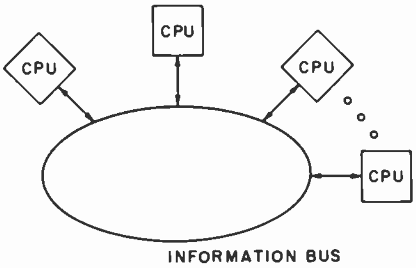


Fig. 13
Ring structure uses a single information bus to interconnect multiple CPUs.

transferred asynchronously with normal processing, thereby improving system throughput. This is the approach taken in the COSMAC interprocessor interface outlined below.

An interprocessor interface for COSMAC

The philosophy behind the current hardware design is as follows: a master-slave organization will be used with one master and an arbitrary (up to 256) number of slave processors. No common memory will be provided and each CPU will have sufficient RAM to handle its dedicated workload. The master CPU will have sufficient RAM to buffer all information that will be exchanged between it and all the slave processors. Interprocessor communication will be either via programmed-mode I/O (type "6" instructions) or via DMA for block transfers. The external flags will be used for handshaking and status information. Finally, the master processor can interrupt any slave, but no slave can interrupt the master.

The system organization is such that each slave CPU will have a dedicated interprocessor interface associated with it. These interfaces will be identical in hardware, with the exception of their device numbers. The master CPU is expected to operate in a two-level I/O environment with each slave CPU viewed as a peripheral with a unique device number. Slave CPUs may or may not use two-level I/O, but they, too, will view the master as a peripheral device.

Fig. 15 illustrates a three-CPU system consisting of a master and two slave CPUs, and hence two interprocessor interfaces.

Figure 16 presents a high-level view of the interface organization. Basically, two 8-bit buffers provide asynchronous communication between the master-CPU and slave-CPU data buses. Note that the two CPUs need not be synchronized, nor need they operate at identical clock rates. Besides the two 8-bit buffers, latches are provided for the external flags (EF1-4) of both CPU's, the master CPU's DMA requests, and the slave CPU's interrupt line. The control logic, which provides timing and clocking signals where needed, is not shown.

The basic hardware organization allows the exchange of information between the master and slave CPUs under programmed-mode I/O via the two buffers and external flag decodes. The master is allowed to interrupt the slave, but not *vice versa*. High-speed block transfers between RAM(1) and RAM(2), or RAM(1) and the RAM of any slave CPU, run via the master's DMA channel, which must be multiplexed among all the slave CPUs. This approach frees the slave-CPU DMA channels for use in conjunction with the subsystems to which they are dedicated (e.g., disc control). Each CPU, in effect, views the CPU it is communicating with as a peripheral device with which it can interact in an asynchronous fashion.

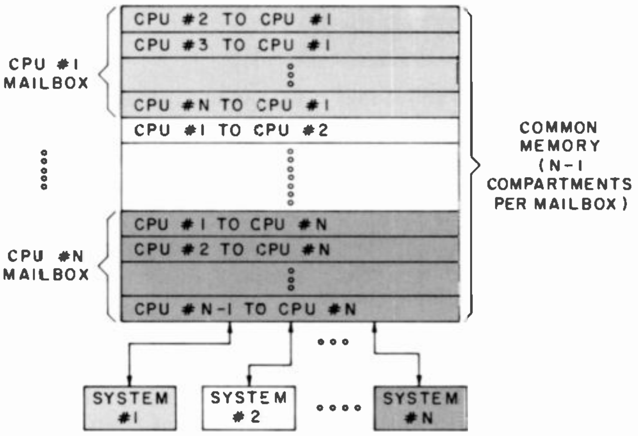


Fig. 14
Shared memory acts as a message center in a multiple-CPU organization. CPUs scan their "mailboxes" for messages from each other.

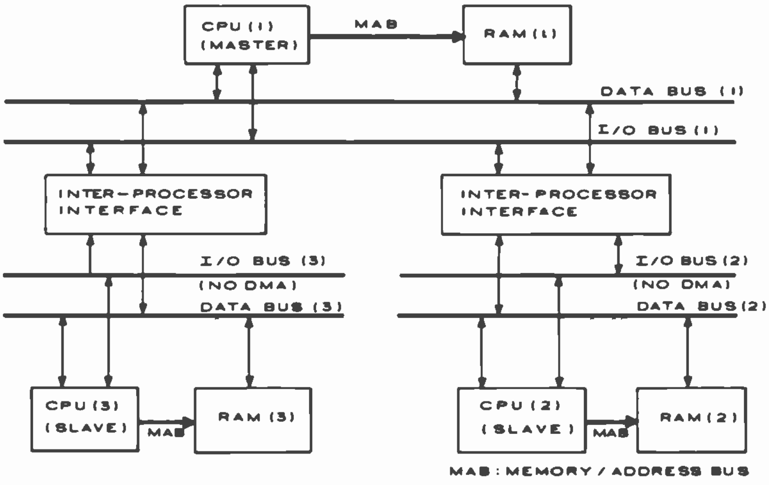


Fig. 15
This three-CPU system consists of master and two slaves; master views each slave as a peripheral with a unique device number.

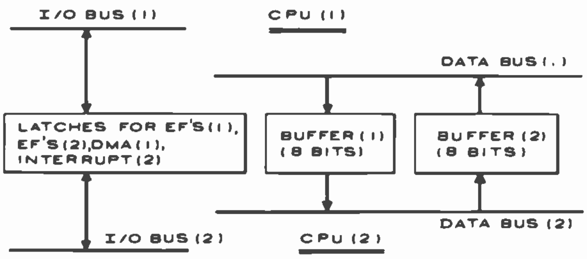


Fig. 16
High-level view of interface organization. Communication is asynchronous.

However, the master and slave CPU protocols are different.

The CPUs communicate via "6" instructions and EF assignments. These I/O instructions are used both for programmed-mode data transfer and to initiate DMA(1) block transfers between RAM(1) and any slave CPU's RAM. Ref. 11 details the above interface.

The advantages of the COSMAC interprocessor interface's organization are numerous. Foremost is the complete lack of contention, since each CPU can make full use of its cycles. Each slave CPU is free to use its DMA channel to communicate with the outside world. The master's DMA channel is time-multiplexed among the various slave CPUs under its control. The master can interrupt any slave, but not *vice versa*. The architecture is open-ended, since most of the bit assignments are user-specified via software. Hence, each interface can be customized in software while the basic hardware remains unchanged. Finally, the communication protocol is such that every processor can operate asynchronously, with a different clock rate. Thus, overall system synchronization is not required.

Concluding remarks

It should be apparent from the above that the design of microprocessor-based systems is part intuition, part art, and part science, but mostly common sense. Once an application is clearly defined, the choice of CPU, memory, and I/O devices follows readily. Designing the I/O architecture is more complex, but is fundamental to the system satisfying its cost/performance objectives. The right mix of I/O techniques usually depends strongly on the I/O facility of the selected CPU. Multilevel I/O structures, DMA, and interrupt facilities should be used effectively to satisfy desired cost and performance levels. Multi-microcomputer system architectures may be optimal for certain applications. They should be approached with caution, however, because of the increased difficulty in software design due to the requirements for synchronization between the various software subsystems associated with the various CPUs.

References

1. Kaye, D.N.: "How to pick a microprocessor, a mini or anything in between," *Electronic Design*, Aug 2, 1975, pp. 26-30.
2. Weissberger, A.J.: "MOS/LSI microprocessor selection," *Electronic Design*, Jun 7, 1974, pp. 100-104.
3. "User Manual for the CDP1802 COSMAC Microprocessor," MPM-201A, RCA Solid State Division, Somerville, N.J.
4. Raphael, H.: "Evaluating a microcomputer's input/output performance,," *Electronics*, Aug 17, 1976, pp. 105-109.
5. Russo, P.M.; and Lippman, M.D.: "Case history: store and forward," *IEEE Spectrum*, Sep 1974, pp. 60-67.
6. Lippman, M.D. and Russo, P.M.: "A microprocessor controlled for international leased data channels," *Proc. of the ICC*, Minneapolis, (Jun 1974).
7. Marcantonio, A.R.: "Interrupt-priority resolution circuit for use with RCA COSMAC development systems," ICAN-6485, RCA Solid State Division, Somerville, N.J.
8. Weisbecker, J.A.: "A practical, low-cost, home/school microprocessor system," *IEEE Computer*, Aug 1974.
9. "COSMAC Microtutor Manual," MPM-109, RCA Solid State Division, Somerville, N.J.
10. Enslow, P.H., Jr., (editor): *Multiprocessors and Parallel Processing*, (John Wiley and Sons, Inc.: New York: 1974).
11. Russo, P.M.: "An interface for multi-microcomputer systems," *Proc. COMPCON 76*, Washington, D.C., (Sep 1976).
12. Gibbons, J.: "When to use higher-level languages in microcomputer-based systems," *Electronics*, Aug 7, 1975, pp. 107-111.

Appendix: The COSMAC I/O interface

The CDP1802 I/O interface is illustrated in Fig. A-1. It consists of an 8-bit data bus and 16 control lines. These 16 control lines break down into 4 lines used for synchronization and CPU state identification (timing pulses, TPA and TPB, state codes SC0, SC1), 4 lines used to identify type "6" I/O instructions, (3 "N" lines combined with memory read to decode 61-67 output, 69-6F input), 4 software-testable external flags (EF1-4), one software-controlled output level, one interrupt line, and two lines for controlling COSMAC's built-in direct memory access (DMA) channel.

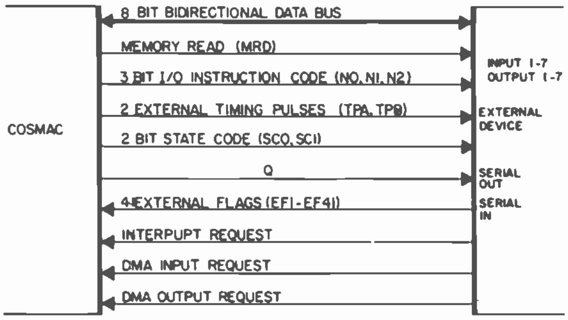
From the above, we can identify four distinct ways in which COSMAC can communicate with the outside world via its I/O interface. Three of these provide both input and output capability—they are programmed-mode I/O (type "6" instructions), direct software testing or setting of levels (EF1-4, Q), and the DMA channel. The fourth way is via the CPU's interrupt capability, which can only alert the CPU that some external event has occurred. Of course, COSMAC can also communicate with the outside world via its memory interface, as discussed in the text.

When multilevel I/O structures (see text) are employed, only the contol lines associated with programmed-mode I/O, EF1-4, and Q should be gated with the device SELECT line. The interrupt and DMA channel control signals should never be gated with SELECT since they control activities occurring asynchronously with normal CPU processing.

The COSMAC microprocessor supports four basic CPU states, identified by SC0 and SC1. Each CPU state is active for one machine cycle (8 clock periods). The CPU states are INSTRUCTION FETCH (S0), INSTRUCTION EXECUTE (S1), DMA CYCLE (S2), and INTERRUPT CYCLE (S3).

The I/O devices need to identify the S1 cycle for programmed-mode I/O, the S2 cycle for data transfer via DMA, and the S3 cycle to acknowledge that the interrupt has taken.

The two timing pulses available are TPA and TPB. TPA occurs early in the machine cycle and is used primarily to latch the upper memory address byte. TPB occurs late in the machine cycle and is used primarily to clock valid data from the bus. Of course, both TPA and TPB can be used to generate strobes to reset or activate devices as needed.



COSMAC
8 BIT BIDIRECTIONAL DATA BUS
MEMORY READ (MRD)
3 BIT I/O INSTRUCTION CODE (NO, NI, N2)
2 EXTERNAL TIMING PULSES (TPA, TPB)
2 BIT STATE CODE (SCO, SCI)
Q
4 EXTERNAL FLAGS (EFI-EF4I)
INTERPUPT REQUEST
DMA INPUT REQUEST
DMA OUTPUT REQUEST
INPUT 1-7
OUTPUT 1-7
EXTERNAL DEVICE
SERIAL OUT
SERIAL IN

Fig. A-1
CDP1802 I/O interface consists of an 8-bit data bus and 16 control lines.

COSMAC resident software development aids

These self-sufficient aids make COSMAC microprocessor software development possible without the need for other computing facilities.

L.A. Solomon

The COSMAC Development System, CDP18S004, is a prototyping system for developing products based on the RCA 1800 microprocessor series. Referred to as the CDS, the system uses a CDP1802 microprocessor as a CPU and includes a complete family of software aids suited to a spectrum of cost/performance-conscious users. These self-sufficient software aids make complete software development possible without recourse to any other computing facilities.

The basic CDS is supplied with paper-tape and cassette versions of resident-editor and resident-assembler programs that allow the users to do program development and debugging on the COSMAC Development System itself. This self-contained system is an attractive alternative to the timesharing approach to software development.

For greater convenience in software design, the editor and assembler programs are also available on a floppy disk system CDP18S800. This system includes two disk drives, each capable of addressing up to 250 kilobytes of stored files on removable diskettes. The floppy-disk system has special CDS-compatible versions of the software aid programs and an interface card that plugs directly into the CDS.

Resident software

A resident software aid is a program that runs on a system (such as the CDS), is stored in or loaded into one of the system's memories, and performs some general function for the user. Such an aid may be classified in two ways. A permanently resident program is stored in ROM, occupying a fixed portion of the addressable memory space, and a temporarily resident program is loaded into some portion of the existing RAM space.

The permanently resident functions in CDS include utility and debug programs. The utility program loads memory with temporarily resident programs and transfers control to them, so they can run; the debug program is for interactive diagnosis. The temporarily resident programs include the editor and assembler.

Assembler program

An assembler is a program that aids a user in writing his own application programs. Its function is to translate a program in assembly language into hexadecimal code, ready for machine operation. Although the user could write out and enter his program in machine language, an assembler offers several very important advantages, including:

- Mnemonic abbreviations rather than numerics for instruction names. This leads to fewer mistakes and makes "reading" the program easier.
- User-assigned mnemonics for the microprocessor's registers and locations in memory. This makes it possible to conveniently insert instructions or move parts of a program around; the assembler makes the necessary address changes automatically.
- Comments can be added to the program at will. This makes reading, debugging, or modifying the program easier, particularly if some time has passed since it was written.

To gain these advantages, the program is written in COSMAC assembly language, which is principally a symbolic representation of COSMAC instructions. The program may be typed in every time it is needed (not practical if it is long!), punched on paper tape, recorded on magnetic cassettes or on floppy disks, or stored in timesharing computer files. These approaches are successively more expensive, but also are increasingly easier and faster to work with. Whatever the program storage medium, the assembler translates the assembly-language program into hexadecimal code ready to run on a COSMAC system.

COSMAC resident assembler

The COSMAC resident assembler (CRA) program assembles COSMAC programs without the use of another computer. CRA runs directly on the COSMAC Development System itself in a stand-alone

Reprint RE-22-5-15
Final manuscript received November 23, 1976.

Larry Solomon was the principal designer of the COSMAC resident assembler and its subsequent enhancements. He is presently a Leader in Software Development and Systems. He had five years of minicomputer system design for real-time data-collection and analysis systems before coming to RCA in 1974.

Contact him at:
Solid State Division
Somerville, N.J.
Ext. 6349

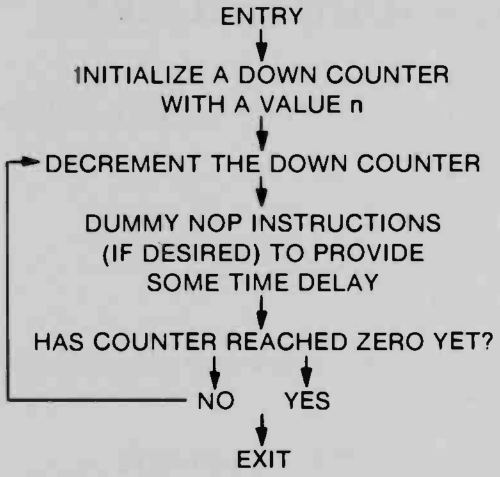

ENTRY
INITIALIZE A DOWN COUNTER WITH A VALUE n
DECREMENT THE DOWN COUNTER
DUMMY NOP INSTRUCTIONS (IF DESIRED) TO PROVIDE SOME TIME DELAY
HAS COUNTER REACHED ZERO YET?
NO
YES
EXIT

Fig. 1
"Time-out" program in flow-chart form.

F8FFB1219191913A0300

Fig. 2
Machine-language version of the program of Fig. 1 is a number of steps away from the flowchart. Figs. 3 through 9 show how to get there.

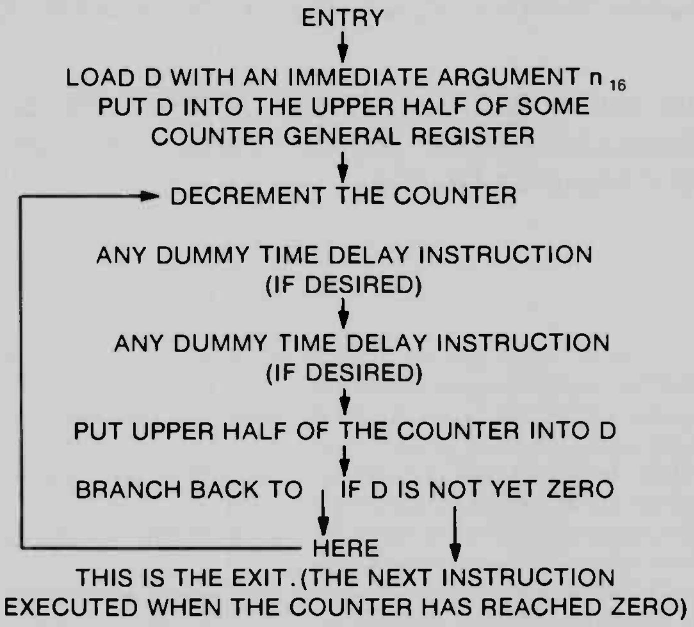

ENTRY
LOAD D WITH AN IMMEDIATE ARGUMENT n 16
PUT D INTO THE UPPER HALF OF SOME COUNTER GENERAL REGISTER
DECREMENT THE COUNTER
ANY DUMMY TIME DELAY INSTRUCTION (IF DESIRED)
ANY DUMMY TIME DELAY INSTRUCTION (IF DESIRED)
PUT UPPER HALF OF THE COUNTER INTO D
BRANCH BACK TO
IF D IS NOT YET ZERO
HERE
THIS IS THE EXIT. (THE NEXT INSTRUCTION EXECUTED WHEN THE COUNTER HAS REACHED ZERO)

Fig. 3
Program in flowchart form, but using specific COSMAC instructions.

manner; it converts source programs written in COSMAC Level 1 assembly language into executable (hexadecimal) machine code.

An assembler permits the programmer to write programs using a convenient set of symbols and expressions. An input or source program consists of a sequence of statements; the assembler normally translates these statements into an equivalent sequence of hexadecimal digits (a single machine instruction or a data field of user-defined constants). This code is then placed in its proper position (i.e., assembled) in an output or object file—the executable machine program. Some statements are special control commands to the assembler. Called assembler directives, they do not directly cause output code to be produced.

CRA is a two-pass assembler. That is, it normally reads the complete source file twice to complete an assembly. During the first pass, it constructs the symbol table in RAM and prints it on the terminal, and also flags syntactic errors. On the second pass, it generates object code using the symbol-table values just derived and then prints an assembly listing. Additional program errors (for example, the detection of undefined symbols) may be flagged on the second pass.

By way of example, let us examine a simple "time-out" program that takes a number and decrements it to zero. The time from ENTRY to EXIT is approximately *n* times the time for one pass through the loop. Fig. 1 is a flowchart for such a program; Fig. 2 is an acceptable machine-language statement for it. The flowchart, of course, is much more understandable than the machine-language version of the program.

Understanding the resident assembler—from flowchart to operation mnemonics.

The time-out test program given above can illustrate some of the essential properties of the COSMAC resident assembler by starting from the flowchart and proceding toward the hex form "by hand." Fig. 3 is a version of the program expressed in terms of specific COSMAC instructions.

Using shorthand mnemonics for the instructions and comments gives us Fig. 4, where:

LDI = load immediate
PHI = put high
DEC = decrement
GHI = get high
BNZ = branch if not zero

Their equivalent hexadecimal codes (for example, 3A for BNZ) can be found in the *COSMAC User Manual* MPM-201. Each line of the program is now beginning to resemble an assembly-language statement.

Mnemonics and comments simplify work for the programmer.

The last version illustrates two fundamental properties of an assembly language—the use of operation mnemonics and the use of comments. An assembler is designed to recognize operation mnemonics, which are much more descriptive to the programmer, and to convert them into their hexadecimal code equivalents. In addition, an assembler is designed to ignore comment text fields in statements when it recognizes their existence. In the program version above, every comment begins with a double period (..) and extends to the end of the line. Comments are valuable to the programmer because they permit him to add documentation to a program's statements.

The assembler also assigns address locations.

The next problem considered is assigning addresses—specifically, the branch address in the last instruction in the loop. Clearly, the address assigned depends on where the program will reside in memory while it is executing. If this location is not presently known absolutely (for example, because the routine's exact location within a larger program may change), a labeling procedure may replace the arrowed path shown. Two examples of labeling are given below in Figs. 5 and 6.

```
    LDI n              .. n IS APPROXIMATELY THE
                       .. NUMBER
    PHI COUNTER        .. OF 256 LOOP PASSES BELOW
┌──►DEC COUNTER        .. REDUCE NUMBER OF PASSES
│                      .. REMAINING
│   DUMMY              .. JUST TO WASTE TIME
│   DUMMY              .. WASTE MORE TIME
│   GHI COUNTER        .. SEE IF COUNT HAS YET
│                      .. REACHED
│   BNZ┐               .. ZERO.LOOP IF NOT
│     HERE
└── EXIT INSTR         ..TIME EXPIRED.GO ON
```

Fig. 4
Mnemonics are a bridge between the programmer's flowchart anc machine language.

```
LABEL:      DEC COUNTER
                 .
                 .
                 .
             BNZ LABEL
```

Fig. 5
Labeling replaces the arrows in Fig.4.

```
BNZ *-m
```

Fig. 6
Addressing method counts a number of bytes past the location of this reference.

```
BEGIN: LDI n       .. n IS APPROXIMATELY THE
                   .. NUMBER
       PHI COUNT   .. OF 256 LOOP PASSES BELOW
 LOOP: DEC COUNT   .. REDUCE NUMBER OF
                   .. PASSES REMAINING
       DUMMY       .. JUST TO WASTE TIME
       DUMMY       .. WASTE MORE TIME
       GHI COUNT   .. SEE IF COUNT HAS REACHED
       BNZ LOOP    .. ZERO. LOOP IF NOT
  EXIT: IDL        .. STOP AFTER TIME DELAY
                   .. HAS EXPIRED
```

Fig. 7
Almost there: assembly-language program does not have all its statement labels referenced.

An assembler permits locations within a program to be identified by English-like symbols (e.g., "LABEL" above). Then, any reference to a location may be made by use of its label (e.g., "BNZ LABEL"). The programmer is free to select almost any sequence of characters for each label. Typically, he chooses a symbol that has some logical meaning within the context of his program (e.g., LOOP, DELAY, TEST1, SEARCH, etc.). During the process of translating the program's statements, the assembler keeps track of the addresses of all the bytes it generates. It starts from some known address reference, such as zero, using an internal location counter for this purpose. Whenever it encounters a LABELed statement, it enters the address of the instruction in a symbol table. All references to that label may then be replaced with appropriate address bytes.

An assembler also normally permits addressing relative to the position at which the reference is found. Often, the special symbol "*" refers to the address of this statement and *m* is a count of the number of bytes from this point.

Two forms of symbolic addressing have been defined. Fig. 7 shows how the "timeout" program now looks using statement labels; it is almost a correct assembly language program. (Three statement labels have been specified, but only one is presently referenced.)

The assembly-language equivalent of our program.

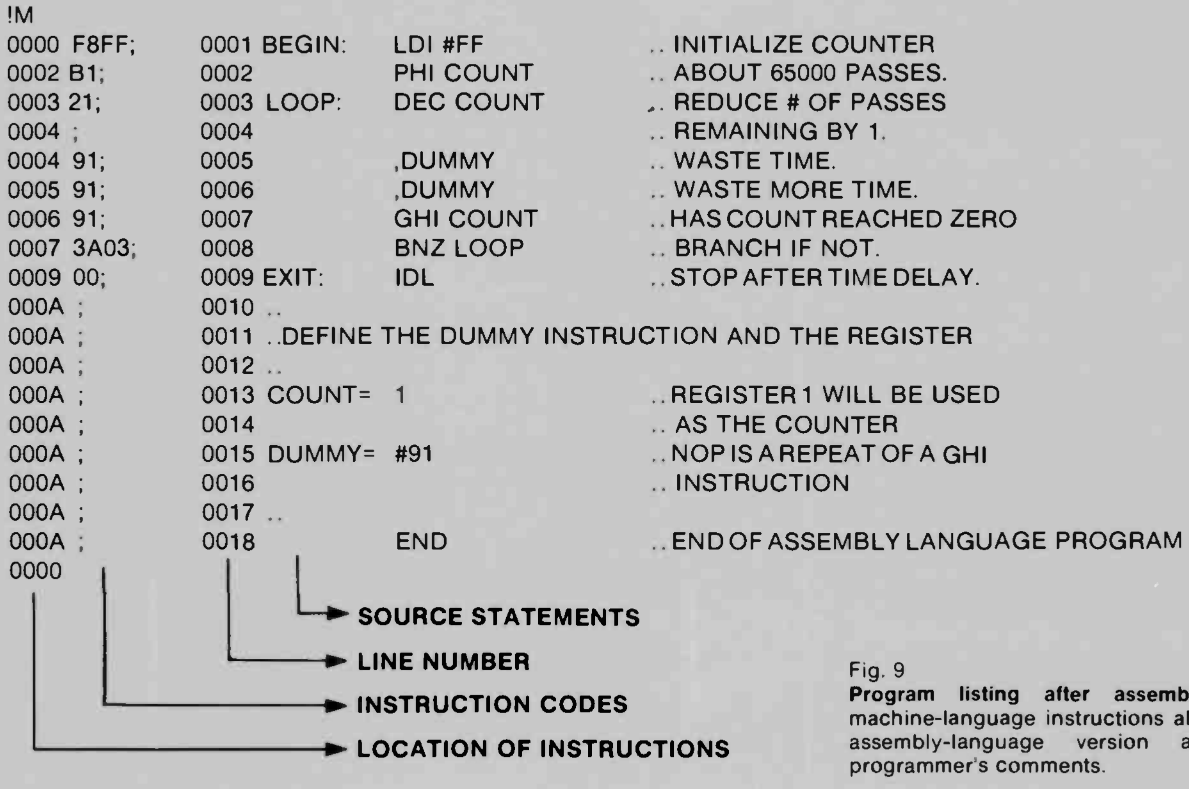
Next comes the selection of the value for *n*, the selection of the COUNT register, and the "DUMMY" instruction. To get the maximum delay, the original version of this program used a hex FF for the immediate byte. The assembly-language statement LDI *#FF* will translate properly. This selection shows that there are still many places in a program where explicit values are specified by the programmer. The "#" indicates the presence of an explicit hex constant. One can similarly explicitly identify the general register to be used as the counter with statements such as PHI 1, DEC 1, etc., assuming R1 was chosen. Suppose, however, that one wished to defer or later modify register assignments. A convenient permissible procedure is to continue to use the symbol as an identifier (in this case, not of a memory location, but of a general register) and give the symbol a value with a special EQUATE statement of the form COUNT=1. In this case, all occurrences of COUNT will be replaced with 1 by the assembler. If, later, one wished to reassign registers, a change to COUNT=10, for example, would automatically change all references to COUNT to the value #0A(hex).

To generate a delay, one may use the NOP instruction or any other time-wasting instruction. The hex program originally given merely repeated the GHI 1 instruction three times. There are several ways by which this instruction can be expressed to

```
BEGIN:        LDI #FF          .. INITIALIZE COUNTER
              PHI COUNT        .. ABOUT 65000 PASSES.
LOOP:         DEC COUNT        .. REDUCE # OF PASSES
                               .. REMAINING BY 1.
              ,DUMMY           .. WASTE TIME.
              ,DUMMY           .. WASTE MORE TIME.
              GHI COUNT        .. HAS COUNT REACHED ZERO
              BNZ LOOP         .. BRANCH IF NOT
EXIT:         IDL              .. STOP AFTER TIME DELAY.
..
.. DEFINE THE DUMMY INSTRUCTION AND THE REGISTER
..
COUNT= 1                       .. REGISTER 1 WILL BE USED
                               .. AS THE COUNTER
DUMMY= #91                     .. NOP IS A REPEAT OF A GHI
                               .. INSTRUCTION

              END              ..END OF ASSEMBLY PROGRAM
```

Fig. 8
Complete assembly-language program.



```
!M
0000 F8FF;   0001 BEGIN:   LDI #FF        .. INITIALIZE COUNTER
0002 B1;     0002          PHI COUNT      .. ABOUT 65000 PASSES.
0003 21;     0003 LOOP:    DEC COUNT      .. REDUCE # OF PASSES
0004 ;       0004                         .. REMAINING BY 1.
0004 91;     0005          ,DUMMY         .. WASTE TIME.
0005 91;     0006          ,DUMMY         .. WASTE MORE TIME.
0006 91;     0007          GHI COUNT      ..HAS COUNT REACHED ZERO
0007 3A03;   0008          BNZ LOOP       .. BRANCH IF NOT.
0009 00;     0009 EXIT:    IDL            ..STOP AFTER TIME DELAY.
000A ;       0010 ..
000A ;       0011 ..DEFINE THE DUMMY INSTRUCTION AND THE REGISTER
000A ;       0012 ..
000A ;       0013 COUNT=   1              ..REGISTER 1 WILL BE USED
000A ;       0014                         .. AS THE COUNTER
000A ;       0015 DUMMY=   #91            ..NOP IS A REPEAT OF A GHI
000A ;       0016                         .. INSTRUCTION
000A ;       0017 ..
000A ;       0018          END            ..END OF ASSEMBLY LANGUAGE PROGRAM
0000
```

Fig. 9
Program listing after assembly gives machine-language instructions along with assembly-language version and the programmer's comments.

the assembler; one method uses another form of EQUATE statement to give a value to a symbol. As will be explained later, a comma may be used to precede many kinds of "constants," some whose values are explicitly stated and some whose values are derived by the assembler. The statement "DUMMY," for example, will cause a substitution of the value for the symbol. Thus, if another statement DUMMY=#91 is supplied, all occurrences of DUMMY will be replaced by a hex 91 (which is a GHI 1 instruction). Finally, the assembler begins assigning address values starting with zero. Fig. 8 shows the final form of one assembly-language equivalent of the hex program we started with.

When this program is assembled, the listing of Fig. 9 is generated. Note the correspondence of the instruction codes in the second column to those listed in Fig. 2.

Editor program

After the user has hand-written his first COSMAC assembly-language program and wants to assemble it, he immediately faces the problem of converting the hand-written source file into a machine-readable form. This conversion involves a keyboard-to-tape operation in which lines on the coding sheet are transcribed to become lines on a source media, a floppy disk for example. It is more likely, however, that the COSMAC resident editor will be used at this point to create the source file. Using the editor assures that the created files are in proper format for later reading by the assembler and for later modification, if necessary, by the editor.

Once a source file has been created and a first assembly-run made, it is very likely that the CRA will return error diagnostics, asking for corrections to the source file to have the program conform to CRA's rules. Typically, the changes required at this point are "trivial" but necessary. For example, spaces may have to be removed in one or more expressions, the same symbol may have been erroneously used for two purposes, an operation mnemonic may have been misspelled, or a punctuation character may have been omitted. The number of possible trivial errors is clearly large.

The editor is used to correct the errors and alter the source file to conform the program to the CRA's rules. Typically, modifications at this point merely involve inserting and deleting single characters or replacing a small string of characters by a substitute string. The erroneous source file is used as an input to the editor and the user generates a corrected source file as an output. The new file is then re-assembled. At this point, other trivial errors that were not apparent to CRA on the first run may appear. For example, an erroneous instruction operand may not have been flagged on the first assembly because its associated statement label or operation mnemonic may have also been in error. Thus, a new edit-reassemble pass may be necessary. Finally, a program is developed to which CRA does not object. At this point, a first run can take place.

The probability of a logical error in the program depends on the length .of the program and the previous experience of the programmer. Assuming one or more logical errors are found (via some "debugging" procedure), the source file must again be modified. Often such modifications are no longer trivial. For example, it may be necessary to find all instructions that branch to a given location and precede some of them with one or more instructions currently not in the program. Often, it may be necessary to delete or insert some code or move some code to a different point in the program. Several duplicated sets of in-line instructions may have to be removed and replaced with calls to one common subroutine that must be added. The user may decide to "clean up" the program logically, in any one of several ways, or to improve its "readability" by modifying its comments or statement formats (by inserting TABs or SPACEs, for example).

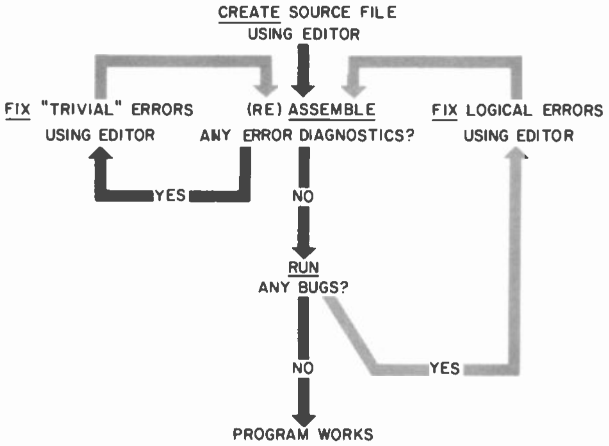


Fig. 10
Steps along the way to a bug-free program.

Such modifications to the source file also involve the editor. After they are completed, a reassembly may again turn up new errors of the "trivial" variety, and so on. Thus, the generation of a bug-free program typically involves the process shown in Fig. 10. It is quite likely that the amount of time spent "conversing" with the editor will be much larger than that spent with the assembler.

A source program may be viewed as a long sequence of characters. When the COSMAC resident editor reads the source file, it places this character-sequence in memory, with the code in each memory byte representing one source-program character. The user is then free to type commands to the editor to manipulate the memory representation of the program. For example, the user may identify a specific location and define a character sequence to be inserted there. He may also identify certain characters to be deleted or altered. He may ask the editor to search for the occurrence of specific character sequences, after which further memory modifications (corrections) may be made.

After he is satisfied that the new memory representation of the file contains all of the changes he desires (frequently an editing session begins with a handwritten list of changes) the user asks the editor to write (create) a new output tape containing the new version of the program. This new file is then used as the input file for a reassembly.

A syntactically "correct" program that is acceptable to an assembler, may still contain logical errors that the assembler is not designed to detect. These errors can only be isolated by running the program to determine where it is malfunctioning. One common debugging procedure is to stop program execution at appropriate logical points and examine (or "dump") the contents of selected registers. It is also useful to reinitialize ("patch") the contents of selected registers before resuming execution at a desired point. Alternatively, the contents of selected registers may be printed ("traced") while the program is executing.

A programmer may explicitly include code to facilitate debugging directly in his program. For example, he may insert stop commands and/or calls to a print routine at various key points. Later, after all the errors have been found, these instructions may be removed or replaced by a "no-op" instruction. Such a process, however, assumes that the programmer can identify those program sections that are error-prone. Furthermore, the procedure is inconvenient and inefficient.

A preferred method is to use a debugging aid, which is generally defined as a facility that permits a programmer to isolate program errors without resorting to special program-preparation steps. This way the programmer is not required to predict beforehand where potential errors may lie.

Program debugging

The CSDP system, which runs on a remote timesharing computer, includes a COSMAC simulator that mimics the real hardware exactly and also includes a rich set of debugging facilities. (See the *Timesharing Manual for the RCA CDP1802 COSMAC Microprocessor*, MPM-202.) These facilities permit a user to debug a program prior to its running on the actual hardware.

The COSMAC Development System's debugging facility (abbreviated ODB, for on-line debugging) is designed to provide many of the debugging features offered by the COSMAC Software Development Package (CSDP). CSDP is a versatile interactive aid for developing and checking COSMAC programs; it uses CSDP-augmented timesharing or in-house computer facilities. ODB, however, operates directly on the CDS in a stand-alone manner. See the *Operator Manual for the RCA COSMAC Development Systêm*, MPM-208, for a description of ODB.

Floppy disk system

As mentioned earlier, the COSMAC Floppy Disk System (FDS), CPD18S800, is a mass-storage unit designed to work with the CDS to facilitate rapid program development. It includes interfacing hardware, special software for loading programs from the floppy diskette into memory, and special versions of the COSMAC resident editor, assembler, and utility programs.

The ROM set provided with the FDS contains a disk utility and monitor program; the system diskette contains the following programs:

- an enhanced assembler program
- an enhanced editor program
- diagnostic programs
- file movement programs
- diskette copying program
- memory save program
- demonstration program

Additional new programs for the FDS are being continuously developed. For example, a new higher-level system for the FDS will be available soon.

The outstanding feature of the floppy-disk software aids is their extreme speed advantage over those provided for use with teletypewriter terminals. For example, the assembly of a one-kilobyte program takes approximately ten minutes with the FDS, but over an hour with a teletypewriter terminal. The floppy disk's speed advantage is very important because this loading and running of the assembler program takes place many times in the course of application software development.

Table I
Subroutine execution times for the CDP1802. Measurements are at a 6.4-MHz clock rate and are rescaled with a change in the system clock rate.

Arithmetic functions

	Add	Subtract	Multiply	Divide
Best	0.042	0.040	0.872	1.405 ms
Worst	0.070	0.080	1.320	1.830 ms

Format conversion

	Binary-BCD	BCD-Binary
Best	1.366	0.097 ms
Worst	2.897	0.834 ms

Subroutine library

In many programs there is often a degree of commonality. To help the COSMAC user develop his program with the least amount of "reinvention" possible, a well documented program library is available. The manual *Binary Arithmetic Subroutines for RCA COSMAC Microprocessors*, MPM-206, for example, describes an extended-precision arithmetic package.

The Binary Arithmetic Subroutine Package is a set of 16-bit 2's-complement arithmetic subroutines designed to operate on COSMAC CDP1802 microprocessor systems, including the COSMAC Development System. The subroutines are coded in Level I assembly language and require 1 kilobyte of memory space; they are also available in source language on a floppy diskette for use with the RCA Floppy Disk System.

A total of 31 subroutines are included: 15 are binary arithmetic subroutines, 14 are utility subroutines, and 2 are for format conversion.

The *arithmetic functions* included in the package are 16-bit 2's-complement addition and subtraction, 16-bit 2's-complement multiplication yielding 32-bit products, and 32-bit 2's-complement division yielding 16-bit quotients and remainders.

A set of special *utility subroutines* allows the user to save and restore a group of registers on a stack or at a user-defined RAM area. These registers are used by the arithmetic-function subroutines to store an operand and point to operands in memory. Other utility subroutines compare 16-bit operands and indicate if a register is greater than or equal to an operand.

The two *format-conversion subroutines* in the package are for interfacing the system to BCD-oriented peripheral hardware. These subroutines provide BCD-to-binary and binary-to-BCD conversions.

The Standard Call and Return Technique, described in the *User Manual*, MPM-201, can be used for all the subroutines.

Table I gives time measurements at a 6.4-MHz clock rate for the best and worst cases of the various arithmetic and format conversion subroutines for the CDP1802. The timing, however, is rescaled by a change in the system clock rate.

A 32-bit *floating-point software package* is now under development. It is intended for use in applications that require high precision in numbers of very large or small magnitude. In addition to the standard four functions (addition, subtraction, multiplication and division) there are also subroutines to perform trigonometric, inverse trigonometric, square-root, and log functions. The total package is less than 2 kilobytes long and will be available in source form. A complete manual will be issued with its release.

Summary

In summary, the COSMAC Development System is supported by a compatible family of resident software aids. Each is designed to help automate the generation of software required by products involving the RCA1800 microprocessor series.

COSMAC support literature: keeping users informed

One reason for COSMAC's acceptance is the extensive body of understandable literature supporting the microprocessor and its associated development systems.

C.A. Meyer

RCA 1800 Microprocessor
Binary Arithmetic Subroutines for RCA COSMAC Microprocessors
RCA 1800 Microprocessor
Timesharing Manual for the RCA CDP1802 COSMAC Microprocessor
RCA 1800 Microprocessor
User Manual for the CDP1802 COSMAC Microprocessor
RCA 1800
Instruction Summary for the CDP1802 COSMAC Microprocessor
RCA 1800 Microprocessor
RCA COSMAC Floppy Disc System CDP18S800 Instruction Manual
RCA 1800 Microprocessor
Operator Manual for the RCA COSMAC Development System

Because many of the potential users of microprocessors have had limited experience with computers, the microprocessor manufacturer must provide a broad range of supporting literature, from the fundamental to the advanced. Recognizing this need, the SSD Microprocessor Products activity has been supporting a well-organized program for the preparation and publication of a large amount of technical literature dealing with microprocessor hardware, software, and firmware. Moreover, it is continuously updating this literature so that users can keep up with the state of the art and thus be able to make optimum use of microprocessor-based systems.

Introductory support literature

Early in 1976, the CDP1802, a one-chip CMOS 8-bit register-oriented central processing unit, was announced. With it was introduced a family of memory and I/O support components, hardware, and software support systems suitable for testing and developing new microprocessor applications, an extensive data sheet for each new component, and a brochure (2M1137) describing the overall program. Part of the early planning for this program included the development of a comprehensive manual written to help users through the maze of new hardware, new software, and new systems.

User Manual—the best place to start.

Such a manual, the *User Manual for the RCA* CDP1802 *COSMAC Micro-*

Reprint RE-22-5-16
Final manuscript received December 10, 1976.

processor, MPM-201, was issued within a few months of the new microprocessor announcement. This manual, written for electronics engineers having only a limited familiarity with computers and computer programming, guides the reader gently through the microprocessor architecture and introduces a set of comprehensive, easy-to-use programming instructions. The 115-page, well-illustrated (125 figures) manual provides a detailed guide to the application of the CDP1802 microprocessor in a wide range of stored-program computer systems, both special and general-purpose.

Many examples are given to illustrate the operation and use of each of the CDP1802's 91 instructions and so aid the design engineer in developing simpler and more powerful products using the wide range of microprocessor capabilities. For systems designers, this *User Manual* illustrates practical methods of adding external memory and control circuits. It also has detailed examples of the use of I/O interface lines, including DMA and interrupt inputs, external-flag inputs, command lines, processor state indicators, and external timing pulses.

The manual covers various programming techniques, with examples, as well as more advanced topics, such as the use of subroutines, interrupt service, and RAM and register allocation.

The *User Manual* is probably the most suitable entrance point for those interested in learning COSMAC microprocessor fundamentals.

Systems support literature

Three major systems, available since the introduction of the CDP1802, help simplify microprocessor programming and application. Two are a combination of software and hardware; one is software only. These three systems, the Evaluation Kit, the COSMAC Development System (CDS), and the COSMAC Software Development Package (CSDP), each have their own comprehensive technical manuals.

Use the Evaluation Kit Manual to learn all you need to build prototypes.

The *Evaluation Kit Manual for the RCA COSMAC Microprocessor,* MPM-203, describes how to install and use the Evaluation Kit CDP18S020 and the hardware and firmware associated with it. The Evaluation Kit provides the key hardware and firmware elements for a computer system based on the CDP1802 Microprocessor. With this kit, augmented with an input/output terminal and a power supply, the user can readily prototype dedicated systems and evaluate software, components, and systems operation.

The *Evaluation Kit Manual* provides detailed information on the kit, its components, its overall configuration, and how it operates. The manual tells how the kit is assembled, how it can be used with various I/O terminals, how to troubleshoot hardware, how the various systems are designed, how they operate, how to check out software, and how to make use of the resident firmware. In addition, the manual contains a number of application notes describing the I/O and control features, the memory systems available and their utilization, and the use of resident firmware for Read and Type routines.

Because the development of microprocessor-based applications is a very dynamic activity, SSD Microprocessor Products is committed to a supporting program of application information and providing a continuous updating to Evaluation Kit users. As part of this update program, application note ICAN-6623 "Memory Address Function for the RCA Microprocessor Evaluation Kit CDP18S020," has been issued. Others are in process.

The Operator Manual is for more sophisticated development work.

The COSMAC Development System CDP18S004, a more extensive prototyping system than the Evaluation Kit, is covered in its technical supporting literature, the *Operator Manual for the RCA COSMAC Development System,* MPM-208. This 133-page manual is an operating guide on using a major prototyping aid for the design of hardware and software systems using the CDP1802 microprocessor. The COSMAC Development System (CDS) is specially designed and structured to provide a testbed in which the hardware and software prototypes of systems containing a microprocessor may be designed, built, and tested.

The *Operator Manual* includes a detailed description of each of the hardware modules in the CDS and an explanation of the functions available from the software supplied with the system. The hardware includes a rack-mountable card nest, a self-contained power supply, an easy-to-use control panel, and a basic set of eleven plug-in modules, including CPU, clock and control, address latch, memory bank select, 4-kilobyte RAM, I/O decoder, bus separator (2), byte I/O port, terminal interface, and monitor. The software supplied, which runs on the CDS in a stand-alone manner, provides a means for rapid coding and debugging of COSMAC programs. The software includes a resident assembler, editor, and utility program having a debugging facility.

The *Operator Manual* MPM-208 also describes input/output interfacing, memory module addressing, and many very useful special operating considerations.

Application notes keep you up to date with system add-ons and improvements.

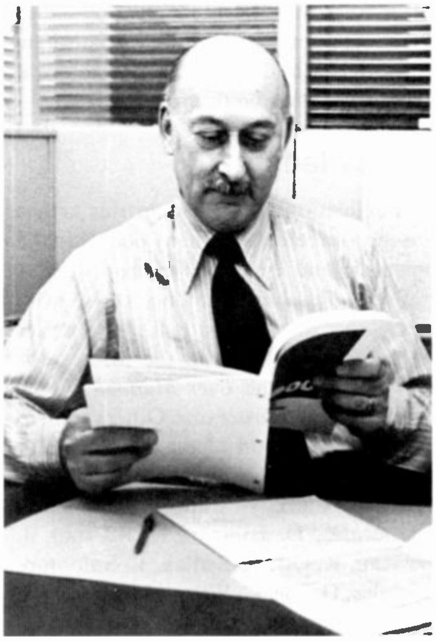
Concurrent with the publication of the *Operator Manual,* a series of application notes has been published describing various circuits and techniques for augmenting the utility of the COSMAC Development System. Notes in print include the following:

- Programable Interval Timer and Counter for Use with RCA COSMAC Development Systems—ICAN-6482
- Interrupt Priority Resolution Circuit for Use with RCA COSMAC Development Systems—ICAN-6485
- Adding Two-Level I/O Interface Capability to RCA COSMAC Development Systems—ICAN-6486
- Hexadecimal Display for Use with RCA COSMAC Development System—ICAN-6487
- Alphanumeric Display for Use with RCA COSMAC Development Systems—ICAN-6488
- Analog-to-Digital Converter for Use with RCA COSMAC Development Systems—ICAN-6490

- Digital-to-Analog Converter for Use with RCA COSMAC Development System—ICAN-6489
- PROM Programmer for RCA COSMAC Development Systems—ICAN-6491
- Hexadecimal Keyboard Interface for Use with RCA COSMAC Development Systems—ICAN-6516
- Register-based Output Functions for RCA COSMAC Microprocessors—ICAN-6562
- Design of Clock Generators for use with RCA COSMAC Microprocessor CDP1802—ICAN-6565
- Power on by Reset/Run Circuits for the RCA CDP1802 COSMAC Microprocessor—ICAN-6581

Charles Meyer has worked on technical literature for a wide variety of technical products since joining RCA in 1946. He has worked on microprocessor manuals, application notes, and other technical publications for the Solid State Division since January 1976. He has served on the advisory board for *TREND*, as a consulting editor of the *RCA Engineer*, and an associate editor of the *RCA Review*.

Contact him at:
Engineering Publications
Solid State Division
Somerville, N.J.
Ext. 6548



For rapid program development, see the Floppy Disk Manual.

Another major piece of technical literature supporting the COSMAC Development System is the *RCA COSMAC Floppy Disk System* CDP18S800 *Instruction Manual*, MPM-211. This manual describes a mass-memory storage unit designed for use with the CDS CDP18S004 to facilitate rapid program development using the advantages of the floppy disk over paper-tape or cassette devices. The manual describes the hardware interfacing to the CDS, the software for loading from a diskette into RAM, and the special versions of the COSMAC resident editor and assembler programs supplied on a diskette.

The manual also describes the special operations required when the floppy disk system is used, how to install the interface card, how to install the ROM set, how to set up the drive mechanism, and what preventive maintenance care to exercise. The Disk Utility and Monitor program is also described, as well as the Loader routine and the Read and Write routines. The manual also gives information on the Disk Assembler and the Disk Editor programs, along with examples of their use. The eight appendices to this manual include program listings and other useful information.

The Timesharing Manual gives you the information to develop microprocessor programs without new hardware.

The third major support system for microprocessor users, the COSMAC Software Development Package (CSDP), has an instruction manual as well as other supporting literature. The *Timesharing Manual for the RCA* CDP1802 *COSMAC Microprocessor*, MPM-202, provides a comprehensive guide to CSDP users. CSDP, with the information provided in its supporting manual, permits the user to develop microprocessor programs without any new hardware, using only a timesharing terminal for existing in-house or leased computer facilities. Through the use of CSDP-augmented timesharing or in-house computer facilities, software and hardware can be developed concurrently.*

CSDP facilities include a cross-assembler, an error-checking simulator, and powerful debugging tools. CSDP is presently available on commercial timesharing systems and, as a source program in FORTRAN IV, it is available with a detailed installation manual (MPM-204) for installation in most computer facilities.

The *Timesharing Manual* reviews the architecture, organization, and operation of the CPU and provides the programmer with a means of writing and modifying programs using convenient mnemonic symbols rather than machine language. The cross-assembler translates these symbols into machine language.

The COSMAC assembly language requires no fixed-field alignment, permits the use of blanks for added readability, and permits multiple instructions per line. By also permitting the use of comments on any line, it enables the program to be self-documented.

Additionally, the *Timesharing Manual* provides detailed coverage of the CSDP constituents, pertinent programming techniques, and a very valuable description of common programming "bugs." The manual describes the development of a sample program including the very important debugging steps. Six useful appendices are included in this 88-page manual, as well as a comprehensive index.

Other CSDP-supporting literature includes the *Installation Manual for COSMAC Software Development Package*, MPM-204, mentioned above, which describes how to install the CSDP in a timesharing in-house computer facility. This manual is intended for programmers acquainted with FORTRAN. In addition to step-by-step installation information, the manual describes some specific featues, not standard FORTRAN, which provide additional utility and convenience to the programmer.

Other key manuals

As part of the overall planning, hardware and software are being continuously developed to augment COSMAC capabilities. In the software area, one important set of programming aids available on a floppy diskette (CDP18-S826) is also covered in a manual. This

*CSDP is available to RCA users on CMS in Cherry Hill and TSO in Somerville.

manual, *Binary Arithmetic Subroutines for RCA COSMAC Microprocessors,* MPM-206, describes a set of 16-bit 2's-complement arithmetic subroutines designed to be operated on CDP1802 microprocessor systems including the COSMAC Development System. In the *Binary Arithmetic Subroutine Manual*, 31 subroutines are descrixed. Fifteen are binary arithmetic subroutines, 14 are utility subroutines are described. Fifteen are sion. The arithmetic functions included are 16-bit 2's-complement addition and subtraction, 16-bit 2's-complement multiplication yielding 32-bit products, and 32-bit 2's-complement division yielding 16-bit quotients and remainders. The 62-page manual includes the complete program listings as well as a sample program illustrating its use.

Another software development, which will be available on a diskette and for which a manual is in preparation, is loosely referred to as the "floating-point package." This package is a group of advanced mathematical subroutines that further enhance COSMAC microprocessor capabilities.

RCA Microprocessor Manuals are available to RCA employees for their use at a considerable reduction from list price.

		List price	*Employee price*
MPM-201A	User Manual for the RCA CDP1802 COSMAC Microprocessor	$ 5.00	$3.00
MPM-202	Timesharing Manual for the RCA CDP1802 COSMAC Microprocessor	10.00	6.00
MPM-203	Evaluation Kit Manual for the RCA COSMAC Microprocessor	15.00	9.00
MPM-206	Binary Arithmetic Subroutines for RCA COSMAC Microprocessors	5.00	3.00
MPM-208	Operator Manual for the RCA COSMAC Development System	10.00	6.00
MPM-211	RCA COSMAC Floppy Disk System CDP18S800 Instruction Manual	10.00	6.00
MPM-212	Instruction Manual for the RCA COSMAC Microterminal	2.00	1.20
MPM-109	COSMAC Microtutor Manual	2.00	1.20

To purchase these manuals, send order with personal check, IPR, or department charge number to:

Publication Services,
Mail Stop 37
RCA Solid State Division
Somerville, N.J.

In the hardware area, a very interesting new development is the COSMAC Microterminal CDP18S021. This data terminal is accompanied by the *Instruction Manual for the RCA COSMAC Microterminal* CDP18S021, MPM-212. This 28-page manual describes the installation and application of a portable hand-held data terminal designed for microcomputer systems using the CDP1802 microprocessor. It is a low-cost, low-power alternative to a conventional data terminal such as the teletypewriter. The Microterminal, together with its associated programmed ROM, provides a means of controlling a COSMAC-based system and supplies hexadecimal I/O capability. The Microterminal is designed to interface directly with COSMAC support hardware systems such as the CDP18S020 Evaluation Kit, but it can also be readily designed into user-built systems to provide the same control, communications, and debugging functions.

The manual includes a description of the hardware and the software programs available in the ROM supplied with the Microterminal and also includes installation instructions and utility-program listings. The operating modes of the Microterminal are described as well as several examples of operating sequences.

COSMAC fun and games

There is one more manual that merits special attention, the *COSMAC Microtutor Manual*, MPM-109. This Manual is an instruction book supplied with a very unusual piece of equipment, the CDP18S011 Microtutor. The Microtutor is a small, simple, inexpensive, and easy-to-understand computer. It comprises 256 words of memory, input switches, a two-digit output display, and the COSMAC microprocessor. The Manual tells how to use the Microtutor to learn about microcomputers with fun and games on a simple computer; it also contains a number of programs for just fooling around. And after the user is hooked on how easy it is to work with binary numbers and hexadecimal code, he is shifted smoothly into COSMAC structure and program development. Before he realizes it, he is a microcomputer expert. Not really, but it is fun to work with and quite a bit of skill and knowledge rubs off. The Microtutor manual is a 48-page gem—I wish I had written it.

Importance of technical support

In the microprocessor field, particularly, customers measure a supplier not only by the hardware, software, and field engineering support that he provides, but also by the quality, availability, and comprehensiveness of his supporting literature. The nine manuals described here, together with the application notes and technical data sheets, already constitute a substantial library providing technical support to COSMAC microprocessor users. With the continuing development of more COSMAC hardware, software, and systems, supporting literature will continue to be provided and the COSMAC technical library will continue to grow.

Acknowledgments

The publications described in this article were the joint efforts of many people in the Somerville and Princeton microprocessor activities. Foremost among those who deserve special mention is J. Weisbecker for his original work on the *Microtutor Manual* and on the *User Manual for the COSMAC Microprocessor*. Others who, to the best information of the author, merit acknowledgments include D. Block, D. Carley, W. Clark, H. Edinger, E. Fulcher, A. Gonzales, D. Hillman, K. Karstad, J. Oberman, R. Ott, R. Sedlak, L. Solomon, N. Swales, G. Tse, C. Wu, and A. Young in Somerville, and M. Lippman, A. Marcantonio, and P. Russo in Princeton.

Five RCA engineers elected IEEE Fellows

The membership grade of Fellow is the highest attainable in the Institute of Electrical and Electronics Engineers. This recognition is extended each year to those who have made outstanding contributions to the field of electronics.

Morrel P. Bachynski

for contributions to the fields of electrogagnetic waves and plasmas

Dr. Bachynski's work concentrated on electromagnetic wave propagation, microwaves, and plasma physics. He was Vice President of the R&D Division of RCA Limited, Canada. (This year a large portion of this division became MPB Technologies, Inc., a new company headed by Dr. Bachynski.)

Bernard J. Lechner

for contributions to flat-panel displays and two-way cable television systems

Mr Lechner has done research on video tape recorders, high-speed computer circuitry, and solid-state displays; he has also directed a group doing pioneering work on two-way cable tv systems and pay-tv systems. He is presently Head, Color Television Research Group, Systems, Research Group, RCA Laboratories, Princeton, N.J.

Josh T. Nessmith, Jr.

for leadership in and technical management on radar systems engineering

Dr. Nessmith has been engaged in advanced radar and fire control system development programs, including systems engineering work on AEGIS. As Manager, Systems, Missile and Surface Radar, Moorestown, N.J., he provides system engineering direction and resources for radar and weapon system design.

Dalton H. Pritchard

for contributions to the development and improvement of color television

Mr. Pritchard has worked primarily in research on color television circuit and systems development and information display systems and devices, including the VideoDisc program and CCD devices. He is a Fellow of the Technical Staff of RCA Laboratories, Princeton, N.J.

Robert O. Winder

for contributions in switching theory, comptuer structures, and microprocessor design and applications

Dr. Winder has worked in computer systems, threshold logic, computer-aided design, data communications, and microprocessor design and applications. He was in charge of the group that developed the COSMAC microprocessor and other LSI technology systems. He is Director, Microprocessor and Memory Products, Solid State Division, Somerville, N.J.

From the top: **Bachynski**
Lechner
Nessmith
Pritchard
Winder

on the job/off the job

COSMAC Applicat

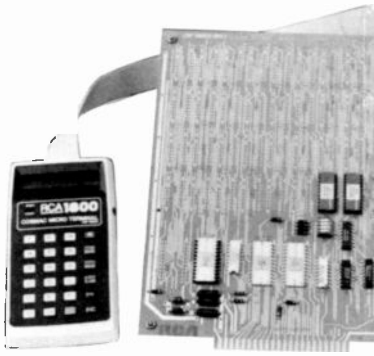
Think of a novel way to use RCA's COSMAC microprocessor. You can gain recognition for your idea and compete for valuable prizes by documenting the application and entering it in the COSMAC applications contest that the Solid State Division is sponsoring in conjunction with the RCA Engineer.

Who is eligible

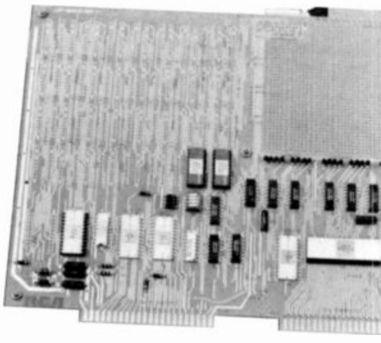
Any RCA employee can submit an application. You do not have to receive the *RCA Engineer*. Retired RCA employees are also eligible. Those employees working with microprocessors as part of their regular job assignments are eligible also; however, the applications they submit *can not be a present* or *past job assignment.*

Here are the prizes

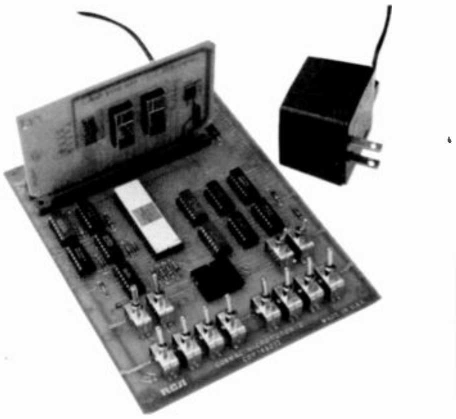
Five prizes will be awarded:



First prize — an Evaluation Kit wit



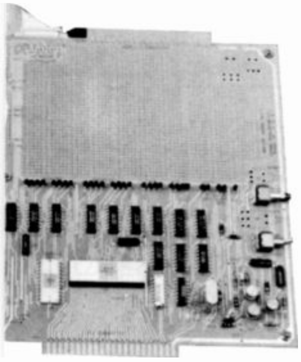
Second prize — an Evaluation Kit



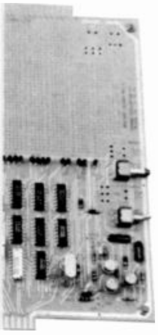
Third prize (three awarded) — a M

In addition, all entries will be ackno
will be reviewed for further publica

tions Contest



ı a Microterminal



icrotutor

wledged in the *RCA Engineer* and tion and patent possibilities.

Here's how to enter

Document your application sufficiently to clarify design details (the parts used, the interconnections, and the software). You don't have to build or debug your design, but you should finish the design to the point where someone else could build and debug it.

Once you have completed the documentation, send all of it to:

COSMAC Applications Contest
c/o *RCA Engineer*
Bldg. 204-2
Cherry Hill, NJ 08101

You can use internal mail. (Be sure to include your return address and phone number.)

All entries must be received at the *RCA Engineer* office no later than June 1, 1977. Winners will be announced in the August/September 1977 issue of the *RCA Engineer.*

Because these applications may further RCA's business interests, all entries must be submitted under the agreement that they become the property of RCA Corporation.

Criteria for judging

The judges, who are RCA engineers working in microprocessor application areas, will apply the following criteria:

- Originality—Has anyone thought of the idea before? If it's not new, has anyone worked out a practical solution?

- Efficiency of design—Does it use the lowest number of parts/function and is it cost-effective?

- Completeness of the design—How complete is the hardware design diagram and the program flowchart? Has the software coding been done?

- Importance of the problem solved—How useful is the application?

- Effective utilization of RCA components—Does the application use RCA parts wherever possible?

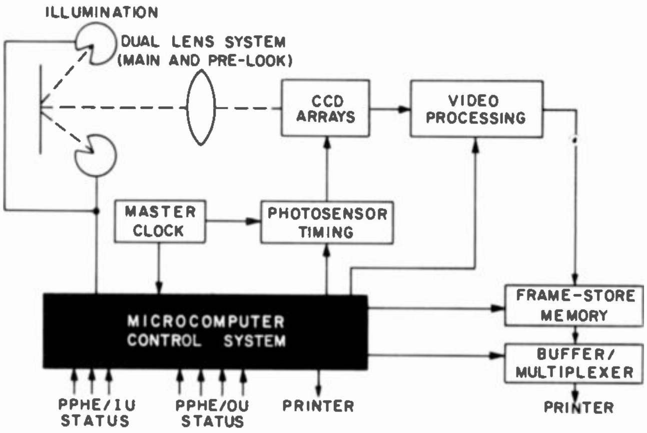
Decisions of the judges are final.

How RCA engineers are using microprocessors

A quick survey throughout RCA shows that microprocessors are being put to work in highly divergent end uses.

Proposed message system to read 10 pages/second

Recently, RCA proposed a program to develop a scanner subsystem that would convert incoming messages to electrical signals for an electronic message service system. This system is designed to perform a vital function for printing and paper-handling equipment (PPHE)—reliably read incoming messages at rates up to ten pages per second. RCA has proposed to use the COSMAC CDP1802 microprocessor controller to monitor the interface signals to and from the PPHE input unit (IU) and PPHE output unit (OU), and then control the scanner subsystem processing, storage, and outputting of data. The microprocessor would provide the necessary command and status signals to maintain a smooth flow of valid documentation information. Decision-making programs would be stored in ROM, so modifications in scanner-subsystem operation could be made by simply reprogramming the ROM.



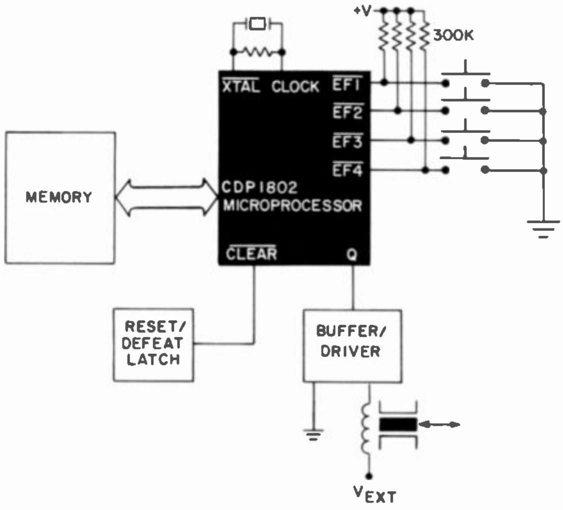
Alan Feigenbaum
Advanced Technology Laboratories
Camden, N.J.
Ext. PC-2853

Digital combination lock has 10^9 reprogrammable combinations

Requiring only five integrated circuit packages, this electronic lock has its combination entered via four pushbutton switches. The debouncing of the switches is handled by the software and only the 4 "tumblerless" switches are visible to the user. The input code is compared against the combination bank by the software, and an error forces the entire sequence to be reentered to open the lock. Internal timing chains force a reset if an excessive time delay occurs between successive inputs. Total current drain is minimal (5mA typical at V_{DD}= 5V and a 1.6-MHz clock), providing battery holdup for extended periods in the event of a power failure.

The heart of the lock is the RCA CDP1802 COSMAC 8-bit microprocessor. The combination code is entered via the 4 serial-input $\overline{EF}$ lines. The lock mechanism is driven by a high-current buffer/driver, which is controlled by the Q flip-flop output of the CDP1802. The Q bit is set or reset under program control in response to the input code.

The number of possible combinations (p) is a function of the available memory storage locations (x), where $p = 4^x$.



This lock has 4^{15} or 1.07 X 10^9 possible combinations. An application note describing the digital combination lock is in progress.

William Clark
Solid State Division
Somerville,N.J.
Ext. 7364

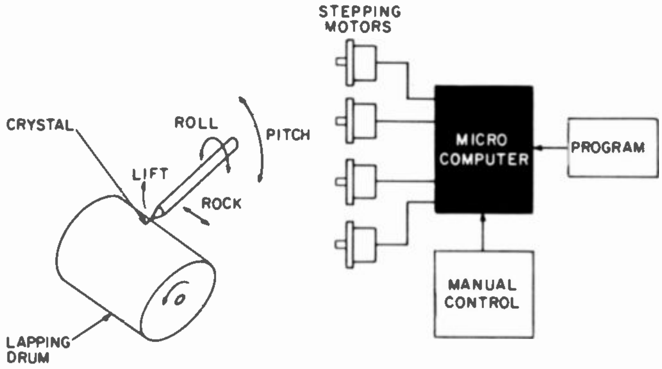
Reprint RE-22-5-6

Manufacturing

Programmable microcomputer shapes VideoDisc crystal

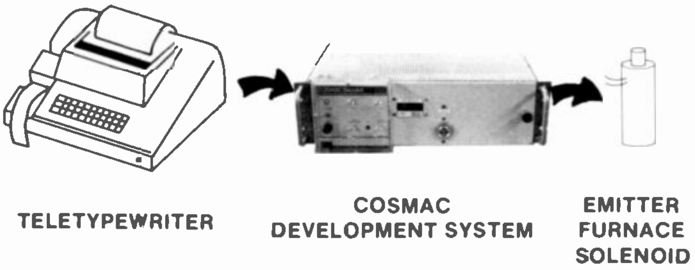
The stylus for playing the RCA VideoDisc is a minute sapphire crystal that must be machined with extreme accuracy. The machine that accomplishes this task controls the crystal's four main motions, whose timing and sequencing must be highly precise and repeatable.

A newly designed production lapper uses four stepping motors whose motions are dictated by a programmable control system (Texas Instruments' 5T1). The microcomputer provides the precision and repeatability required. In addition, making changes in the program permits modifications in the crystal shape and the generation of new shapes.



B.E. Mesa
SelectaVision Project
Indianapolis, Ind.
Ext. VR-3237

Microcomputer controls bipolar IC emitter furnace



A COSMAC Development System has been controlling a diffusion furnace at the Solid State Division's Findlay, Ohio plant since June 1976, three shifts a day. Stored in the system's memory are over 100 diffusion schedules for the different device types presently in production. When the operator inputs the device types making up the furnace load via the teletypewriter, the system checks the schedules to make sure that the entire furnace load requires the same schedule. Error message and alarms prevent incorrect combinations.

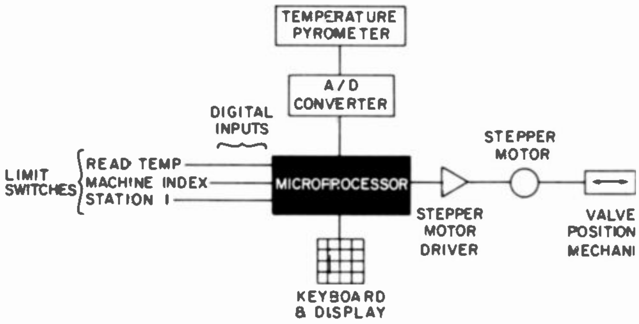
If all the device types in the load call for the same diffusion schedule, the microcomputer types out a RUN PROCESS message and indicates what the schedule will be. The load is then put in the furnace tube, a button is pushed, and the furnace time-out is under control of the microcomputer. A solenoid is actuated at the proper time to introduce dopant gases into the furnace tube. An LED display on the front panel indicates time remaining; when the process is complete, an audible alarm alerts the operator.

Luke Dillon
Manufacturing Systems and Technology
Cherry Hill, N.J.
Ext. PY-4381

Microprocessor control for glass forming

An indexing glass-forming machine used at the Picture Tube Division's Circleville plant has eleven stations, with a mold at each station. Although cooling air is applied to each station, the air-control valve is adjustable only at one station and so remains at that position for the other ten. Also, the mold temperature can only be read at one station.

To control the temperature of each mold, the temperature history of the individual molds must be stored, along with their unique control-valve positions. Then, when a mold indexes to the control station, the microprocessor performs calculations to determine the adjustment for that cooling valve.



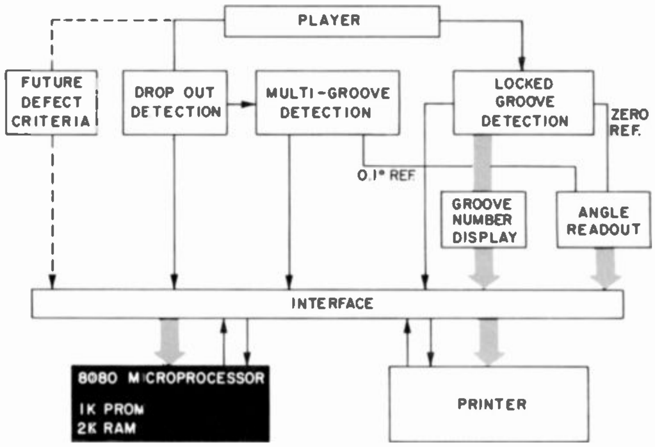
Richard W. Marshall
Picture Tube Division
Circleville, Ohio
Ext. 312

Testing

Microprocessor control for VideoDisc testing

A microprocessor-controlled test system is being designed to locate the position of defects that may be present on the surface of a VideoDisc. When a specific type of defect is detected, the radius and angle of its location are printed out. A microprocessor was chosen for this application because of its speed, cost, and data-storage and data-manipulation capabilities.

A three board Pro-Log microprocessor system with an Intel 8080 CPU was chosen because complete card systems that were compatible with existing test systems were available with PROM capability.



J.W. Stephens
SelectaVision Project
Indianapolis, Ind.
Ext. VR-3088

Microprocessor system tests vehicle engines rapidly and accurately

The STE/ICE (simplified test equipment/internal combustion engine) system performs more than 45 types of tests and measurements on transport and tactical vehicle engine and accessory systems. In addition to measuring pressure, temperature, speed, voltage, and current, the system electronically performs a power test on gasoline and diesel engines without the need for an external dynamometer.

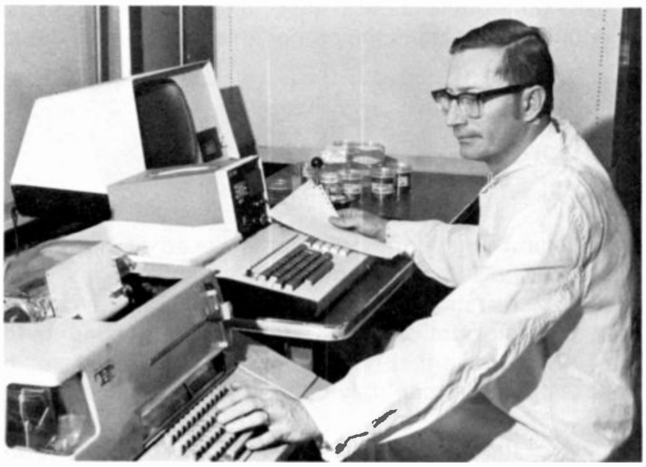
Using the test meter shown here with a Diagnostic Connector (connected to built-in transducers and test points in the vehicle), a mechanic needs only a few minutes to perform a series of tests that usually takes hours using conventional test equipment. This flexibility in a hand-held vehicle test meter is achieved by applying a microprocessor for measurements, conversions, displays, and solving test algorithms. The first prototype units, which were designed in 1973, used the Rockwell PPS-4 microprocessor. The production design, which will be fielded in 1978, uses RCA's COSMAC CDP1802 microprocessor.

R.T. Cowley
Automated Systems
Burlington, Mass.
Ext. 3185

Programmable calculator produces savings during transistor manufacturing

In the manufacture of hometaxial transistors, it is necessary to categorize wafers after boron deposition and prior to base diffusion by measuring their voltage-current (V-I) characteristics. The reason for doing this is that each V-I range requires a different base-diffusion time, and an over-diffused wafer is a lost wafer. A system has been put together using a microprocessor-based programmable calculator to do the following: take 5 V-I readings on the wafer; calculate the average and categorize it; present the operator with the category; maintain lot history; present histograms on demand; and record all historical data on a cassette at the end of a lot run.

The system has been operating for about nine months, and has increased first-run sales in the diffusion department by about 50%. In addition to the obvious savings, it would have been impractical to make these measurements by conventional methods because of their excessive labor content.



E. M. Mihalick
Solid State Division
Mountaintop, Pa.
Ext. 646

Communications and signal processing

Using a microprocessor as a coordinate converter

Test-range instrumentation equipment usually requires a means of obtaining real-time target acquisition information to and from a remote location. A data format using the earth-centered EFG coordinate system is in common use today. However, the instrumentation sensor must transform data from the EFG system to the local XYZ cartesian and Azimuth-Elevation-Range (AER) systems to be useful.

Most microprocessors are 4- or 8-bit machines and do not provide sufficient calculation accuracy to do this without lengthy software algorithms. Since the conversions should be performed at a 10-pps rate, such a processor soon runs out of time. The Digital Equipment LSI-11 microprocessor, however, has enough accuracy and speed for this because it is a 16-bit machine with an extended arithmetic option that provides double-precision fixed-point anc floating-point instruction. If the LSI-11 converter program is placed into a read-only memory, the system performs like a piece of hardwired equipment and needs no peripheral equipment.

D.J. Kleinschnitz
RCA Service Co.
Patrick AFB, Fla.

Spacecraft command and control system uses two COSMAC microprocessors

A unified low-power command and telemetry system using RCA CDP1802 microprocessors has been developed at RCA Astro-Electronics. The system's microprocessor-based central control unit interacts with two subsystems—the command decoder, itself microprocessor-based, and the remote distribution units (RDUs).

A transformer-coupled data bus carries supervisory command and telemetry words to the RDUs, which then return telemetry data to the central control unit for output buffering and final formatting. The command decoder takes signals from the spacecraft receiver and performs message inspections and parity checks on them. The subsystem, which consists of less than one board of logic, decodes a set of 256 critical commands and directly distributes them to perform mission-critical command operations. All other properly formatted uplink data are transferred to the central control unit; the command decoder informs the ground station of improperly received data. The central control unit can perform both real-time and delayed-time processing. It also provides fixed and programmable telemetry format options, plus several rate options and telemetry storage features.

Charles Staloff
RCA Astro-Electronics
Hightstown, N.J.
Ext. 2256

Microprocessors in radar systems

Current applications of microprocessors to radar systems have centered on bit-slice bipolar microprocessors; two functions are considered prime candidates for microprocessor implementation in a phased-array radar system.

Coordinate-conversion is currently being implemented for a phased-array radar using the Advanced Micro Devices 2901 bit-slice microprocessor in a 24-bit configuration. This processor interfaces with a three-angle attitude reference system and performs the matrix calculations required to translate the moving antenna coordinates to stable coordinates. The result is a 50% processing load reduction for one of the main system central processing units, making vital computation capacity available for other functions.

A beam-steering controller is also being implemented using a 16-bit processor to compute the phase tapers required to point the array. Other radar microprocessor applications under consideration include signal-processing functions, such as interpolation and monopulse implementation, and range and angle servo processors for tracking radars.

R.J. Smith
Missile and Surface Radar
Moorestown, N.J.
Ext. PM-2703

Low-power "micro-signal processor" made for smart sensor application

A signal processor based on the 8-bit COSMAC CDP1802 has been developed and brassboarded for sensor application. The processor's multiplier-accumulator circuit provides the micro with the substantially improved computational capability required to perform the various processing functions associated with "smart" sensors. The signal processor can automatically and periodically sample the analog outputs from seismic and/or acoustic transducers and then perform feature extraction and a variety of pattern-recognition algorithms suitable for target identification, discrimination and classification. Feature extraction is currently based on frequency-domain analysis using the FFT algorithm. Computations are done with either 16-bit fixed-point or 24-bit floating-point accuracy. Specifications for the processor include the ability to perform, for example, a 32-point FFT within 20 ms, with power dissipation under 100 mW.

K. Prost, G. Dooley, and D. Hampel
Government Communications Systems
Somerville, N.J.
Ext. 6100

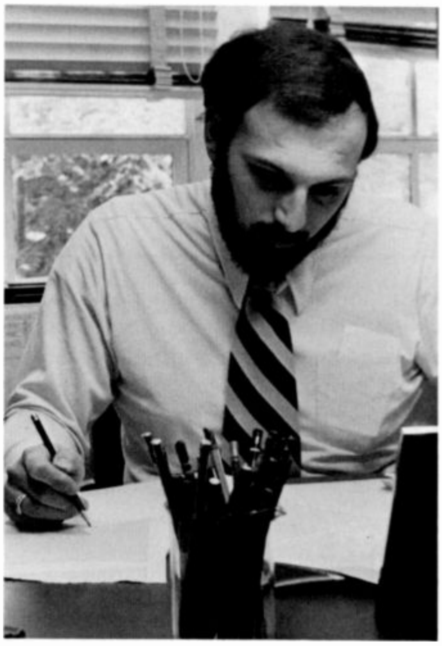
Microprocessors in telecommunications

M.D. Lippman

With microprocessors, telecommunications engineers can offer their customers new levels of cost/performance, increased reliability, and functions that were previously impossible or extremely costly to implement.

Michael D. Lippman joined RCA Laboratories in July 1966 as a member of the Research Training Program. In 1967 he joined the Communications Research Laboratory where he initially performed research aimed at developing efficient coding techniques for digital image storage and transmission. Since 1972 he has been working on microprocessor applications in data communications and broadcast equipment.

Contact him at:
Communications Research Laboratory
RCA Laboratories
Princeton, N.J.
Ext. 2955



Microprocessors are appearing in virtually every type of telecommunications equipment. In telephone switching systems, intelligent terminals, data network controllers, multiplexers, communications test equipment, and other applications, the microprocessor is such a fundamental and pervasive tool that it is markedly affecting the basic architecture of telecommunications systems.

This article explores some of the benefits of microprocessors as they relate specifically to telecommunications. Also, the effects that microprocessors are having on the evolution of system architecture are discussed. Examples of microprocessor-based telecommunications equipment are given to demonstrate the advantages to be gained, to illustrate the effects on architectural development, and to give the reader an appreciation for the variety of applications that already exist.

Cost

LSI reduces cost-per-function.

The telecommunications industry has been turning to digital technology to meet an explosive increase in demand for services and equipment. This growing reliance is due primarily to the plummeting cost-per-function of digital circuitry over the last decade, compared with a modest decrease for analog circuits. Through increasing levels of integration, the functional complexity of digital integrated circuits has doubled each year while costs have stayed relatively constant.

The microprocessor lowers cost-per-function even further than custom LSI.* This is so because the microprocessor is general and can be used to perform different tasks, whereas custom LSI devices are specific and require high production volumes before they are cost-effective. Therefore, many systems that could previously use only moderate levels of integration—the MSI** circuits used to implement hardwired, logic—can now realize the cost advantage of LSI via the microprocessor.

LSI circuits lower more than just component costs.

The smaller size and reduced power requirements of an LSI implementation, as compared to discrete logic, can greatly reduce power supply and packaging costs. In addition, fewer interconnections yield increased reliability, and this means lower maintenance costs. This is a very important consideration in telecommunications systems where equipment is often geographically remote and sometimes inaccessible. It is the combination of reduced component, system, and support costs that makes the LSI circuit so economically attractive. And, although discrete logic will always be most cost-effective for some applications, the microprocessor is extending the total cost advantage of LSI to many medium, and even low-volume, systems and devices.

Intelligent telephones are coming.

An excellent example of an application in which low cost and small size are of great importance is AT&T's Transaction™ telephone. This is a special telephone set that functions as a remote terminal for credit authorization. It features automatic dialing, magnetic card and touch-tone pad data entry, and automatic operator

*Large-Scale Integration.

**Medium-Scale Integration.

Reprint RE-22-5-9
Final manuscript received August 9, 1976.

Multiprocessor structure. Codex 6030 Series Intelligent Network Processor is built around an Intel 3000, controlling up to eight Motorola 6800S. It performs high level network management functions.

Monitoring and diagnosis. This B280 ESSS Maintenance Monitor by Lynch Communications Systems Inc. uses a microprocessor to isolate faults down to the printed-circuit-board level.

Flexibility. This GTD-120 stored-program PABX from GTE Automatic Electric is a good example of how the microprocessor can be used when a minicomputer cannot.

prompting. A Rockwell PPS-4 five-chip microcomputer is employed. This is a forerunner of what is almost certain to come: solid-state, "intelligent" telephone sets.

Flexibility

Telecommunications systems are constantly changing.

The programmability that makes the microprocessor a general-purpose (rather than specific) LSI device also allows microprocessor-based equipment to be reconfigured quickly and easily. Functional changes are made by changing the program, most often done simply by replacing a ROM. Thus modifications can be readily accomplished in the field, in sharp contrast to the rewiring required to change discrete logic, or the complete redesign needed with custom LSI. This flexibility is especially important in telecommunications, where systems must be very dynamic, expanding to meet new requirements and changing to accommodate new and better services, operating protocols, and hardware devices.

With microprocessors, flexibility is no longer a luxury.

To achieve this flexibility, minicomputers have been widely used in telecommunications applications. However, there are many areas in which the cost of minicomputers and their associated peripherals cannot be justified. In these applications, flexibility was previously a luxury that could not be afforded. But just as the microprocessor extends the economic benefits of LSI to lower volumes, it also extends the flexibility inherent in stored-program control to smaller, simpler systems and devices.

The M1308 Multitran multiplexer by Computer Transmission Corp. illustrates the flexibility gained with microprocessor design. This device, designed for point-to-point multiplexing between terminal clusters and central computers, can handle many different kinds of communications channels. By changing the program, the user can completely rearrange the channel-scanning and data-handling sequence—an extremely costly, if not impossible, feat for random logic.

Digital private automatic branch exchange (PABX) telephone switching systems are good examples of how the microprocessor can be used when a minicomputer can't. The addition of stored program control to PABXs allows great flexibility in assigning lines and sophisticated functions such as call forwarding, camp-on,* and multistation hunt groups.** In large systems, the cost of the common control is shared by hundreds of lines, and the cost per line is low. In this case, a minicomputer is cost-effective. In small switches with one hundred or fewer lines, the cost per line of minicomputer common control is too high. Therefore, until the microprocessor arrived there were few, if any, small stored-program PABXs. Now there are several, including the GTD-120 120-line, 28-trunk PABX from GTE Automatic Electric.

With the low-cost microprocessor, even very small telephone electronic switching systems, along the lines of the Call Director, are feasible, offering more flexibility, more functions, and much lower cabling costs than electromechanical, button-per-line devices.

New functions

The microprocessor has made new functions possible for telecommunications equipment.

The microprocessor can implement functions that are difficult or impossible to implement with random logic. Many examples exist among current telecommunications devices. The Transaction™ telephone described earlier would certainly not be feasible if implemented with discrete logic. The intelligence that is routinely built into terminals today was very difficult to achieve before the microprocessor. Intelligence is now being added to a wide variety of test equipment resulting in greatly expanded functional capabilities. For example, Hewlett-Packard's Selective Level Measurement Set can make unattended measurements of FDM transmission parameters. The computational capability of the Intel 8008 microprocessor around which this instrument is built allows automatic analysis and data reduction. Rhode & Schwarz makes a Radiotelephone Test Assembly that, among other functions, corrects operator errors. Self-calibrating communications test equipment is being developed using microprocessors. New functional capabilities, made feasible by the microprocessor, improve the accuracy and simplify the operation of a wide range of telecommunications equipment.

*In camp-on operation, a call placed to a busy number will be retried automatically until it can be completed.

**When a call is placed to a busy extension that is part of a multistation hunt group, other numbers in that group will be successively tried.

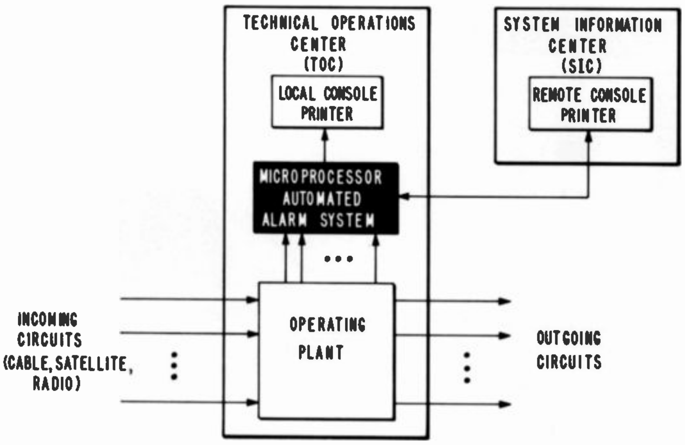

TECHNICAL OPERATIONS CENTER (TOC)
LOCAL CONSOLE PRINTER
SYSTEM INFORMATION CENTER (SIC)
REMOTE CONSOLE PRINTER
MICROPROCESSOR AUTOMATED ALARM SYSTEM
INCOMING CIRCUITS (CABLE, SATELLITE, RADIO)
OPERATING PLANT
OUTGOING CIRCUITS

Fig. 1
Microprocessor Automated Alarm System is used by RCA Global Communications, Inc., to monitor various communications channels. The RCA COSMAC microprocessor provides control for the system.

The microprocessor is being applied to the vital function of fault monitoring and diagnosis.

Built-in diagnostic capabilities have long been overlooked for telecommunications equipment because they were so costly and difficult to implement. Microprocessor-based diagnostic systems are beginning to appear on the telecommunications scene. An example is the B280 Maintenance Monitor built by Lynch Communications to isolate faults down to PC-board level in its B280 electronic subscriber telephone switching system.

Another fault-monitoring system of this type is the Microprocessor Automated Alarm System (Fig. 1) developed by RCA Global Communications. Built around the RCA COSMAC microprocessor, this system monitors the status of a variety of communications channels including transoceanic cables, satellite circuits, and radio links. The system, located at RCA Globcom's Technical Operations Center (TOC), generates line status reports on two console printers. A local console in the TOC receives failure and restoral reports (Fig. 2) for specific outages as they occur. A remote printer at a Systems Information Center (SIC) can request summary failure reports (Fig. 3) showing which lines are currently out of service. The current stand-alone system will eventually be tied into a microprocessor-based centralized monitoring system. It is almost certain that this type of diagnostic and monitoring capabilitiy will be built into most large telecommunications systems in the near future.

Systems architecture

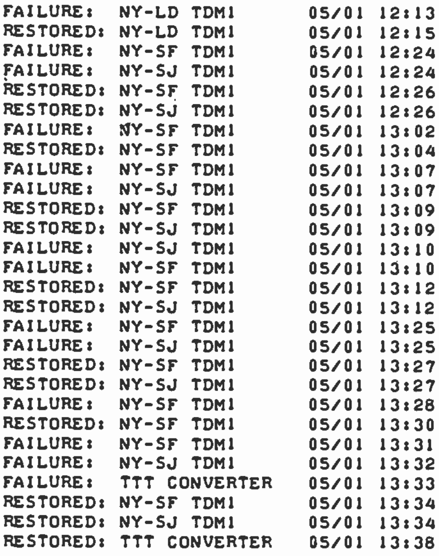
The trend towards distributed architectures is very strong in telecommunications because these systems are inherently distributed, both geographically and functionally.

The microprocessor has begun to influence the basic structure of many types of electronic systems. By offering very small, low-cost increments of processing power, the microprocessor allows the workload to be distributed among various components in a system. While the minicomputer started this trend in large systems, it has taken the microprocessor to extend the range of this architectural approach to smaller, simpler systems and devices. A classic example of the type of structures that are evolving is the use of smart terminals, remote devices that have processing capabilities and can function to some extent independently of the central processing facility. This arrangement offers several advantages. First, some routine tasks and preprocessing can be removed from the central processor, freeing it for more important work. Second, if the central processor or communications link becomes inoperative, the terminal can still function, even if only in a limited way. Thus, for example, remote data collection need not stop just because the central computer is down.

Front-end processors represent another important application.

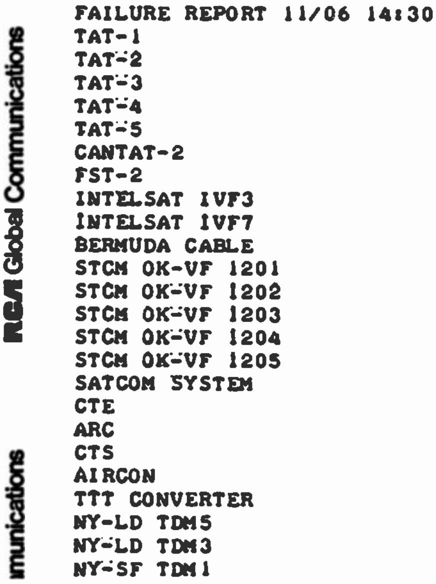
Another important distributed-system application for microprocessors is beginning to emerge: front-end communications processors for data networks. The front-end processor, usually implemented by a minicomputer, is designed to be connected between the central processor and its remote terminals (see Fig. 4). It performs most of the tasks associated with terminal communications, such as character and block assembly, error control, and line supervision, thereby freeing the main processor for more applications-oriented jobs. The front-end processor also assumes the burden of interfacing the specific terminals and devices, and presents a single, standard peripheral interface to the main processor.

The front-end processor must therefore be able to accommodate a wide variety of terminal types. In addition, it must be able to handle different communications



```
FAILURE:  NY-LD TDM1          05/01 12:13
RESTORED: NY-LD TDM1          05/01 12:15
FAILURE:  NY-SF TDM1          05/01 12:24
FAILURE:  NY-SJ TDM1          05/01 12:24
RESTORED: NY-SF TDM1          05/01 12:26
RESTORED: NY-SJ TDM1          05/01 12:26
FAILURE:  NY-SF TDM1          05/01 13:02
RESTORED: NY-SF TDM1          05/01 13:04
FAILURE:  NY-SF TDM1          05/01 13:07
FAILURE:  NY-SJ TDM1          05/01 13:07
RESTORED: NY-SF TDM1          05/01 13:09
RESTORED: NY-SJ TDM1          05/01 13:09
FAILURE:  NY-SJ TDM1          05/01 13:10
FAILURE:  NY-SF TDM1          05/01 13:10
RESTORED: NY-SF TDM1          05/01 13:12
RESTORED: NY-SJ TDM1          05/01 13:12
FAILURE:  NY-SF TDM1          05/01 13:25
FAILURE:  NY-SJ TDM1          05/01 13:25
RESTORED: NY-SF TDM1          05/01 13:27
RESTORED: NY-SJ TDM1          05/01 13:27
FAILURE:  NY-SF TDM1          05/01 13:28
RESTORED: NY-SF TDM1          05/01 13:30
FAILURE:  NY-SF TDM1          05/01 13:31
FAILURE:  NY-SJ TDM1          05/01 13:32
FAILURE:  TTT CONVERTER       05/01 13:33
RESTORED: NY-SF TDM1          05/01 13:34
RESTORED: NY-SJ TDM1          05/01 13:34
RESTORED: TTT CONVERTER       05/01 13:38
```

Fig. 2
Line status reports for the alarm system shown in Fig. 1 identify specific line failures and restorals as they occur.


RCA Global Communications
munications

```
FAILURE REPORT 11/06 14:30
TAT-1
TAT-2
TAT-3
TAT-4
TAT-5
CANTAT-2
FST-2
INTELSAT IVF3
INTELSAT IVF7
BERMUDA CABLE
STCM OK-VF 1201
STCM OK-VF 1202
STCM OK-VF 1203
STCM OK-VF 1204
STCM OK-VF 1205
SATCOM SYSTEM
CTE
ARC
CTS
AIRCON
TTT CONVERTER
NY-LD TDM5
NY-LD TDM3
NY-SF TDM1
```

Fig. 3
Summary failure report shows which lines are out of service at any given time.

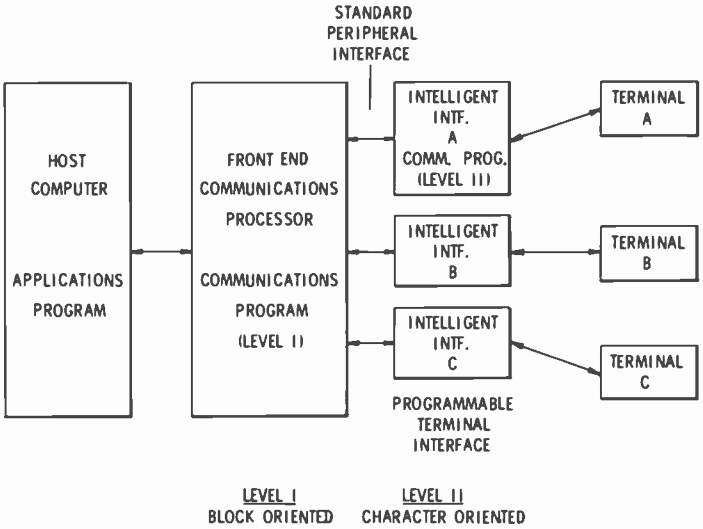


Fig. 4
Front-end communications processor provides a connection between remote terminals and a central processor.

protocols and control procedures. In the past, this flexibility has been provided by a family of line interface units and custom software. Recently, however, microprocessors have been incorporated in intelligent line interfaces (Fig. 5), which are able to accommodate different terminals and protocols in software. Thus, only a single hardware module is needed. In addition, line interface units can ease the burden on the front-end processor.

For example, the microprocessor can easily perform the character-by-character processing on its dedicated line. The front-end processor then performs only block processing instead of the character-by-character processing for the aggregate of all lines. Thus, processor throughput requirements are substantially reduced. The recently announced DMC-11 network link from Digital Equipment Corporation is an example of such a device. It performs all of the functions of DEC's DDCMP line protocol including message assembly, header processing, and error checking. This programmable line interface approach can be extended to microprocessor-based format translators that convert messages from diverse input devices into a standard format for processing.

Multiple-processor structures have several advantages.

The multiprocessor network is another distributed architecture in which the microprocessor is finding application. Such a structure comprises several processors each performing a part of the total task. For example, several new terminals have come on the market recently, using two microprocessors, one for editing functions and one for communications functions.

A good example of the benefits of multiprocessor configurations is the Model 6000 Intelligent Network Processor from Codex. This device, built around an Intel 3000 controlling up to eight Motorola 6800s, performs high-level network management functions such as automatic channel assignments, error checking, and system diagnostics. It can dynamically reconfigure a network to bypass faulty or busy links. The system is reputed to have the processing power of two medium-sized minicomputers.

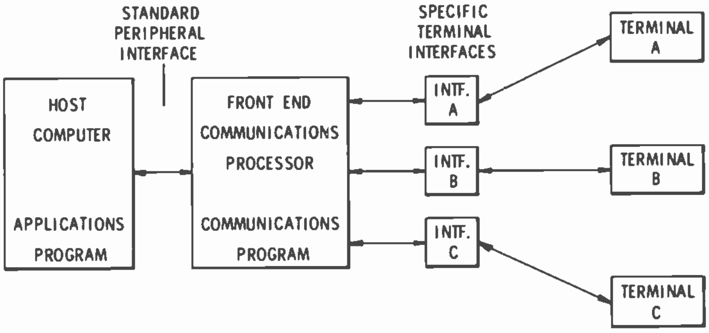


Fig. 5
Line-interface units can perform character-by-character processing (level II), freeing the front-end processor for block processing (level I).

Multiple-processor structures have several other advantages. They provide a high degree of modularity since functions can be added independently. They increase flexibility because it is easier to change individual functions once they are isolated. It is also possible that, in the future, different microprocessors will be optimized for different functions (e.g., bit manipulation for communications, or BCD arithmetic for calculations) which will make the multiprocessor approach even more attractive.

Conclusion

Only with extremely cost-effective equipment can the telecommunications industry sustain, within the limit of available capital, the phenomenal growth rate that is being dictated by customer demand. Currently available microprocessor devices have already proved to be flexible and economical systems building blocks. Microprocessors are already being widely used to replace random logic, as lower cost alternatives to minicomputers, and to provide entirely new functions and devices.

In spite of the large number of microprocessor applications that already exist in telecommunications, what we have seen up to now is only the beginning of the role the microprocessor will eventually play. As devices that are cheaper and more powerful emerge, entirely new areas of application will be opened up to programmable designs. For example, one of the next steps in the evolution of LSI will be the microcomputer on a chip: CPU, RAM, ROM, and I/O all in a single IC. Simple devices like this are already available, and newer, more-sophisticated ones have been announced. The potential of such devices for all areas of electronics is truly astounding.

References

1. Garen, E.R.; and Lazar, L.; "Microprocessors in telecommunications," *Telecommunications*, (Apr 1976), pp. 43-48.
2. Kaye, D.N ; "Microprocessors help to communicate by voice and bit stream," *Electronic Design* (Apr 12, 1976), pp. 58-61.

Microprocessors in manufacturing and control

C.C. Wang
C.T. Wu
P.M. Russo
A. Abramovich
A.R. Marcantonio

The dedicated microprocessor may foster a "second industrial revolution." Three reasons for this are the standard interface bus, EPROM software development, and integer-arithmetic computation.

As computers become smaller and less costly, they no longer play the role of superstars waited on by rooms full of peripherals. On the contrary, machines and equipment are becoming centers of attention catered to by microprocessors. This decentralization of computing power has been a natural consequence of cost drops in semiconductor components, coupled with rising prices for mechanical subsystems.

In industrial applications, microprocessors hold the promise for what some enthusiasts call "the second industrial revolution." Intelligent machinery with plenty of muscle and brain power should be the rule, rather than the exception, in the future. Instead of being limited to the classical role of computations on abstract quantities, microcomputers are at home with real attributes such as voltages, temperatures, pressures, etc. In this context, truly innovative uses of computers are just beginning to be explored.

This paper reviews some of the applications of microprocessors in this area and also describes some of the considerations associated with the design of such applications.

Computer-aided manufacturing

In the electronics industry, the pressures of increasing labor costs and the need for better quality control pushed the need for computer aids to manufacturing (CAM) to the surface. Microprocessors are making impressive gains in this field because they can provide solutions to long-existing problems in the factory. Although automation, up to now, has achieved its biggest successes in heavy industries such as steelmaking, oil-refining and chemical-processing, all aspects of manufacturing are moving toward more automation.

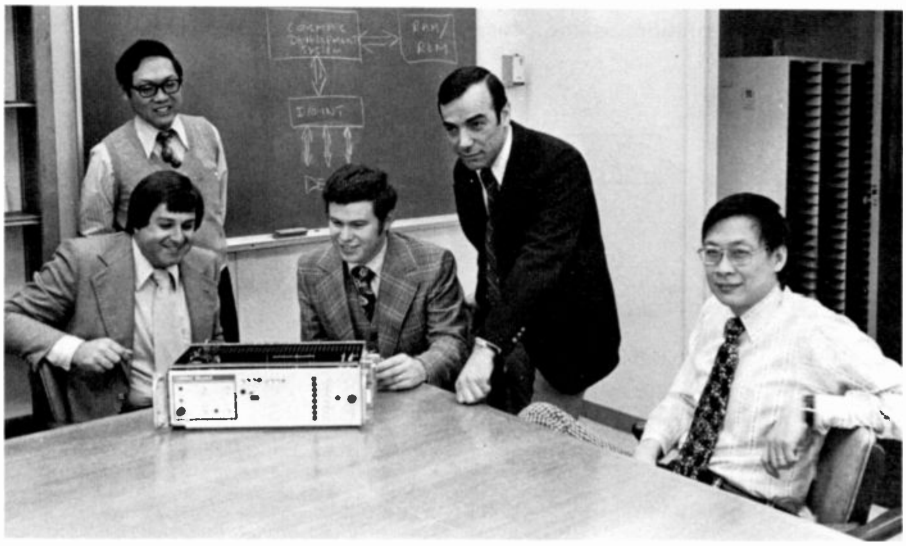
The manufacturing process can be roughly divided into five phases, as far as electronics use is concerned. They are:

- design
- documentation
- fabrication and assembly
- testing
- stocking and distribution

Reprint RE-22-5-7
Final manuscript received December 22, 1976.

The biographies of **Paul Russo** and **Angelo Marcantonio** appear with their other papers in this issue.

Authors Wang, Russo, Abramovich, Marcantonio, and Wu.



Abe Abramovich joined RCA in 1976, and is currently involved in exploring microprocessor applications. He is also attending the Graduate School of Engineering and Applied Science at Columbia University.

Contact him at:
Systems Research Laboratory
RCA Laboratories
Princeton, N.J.
Ext. 2750

Chih-chung Wang was a senior scientist with B.F. Goodrich Co. before he joined RCA Laboratories in 1973. Dr. Wang has been working in computer-aided design, software development for integrated-circuit testers, and, more recently, microprocessors and microprocessor applications.

Contact him at:
Systems Research Laboratory
RCA Laboratories
Princeton, N.J.
Ext. 3325

Chin Tao Wu has worked on various aspects of computer systems design since joining RCA in 1965. He has been interested in microprocessors since 1971 and was active in the early support work for RCA's COSMAC microprocessor.

Contact him at:
LSI System Design
Solid State Technology Center
Somerville, N.J.
Ext. 6061

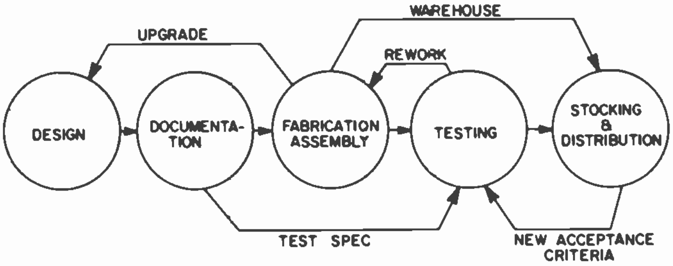


Fig. 1
Computer-aided manufacturing can be beneficial at any point in the manufacturing process. Large main-frames and minicomputers are already working extensively in design, documentation, and stocking and distribution; microcomputers are becoming more evident in the fabrication and testing areas.

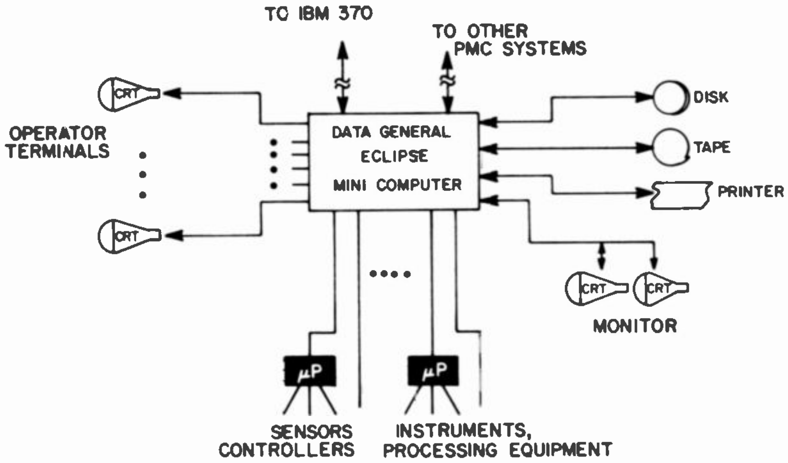


Fig. 2
"Local intelligence"—microprocessor controllers with local-loop capabilities—and a central minicomputer make up the process monitoring and control system.

Fig. 1 illustrates the process flow in such a manufacturing environment. Computers have been integrated into the different phases of this scheme with varying degrees of success. For design, documentation, and stocking and distribution, large main-frames and minicomputers are being used most extensively. In the area of testing, minicomputer CPUs (Central Processing Units) often control the switch-sequencing for high-speed testers. The microprocessor is, of course, a more cost-effective solution in low-speed testing.

In fabrication and assembly, tasks are often tedious, repetitive, and demanding of operator concentration. Microprocessors taking over these burdens will have the greatest impact on the eventual automation of this phase. Robots equipped with "eyesight" are already sitting on some of our assembly lines.

Wafer fabrication in the semiconductor industry is an example showing how better monitoring and control of processing steps can definitely pay off. In wafer-fab, many cycles of masking, etching and diffusion are required. Any deviation in the sequence means that subsequent steps are completely wasted. The penalties in lost time and poor yield are severe.

Precise process-monitoring can prevent a bad lot from propagating through a line. More accurate control, on the other hand, can minimize the occurrence of these lots. Both functions are within the realm of microprocessors.

Microprocessors are now being used in process monitoring and control for semiconductor manufacturing.

In the past, various industries have successfully applied minicomputers in controlling product flow. In the RCA Solid State Division, a system for collecting product flow data called OPEN (On-line Production Entry Network) has been implemented and is operating in several product lines. The OPEN system, however, collects logistical data (i.e., where products are at any time) rather than the process information giving rise to these products. Recognizing the need for collecting critical process parameters themselves, an ambitious process monitoring and control program (PMC) has begun.

The PMC system (Fig. 2) is based on a Data General Eclipse minicomputer and a network of CRT terminals similar in configuration to the OPEN system. Intelligent instruments and controllers with local-loop capabilities perform the measurement and control functions in order to minimize the effect of host-system crashes. Microprocesors form the basis for local intelligence in this level of the distributed system.

Microprocessors are the brains of the intelligent instruments that measure process variables and communicate them to a central data base, where the information is sorted and analyzed for statistical control purposes. A COSMAC-based capacitance voltage monitor[2] is an example of such a satellite instrument that, in addition to communicating the information, frees the operator from cumbersome calculations.

While process-monitoring provides the necessary feedback for corrective actions where man is in the loop, automated feedback control of complex processes is also important. Semiconductor-equipment vendors have designed closed-loop control into machines such as epi-reactors and thin-film depositors by using microprocessors. When linked with a host computer as components in a distributed system (such as PMC), these equipments acquire the additional flexibility of changing their control functions on request. With the rapid pace of technological obsolescence in the semiconductor industry, this capability is an important asset, since the same equipment can then perform many variations of a basic process.

Automatic testing had been confined to complex systems, but microprocessors are making it economical for subsystems and even components.

Automatic Test Systems (ATS), which can be used to automatically identify component, subsystem and system faults, are becoming increasingly important from the view of reliability and maintainability. Reliability is improved with ATS since marginal components and subsystems are

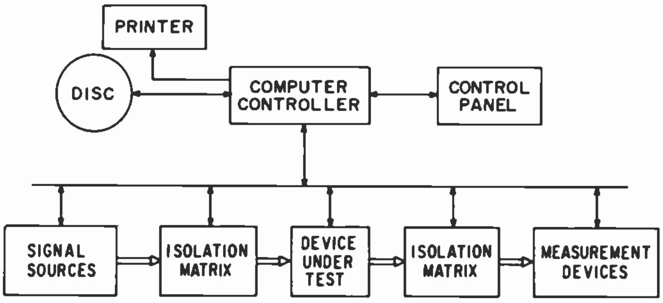


Fig. 3
Typical automatic tester includes signal package, measurement package, and a processor controller. I/O, data-storage, and operator-control subystems complete the unit.

replaced prior to equipment shipment, resulting in a lower probability of field failure. Ease of maintenance is also improved since system-level tests can identify the probable subsystem failures, and so give rapid repair turnaround and hence reduced downtime.

The principal subsystems of an automatic tester are a signal package, a measurement package, and a processor acting as system controller. Additionally, some operator control is needed along with some output devices (displays, meters, printers) to identify the test results. Finally, a disc or other mass-storage system may be desirable for storing data for later statistical analysis. Fig. 3 presents a block diagram of a typical test system where the subsystem under test is isolated from the signal sources and measurement devices. These can be relays, solid-state switches, or simple resistors, as required. Note that the system of Fig. 3 is rather complete—not every test system will contain all of its indicated subsystems.

ATS has historically been used in large military applications and in automatic testing of complex parts, where expensive test fixtures can be economically justified. The advent of microprocessors has opened many avenues for low-cost dedicated testers, which means that industries can now economically justify automated and more complete testing of low-cost/low-complexity subsystems right on down to the component level.

Current economic factors are tending to accelerate the swing toward more fully automated testers. Ever-increasing labor costs, decreasing hardware costs, and demands for constantly-improved product reliability and safety all dictate increased automation. Additionally, it is becoming increasingly desirable to catch failures early in the assembly process, since the value added (cost to troubleshoot and repair) goes up exponentially with the number of components in the subsystem under test.[3]

Many types of testing are possible. These include functional, dc, and ac (dynamic) tests. Functional tests establish that the subsystem under test is performing a desired digital or analog function. For example, a typical digital functional test would be to determine if a logic function satisfies a specified truth table, and an analog functional test would be to see if an amplifier meets specified linearity limits over a prescribed input-signal range. DC tests usually imply static measurements of component values, PC board continuity, amplifier gain, saturation voltages, etc., and ascertain whether these satisfy a given set of specifications. Dynamic tests establish relevant time-domain parameters such as system delay, overshoot, etc., and may also be used to measure reactive components.

The test methodology for a system defines the broad strategy of what will be tested and how. Designers must give the methodology considerable thought, for no matter how well the actual tester is performing, it will be of little use if it fails to test the critical components or critical functions.

Once a test method has been chosen and the test system developed, the test specifications must be defined. This step is crucial and very complex since it involves subtle trade-offs between performance/reliability objectives and system cost. Too-rigid test limits will increase system cost (more rejects), and too-liberal test limits may lower the average system performance and, perhaps, reduce reliability.

From this we conclude that both test methodology and test specifications are strongly influenced by economic and marketing factors, which will dictate the trade-offs of optimal cost per test vs. system performance and reliability.

If you are looking for more information on automated testing, Refs. 3 and 4 present excellent overviews of its desirability and economic benefits. Ref. 5 describes a host of testers having microprocessors as their controllers, and Ref. 6 describes a COSMAC-based tester-oriented development system described in depth elsewhere in this issue.

Dedicated controllers are replacing mainframe computers.

Before the use of microprocessors became widespread, mini or large computers were the hearts of complex controllers. Because of this centralization, controllers were economically practical only for large and complex systems. Furthermore, these early systems had all the attendant disadvantages of centralized control—large cabling costs, decreased reliability, and less flexibility. Smaller systems were designed with relay logic, sequencers, or, perhaps, analog feedback control.

With the advent of low-cost microprocessors, it became possible to decentralize the control function and use intelligent controllers dedicated to specific tasks. Many advantages result from local-loop and distributed control, including lower cabling costs, reduced noise pickup, improved overall reliability (if one small system fails, the remainder keep operating), simpler maintenance, and more flexibility. Several brief examples of dedicated microprocessor-based controllers follow. Refs. 7-11 discuss the benefits and applications of microprocessors to industrial control in more depth.

One example is in using a microprocessor to control the speed of a hysteresis synchronous motor to a high degree of precision. These motors maintain accurate average speeds, but their instantaneous speed may vary considerably. A microprocessor may be used to monitor the motor speed (say by counting crystal-controlled clock pulses per revolution) and then issue signals to move the driving frequency in the right direction, Fig. 4. With this technique, an all-digital motor-speed-control system is capable of speed accuracy of one part in 10^6.

A second example, microprocessor-based automotive spark-timing control, has the

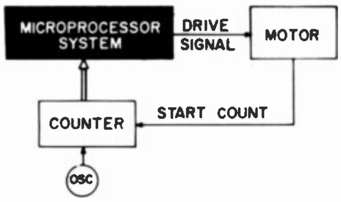


Fig. 4
Motor controller is an example of dedicated control. All-digital system is capable of one-part-in-a-million speed accuracy.

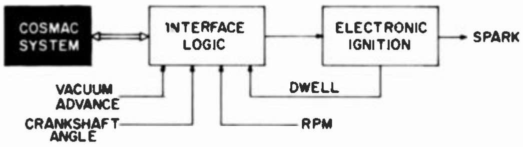


Fig. 5
Dedicated COSMAC microcomputer controls automotive spark timing more accurately than the mechanical system it replaces.

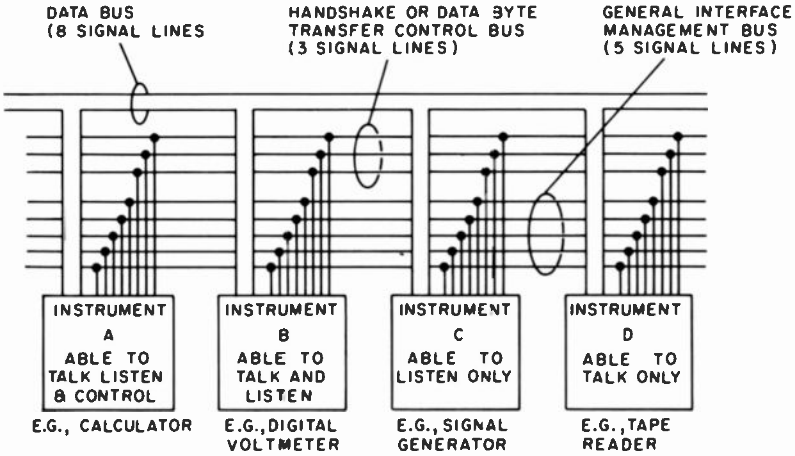

DATA BUS (8 SIGNAL LINES
HANDSHAKE OR DATA BYTE TRANSFER CONTROL BUS (3 SIGNAL LINES)
GENERAL INTERFACE MANAGEMENT BUS (5 SIGNAL LINES)
INSTRUMENT A ABLE TO TALK LISTEN & CONTROL
E.G., CALCULATOR
INSTRUMENT B ABLE TO TALK AND LISTEN
E.G., DIGITAL VOLTMETER
INSTRUMENT C ABLE TO LISTEN ONLY
E.G., SIGNAL GENERATOR
INSTRUMENT D ABLE TO TALK ONLY
E.G., TAPE READER

Fig. 6
Standard interface, the IEEE 488 bus, allows instruments to talk to and control each other. Sixteen signal lines are divided into groups for data I/O, "handshaking" for data transfer between devices, and controlling.

motivation of improved engine performance, fuel economy, and exhaust emission. The speed of combustion of the air fuel mixture is a strong function of engine operating conditions (rpm, load, etc.). Hence, it is important to have variable spark timing since it initiates the combustion process.

A.D. Robbi and J.W. Tuska have developed such a COSMAC-based timing system.[12] The system (Fig. 5) senses the vacuum advance, crankshaft angle, rpm, and dwell, and then triggers a standard electronic ignition subsystem. The CPU adaptively computes, for each spark, the desired firing angle (spark timing).

Other microprocessor control applications abound. Microprocessors are currently being designed into communications controllers (Harris 8400 multiplexer, Codex 6030 intelligent network processor), numerical-control machine tools, tracking systems, and instruments (HP 1722A scope, Systrom-Donner 7115 digital multimeter, DANA 9000 counter/timer). It is apparent that we see only the tip of the iceberg!

Design considerations

The design of a microprocessor-based application consists of selecting the central processing unit (CPU), memory, and I/O devices, and then developing customized I/O hardware, I/O interface, and software. Instead of presenting an extensive overview on the subject, only a few selected topics are discussed in detail here—specifically, the IEEE 488 Interface Bus, EPROMs and software development, and integer arithmetic. Ref. 13 gives an overview of system design.

The I/O interface design comprises a major portion of the hardware development for most of the microprocessor-based applications. Using the IEEE 488 Standard Interface Bus cuts back on this development work, since a wide variety of standard instruments can be interconnected quickly and easily with it.

An EPROM (Erasable Programmable Read Only Memory) is a very flexible medium for program storage. It is very popular in prototyping and low-volume applications, and it provides an ideal medium to integrate software into the application hardware.

Since most microprocessors perform multiplication and division with time-consuming software subroutines, microprocessor applications have been limited in the control area. Integer arithmetic is a technique that avoids multiplication and large intermediate results in control algorithms; it is especially suitable for microprocessor applications.

The standard interface bus lets instruments talk to, listen to, and control each other.

The IEEE 488 Interface Bus provides a versatile and effective communication link[14,15] for exchanging digital information in an unambiguous manner. It can accommodate a wide range of devices with similar protocols and is being proposed as an international standard—the number of instruments supported by the bus grows almost daily.

The interface bus consists of a group of 16 signal lines (Fig. 6) interconnecting a number of devices. Any device connected to the bus may either be idle or functioning as a talker, listener, or controller. As a talker, a device can send digital information across the bus to any device acting as a listener. Conversely, listeners can receive digital information across the bus from talkers. A controller can address other devices as talker or listener, act on service requests, and issue commands. A system controller can perform all the controller functions, remotely control other devices connected to the bus, and inititalize the bus. At any given time, only one device can be functioning as the system controller.

The 16 signal lines that comprise the bus may be classified into three separate sets—the data bus, the handshake group, and the bus-management lines. The data bus consists of 8 bidirectional I/O data lines and the handshake group is made up of 3 lines that help synchronize data transfers between devices. The controller uses the 5 bus-management lines to process service requests, sense and control the end-of-record line for data transfer, and set the attention line high or low to indicate if the data bus is in the address or command mode. The system controller uses the bus-management lines to remotely control other instruments on the bus and send the interface "clear" for a system reset.

If an instrument is being designed to interface to the IEEE 488 standard bus, the instrument designer must select which functions to include, as well as the capabilities within the chosen functions. In the event that an interface is being developed for a controller, and the controller has software-processing capabilities, it's up to the designer to decide what parts of the interface should be done in hardware and in software.

EPROMs simplify software development.

In developing microprocessor-based applications, the software-storage medium is an important subsystem. For sophisticated systems, we can depend on permanent program storage with paper tape, cassettes, floppy disks or even the storage media of a large computer. But, obviously, these methods cannot be used with most of the intended microprocessor applications. For most applications, ROMs (read-only memories) and EPROMs (erasable programmable read-only memories) provide attractive alternatives for nonvolatile program storage.*

ROMs are mask-programmable in the sense that the software is built into the IC chip. Because a mask must be designed to customize the chip for each piece of software, ROMs are only economical for large-volume applications. ROMs have the disadvantage that once manufactured, they cannot be modified. Also, it usually takes 6-8 weeks for delivery of a new ROM; this delay is intolerable in the product-developing and prototyping stage.

These problems can be avoided with EPROMs, which are based on the floating-gate MOS structure[16]—they store information in the memory cell as electrons trapped in the floating gate. The programming takes only a few minutes. Because EPROMs can also be erased and reprogrammed for program modification, they are an ideal storage medium in the product-developing and prototyping stage. They also have the economic edge over ROMs for low-volume production applications.

To appreciate the advantages offered by the EPROM, we have to understand the approach to software development for microprocessor-based applications. It consists of some or all of the following steps.

1) Software developed and debugged on a time-sharing system.
2) Software developed, tested and debugged on a microprocessor-based development system.
3) Software tested on the prototype.
4) Software integrated into the final product.

EPROMs can be used as the program storage medium in each of these steps except for the first, but the most important applications of EPROMs are in the last two steps. In the time-sharing and microprocessor-based approaches, EPROMs perform the same function—transferring software to the prototype and the final product.

Most of the microprocessor-based development systems provide certain basic system utilities, e.g. examining memory, modifying memory, setting breakpoint, etc., in order to generate, edit, and debug the software. These primitive tools, coupled with an EPROM's program storage ability, are adequate for certain application developments. At the beginning of each work session, the contents of the EPROM are loaded into the RAM of the development system for editing and testing, and at the end of the session, the contents of the RAM are dumped into the EPROM for storage. After the program is debugged, the EPROM can be used directly in the prototype or the final product.

Since this approach deals with everything at the machine-code level, its capability for system development is limited. It does, however, provide a simple method for upgrading or customizing the software for a microprocessor-based product that uses an EPROM for program storage.

Integer arithmetic solves control problems at high speed.

Integer arithmetic is a numerical technique for computing solutions to initial-value problems for functions $f(x,y,\ldots) = 0$. Here, the term "initial-value problem" signifies that rather than solving the equation for an exact solution, an arbitrary starting point is chosen at a set of coordinates $x,y,\ldots$ From that starting point, the integer-arithmetic technique converges onto the desired contour, and then follows it.

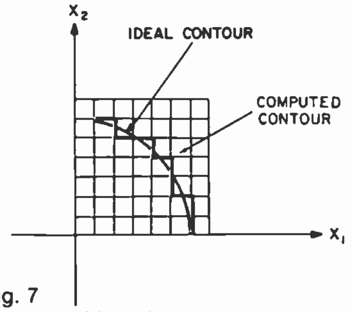


Fig. 7
Integer arithmetic solves equations to the closest grid point (which can be set as accurately as needed for a given control application).

This technique is designed for digital computation, and it is especially suited for microprocessors because it avoids time-consuming multiplication routines and so does not require the storage of large products. In solving initial-value problems of the type $f(x,y) = 0$, a new variable, F, is introduced, and the equation is set to $f(x,y) = F$. The problem is then solved on a set of Cartesian coordinates only to the closest grid point (Fig. 7). Since most digital control applications require a finite accuracy, the grid is chosen so that its smallest increment is consistent with the system's tolerable error. The resulting error of the integer-arithmetic solution is predictable and noncumulative because F is evaluated at incremental integer points and expressed as a difference for each variable, holding the other variables constant. In a two-dimensional case, the incremental values are

$$F_{\pm x} = \pm 2x + 1$$
$$F_{\pm y} = \pm 2y + 1$$

The technique can solve any initial-value problem, linear or non-linear, of any order.

Fig. 8 shows one of the several computing algorithms that have been formulated for integer arithmetic. This one is designed to examine the magnitude or sign of $F|_{x,y}$ and then decide to make either a positive or native change in x or y in order to keep the F variable within the allowable error limits.

An Intel 8080A microprocessor has used this algorithm to solve for the set of points on a circle, given the initial x,y coordinates (Fig. 9); the introduction of a damping factor yielded the spiral contour. The program consisted of about 150 in-

*For certain applications, a battery-backed CMOS RAM, e.g. RCA CDP1821 (1024×1) or CDP1822 (256×4), may also be very desirable.

structions, which yielded a new solution to the equation once every 600 microseconds. It would not be possible to execute the solution to this problem at comparable speed using the classical method.

This algorithm may be applied with equal ease to solve contours for other second-order equations, and the integer-arithmetic technique may be applied to control applications that require high-speed real-time solutions.

Conclusions

As the engineering community's familiarity with microprocessors increases, applications in manufacturing and control will undoubtedly become wider and deeper. Mass-produced microprocessors provide cheap and flexible substitutes for relay logic, sequencers, or perhaps analog feedback control. They are being designed into more and more machines, equipment, and instruments, providing local programmable intelligence and control. The spread of these intelligent machines over the manufacturing plant distributes the processing power (which used to be centralized in large main-frames) and lays the foundation for the distributed processing system in the future.

References

1. Cassell, D.A.; *Introduction to Computer-Aided Manufacturing in Electronics* (Wiley Interscience; 1972).
2. Wu, C.T.; "CVM: a microprocessor-based intelligent terminal," *RCA Engineer*, this issue.
3. Lyman, J.; "Automatic circuit testers lead the way out of continuity maze," *Electronics*, Aug 1974, pp. 87-95
4. Eleccion, M.; "Automatic testing: quality raiser, dollar saver," *IEEE Spectrum*, Aug 1974, pp. 38-43.
5. Runyon, S.; "Microprocessors showing promise in test equipment, but haven't made it big yet," *Electronic Design*, Apr 26, 1974, pp. 90-95.
6. Marcantonio, A.R.; and Russo, P.M.; "A microcomputer for test and control applications," *RCA Engineer*, this issue.
7. Weissberger, A.J.; "Microprocessors simplify industrial control systems," *Electronic Design*, Oct 25, 1975.
8. Olson, K.; "The tractors of the computer industry—part 1," *Digital Design*, May 1975.
9. Olson, K.; "The tractors of the computer industry—part 2," *Digital Design*, Jun 1975.
10. Weissberger, A.J.; "Microprocessors expand industry applications of data acquisition," *Electronics*, Sep 5, 1974.
11. Marmala, A.D.; "Benefits of localized control with microcomputers," *Computer Design*, May 1975.
12. Robbi, A.D.; and Tuska, J.W.; "A microcomputer-controlled spark advance system," *RCA Engineer*, Vol. 22, No. 2 (Aug - Sep 1976).
13. Russo, P.M.; "Architectural trade-offs in the design of microcomputer systems," *RCA Engineer*, this issue.
14. IEEE, "IEEE standard digital interface for programmable instrumentation," *IEEE Std*. 488-1975.
15. Ricci, D.W.; and Nelson, G.E.; "Standard instrument interface simplifies systems design," *Electronics*, Nov. 14, 1974, pp. 95-106.
16. Frohman-Bentchkowsky, D.; "A fully decoded 2048-bit electronically programmable FAMOS read only memory," *IEEE Solid-State Circuits*, Vol. SC-6, (Oct 1971) pp. 301-306.
17. Gorman, J.E.; and Raamot, J.; "Integer arithmetic technique for digital control computer," *Computer Design*, Vol. 19, No. 7, (Jul 1970) pp. 51-57.

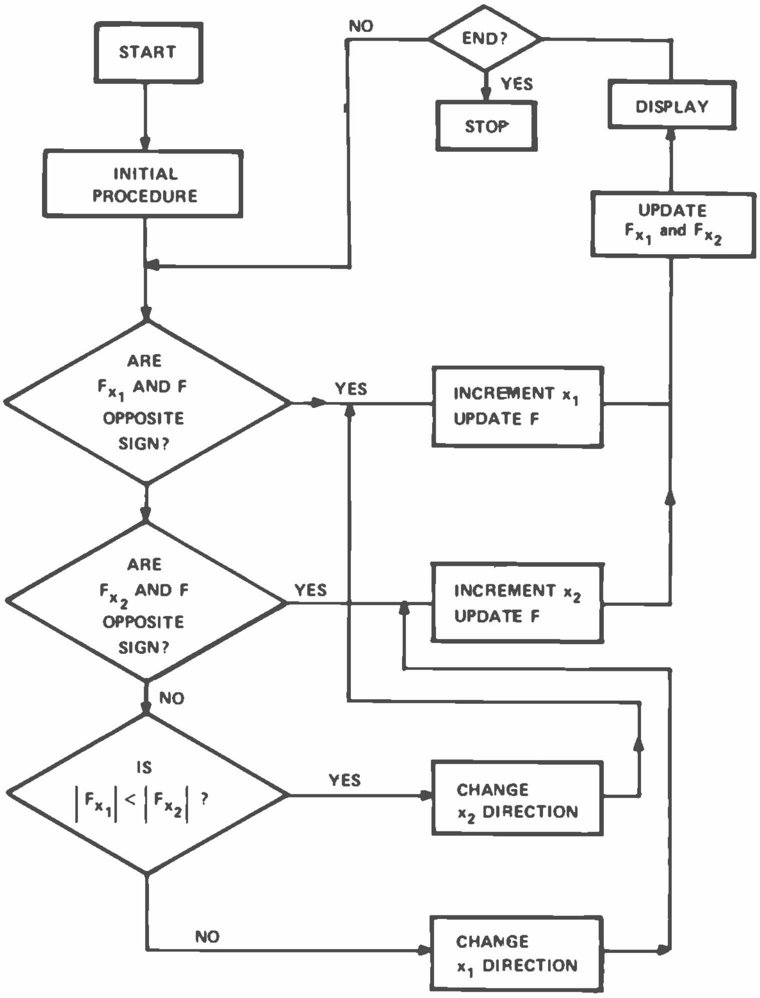


Fig. 8
Integer-arithmetic technique makes high-speed microprocessor solutions to equations possible by eliminating multiplication and storage problems. Algorithm shown here solves contours for second-order equations.

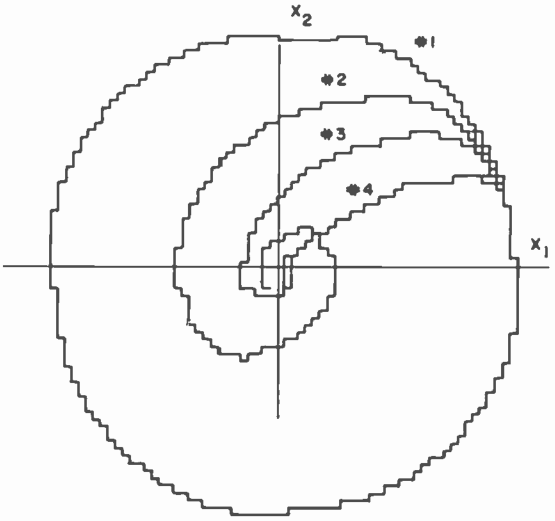


Fig. 9
Results of using the algorithm of Fig. 8 to follow the contours of a circle and spirals.

A microcomputer for test and control applications

A.R. Marcantonio
P.M. Russo

This versatile microcomputer system controls instruments as diverse as tv displays, programmable power supplies, timer/counters, and other microprocessors.

Low-cost microprocessors, memories, interface circuits, sensors, and transducers are now making it possible to decentralize test and control functions. With flexible, modular, economical, and expandable microprocessor-based hardware available, many test and control functions that can not economically justify a powerful minicomputer-based system may now be ready candidates for the low-cost microprocessor-based approach.

Advantages of microprocessor control

The primary advantage of computer-based testers and controllers is the ease with which they can be reprogrammed to satisfy a new test or control sequence using different parameter values. With modular hardware, the majority of functions can be provided by simply plugging in the required interfaces and modifying the software. The only custom circuitry required is used to condition the signals between the computer system and the device under test or control. This standardization yields tremendous benefits from the maintenance and backup-unit points of view; furthermore, it reduces the cost per tester, since the basic hardware may be reproduced many times over.

System organization

Fig. 1 details the components of a microcomputer test and control system designed at RCA Laboratories. The COSMAC Development System (CDS)[1], which is the center of the system, is equipped with 4K bytes of RAM and supports two-level I/O.[2]

In defining the device interface architectures for this system, the overall aim was to keep them both general and easy to use. No attempt was made to conserve I/O instructions, since two-level I/O makes the entire I/O interface available to each device. Finally, I/O instructions can easily be incorporated into a software interpreter aimed at a particular application.

The following paragraphs give more information on the interfaces listed in Fig. 1.

The analog-to-digital converter reads and converts up to 16 input ports.

The analog-to-digital interface uses the 0-10 V 12-bit Datel ADC-M12BIBI A/D converter with a 13-μs conversion time and a 10-megohm input impedance. It allows an analog voltage to be read from any one of sixteen input ports (two of which are tied to stable reference sources) and converted to a binary value that can then be read by the CPU. The interface architecture allows for compatibility between 8-, 12-, or 16-bit converters without altering the interface logic design or the I/O instructions. The same data format is used in both the D/A and A/D converters to allow for compatible data-handling software.

Two of the program-selectable input ports are dedicated to stable voltage references—5V and ground—to permit a certain amount of self-checking and data correction via suitable software techniques. An erroneous digital value for the 5-V reference indicates an error in gain, offset, or both, while an erroneous digital value for the ground reference indicates a positive offset error. The gain error is the difference in slope between the actual transfer function of the A/D converter and the ideal function. Offset error is the amount by which the transfer function of the A/D fails to pass through the origin, referred to the analog axis.

Multiple-device interrupts are resolved quickly by hardware priority and software vectoring.

The hardware[3] operates as follows: During each machine cycle, the status of 8 interrupt lines is sampled and latched; the output of the latches drives a priority encoder, which can be read by the CPU as a normal memory location. The distinguishing feature of this method is that it accomplishes a direct jump to the interrupt service code of highest priority by stuffing the output of the priority encoder into the low-order byte of the program counter. The interrupt priority-resolution hardware features are summarized below:

—The highest-priority interrupt is always serviced first.

—Any of the eight devices may be inhibited from interrupting the CPU.

—Interrupt-resolution hardware appears as a memory location, so no I/O instructions are required for access and it is always selected.

—A jump to a respective device's service code can be done with two instructions.

—A single address accommodates both the priority encoder on read and the interrupt-enable latches on write.

The programmable counter/timer interface makes it possible to control an existing 16-bit interval timer.

A single output instruction is used with bit patterns in M(R(X)), specifying commands to the interface. The controller, together with the interval timer,[a] sets the timer clock rates, counts external events, and signals the CPU at preset intervals. Features of the counter/timer control interface[4] are as follows:

—Rates from 131.3 μs to 16.804 ms in multiples of two.

—Repeated time-out pulses at the rates given above, or a single programmed-length time-out to a maximum of 18.35 minutes.

—Choice of setting external flag *and* interrupt at time-out, or just external flag.

Reprint RE-22-5-8
Final manuscript received July 19, 1976.

[a] Developed by C.T. Wu, RCA SSTC, Somerville, N.J.

— External-event counter input available for positive- or negative-going inputs, allowing counts to 65,536, independent of CPU processing.

— Buffered time-out output that can be used to signal an external device after a preset interval.

The digital-to-analog converter interface enables CDS users to convert digital data to an analog voltage and direct it to any one of eight program-selectable output ports.

The D/A converter interface uses the 0-10 V 12-bit Datel DAC-VR12B1B D/A converter with a 2-μs settling time. The data format is identical to the A/D converter's. Features of the D/A interface are listed below:

— Single fixed analog output port, which additionally can be switched to any one of eight program-selectable output ports, if desired.

Microcomputer interface summary

A/D converter with 12-bit accuracy, 0-10 V range, 10-megohm input impedance, 16 program-selectable input ports, 13-microsecond conversion time, and selectable stable voltage reference.

Interrupt priority resolution logic, providing vectored interrupt servicing within 2 instruction times.

Programmable timer/event counter that can count external events and generate single or repeated time-outs at programmed rate.

Interval timer with 16-bit resolution used in conjunction with the programmable timer/event counter in a countdown mode.

D/A converter with 12-bit accuracy, 0-10 V range, 15-mA output drive current, 8 program-selectable output ports, 2-microsecond settling time, and input storage register.

Programmable power supply with 10-bit accuracy and facility to zero output and control ac connection on command.

Dot-matrix interface to standard tv, providing display in 32X32, 16X64, or 32X64 dot format.

Buffered cable to allow CDS backplane expansion into an external card nest.

Software-controlled relay matrix expandable in 32 relay increments for interfacing to "bed of nails" for PC-board testing.

Signal multiplexer to permit switching and signal conditioning of eight relay-based inputs and eight relay-based outputs.

Interprocessor interface to facilitate multimicrocomputer systems.

Program-mode I/O logic and hardware (eight switches, pushbutton, prompt LED, and audio alarm) to enable data entry during program execution.

Status and control-panel logic and hardware, consisting of nine LEDs and eight pushbuttons for displaying status information and generating a unique three-bit code.

Standard 4½-digit autoranging multimeter for measuring ac/dc current, ac/dc voltage, and ohms.

Standard digital counter for frequency, period, event and time measurement.

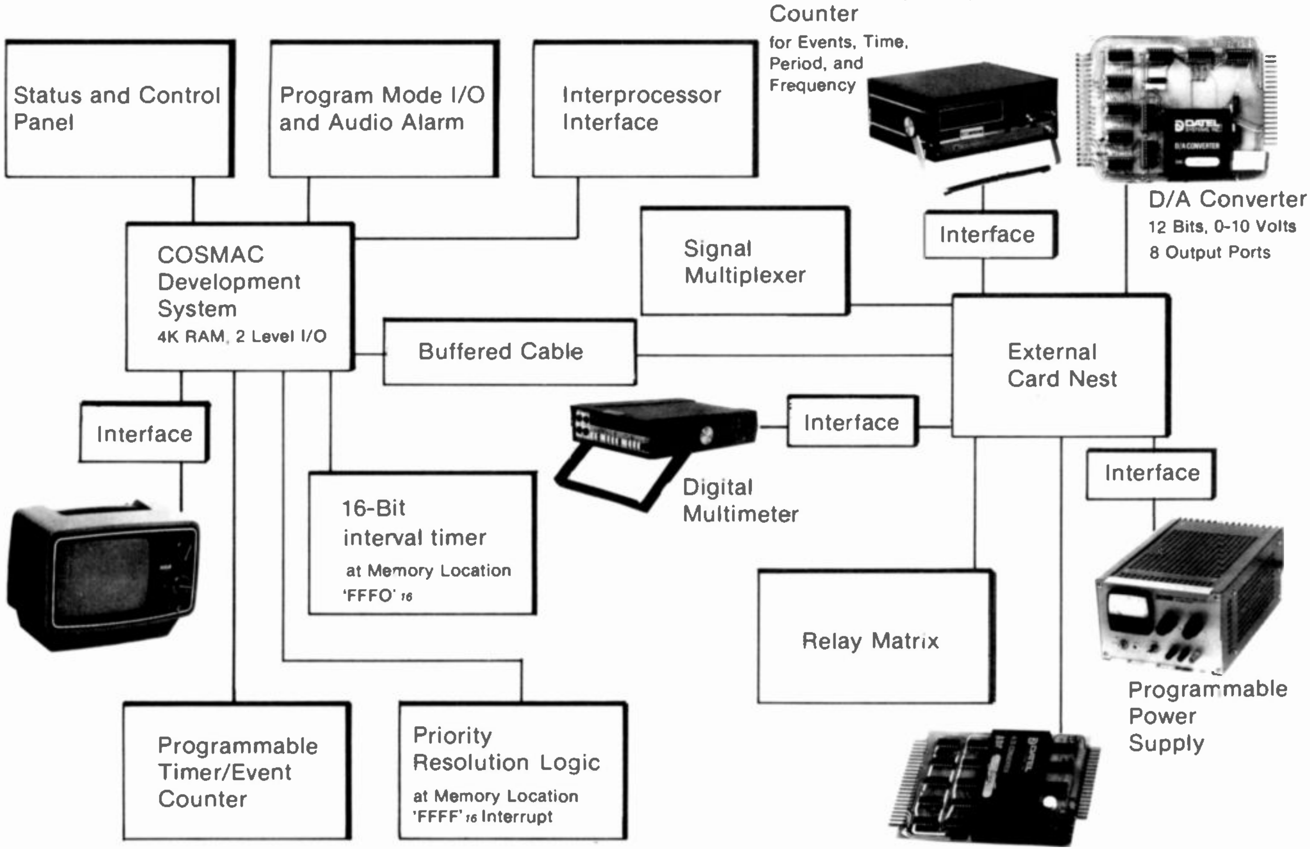


Fig. 1
Test and control system uses a standard COSMAC Microkit with two-level I/O, making the entire interface available to each of the controlled devices.

—Both hardware design and software instructions applicable to 8-, 12-, and 16-bit D/A converters.

—Input storage register that avoids output transitions due to partial field loading for 12- or 16-bit units.

Programmable power-supply interface controls a 10-bit digital-to-analog programmer.

Made by Sorenson, the programmer can, in turn, control Sorenson's complete line of programmable power supplies. A DCR 15-3B (150 V, 3 A at 300 ms) supply satisfies our high-voltage/high-current needs, but is relatively slow. To fill the need for a fast-response, lower-voltage supply, a QRD 60-1.5 (60 V, 1.5 A at 25 μs) was chosen.[b]

The interface provides, under software control, addressing logic for up to 4 DAP programmers, a *clear* signal, 10 data bits with an associated strobe, and a pair of signals used to drive the ouput to zero and/or disconnect the ac line from the power supply by means of an external relay. In addition, each power supply provides a flag that indicates when the output is within a 2.5% tolerance.

A tv interface provides a dot-matrix display for a standard television receiver.

Based on a custom RCA television I/O chip,[c] this interface can put the display in any one of three program-selectable modes—32×32, 16×64, and 32×64 dots, requiring 128, 128, and 256 bytes of memory, respectively. Each dot of the display represents a bit in the CDS memory and can be modified by program. Two I/O instructions are needed, one for selecting or enabling the interface and another for setting the mode of the display and turning it on/off.

In addition to producing an extremely flexible character and graphics disply, the tv interface has been very useful in testing and debugging. For example, this interface, with its dot-matrix format, can help identify faulty memory chips radidly when used with a corresponding test program. This is done by storing a bit pattern in the memory area under test, displaying the pattern, and then visually detecting and identifying incorrect bit fields.

A 64-wire buffered cable makes it possible to add an additional card nest to the COSMAC Development System.

Each signal goes through a logic driver at each end of the cable, thereby applying only a single load to the signal source and providing substantial drive at the signal destination. Additionally, certain lines usually connected in a "wired-or" configuration are supplied through transmission gates to provide tri-state capability.

The relay matrix can test up to 256 test points.

The relay matrix interface[d] requires a minimum of two relay matrix cards to complete a circuit under test. Each card contains an array of 32 relays, any one of which can be closed under software control. The interface is most useful in applications where complete isolation is required between the measuring and measured equipment (e.g., a "bed of nails"). Each card has four bank-select switches whose settings distinguish it from a maximum of sixteen possible cards. Summarized below are the features of the relay matrix interface:

—Expandable to 16 relay cards, facilitating exhaustive testing of up to 256 test points.

—Response to simple continuity/open test, directly testable under program control using external flag.

—Each relay card may be selected with or without energizing its relays.

—A single command is available to de-energize all relays, on all cards, simultaneously.

—A four-element DIP switch enables easy assignment of relay card selection numbers.

The signal multiplexer permits time-multiplexing of eight input-signal sources and eight output signals through a bank of relays.

This device is especially useful when used in conjunction with single input/single output devices such as the relay matrix. The board has space for conditioning any of the input and/or output signals required. Transmission gates are provided so that conditioned input/output signals can be connected directly to the flag lines. This may be advantageous in certain configurations, as it permits more rapid software determination of the signal state.

The inter-processor interface architecture permits interconnecting up to 256 COSMAC-based slave systems to one master.

In a master-slave configuration, all inter-processor communication goes through the master. No common memory is provided,

[b] The interface logic was designed by T.F. Lenihan, RCA Laboratories, Princeton, N.J.

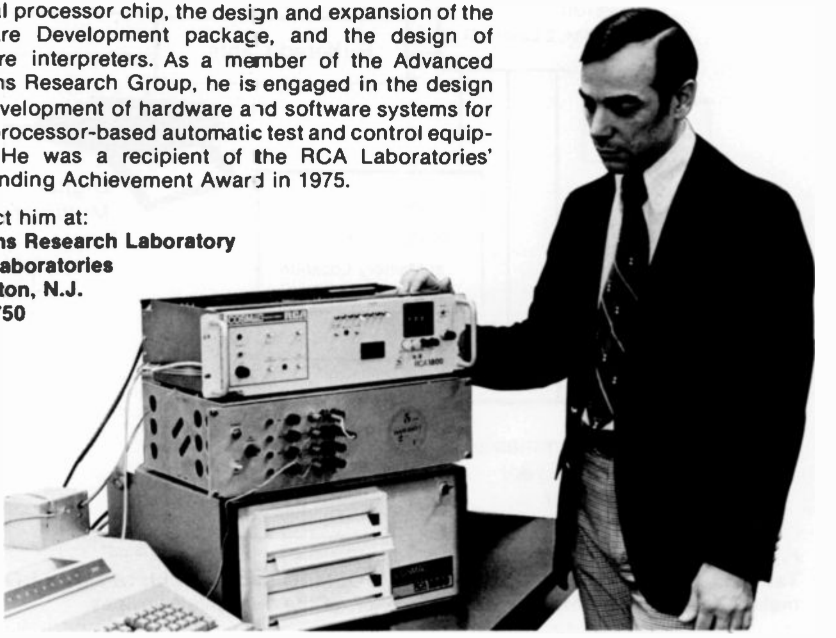
[c] This chip was designed by J.A. Weisbecker, RCA Laboratories, Princeton, N.J. Additional information on the tv display is presented in Ref. 6.

[d] Developed in cooperation with M. D. Lippman.

Paul Russo's photograph and biography appear with his other article in this issue.

Angelo Marcantonio contributed to the first microprocessor developments leading to the COSMAC and worked on the logic simulation and fault testing of the original processor chip, the design and expansion of the Software Development package, and the design of software interpreters. As a member of the Advanced Systems Research Group, he is engaged in the design and development of hardware and software systems for microprocessor-based automatic test and control equipment. He was a recipient of the RCA Laboratories' Outstanding Achievement Award in 1975.

Contact him at:
Systems Research Laboratory
RCA Laboratories
Princeton, N.J.
Ext. 2750



and each CPU will have enough RAM to handle its dedicated workload. Interprocessor communication will be via programmed-mode I/O or via DMA (for block transfers). The external flags are used for status, handshaking, and command encoding. Additional information is presented in Refs. 7 and 8.

The program-mode I/O hardware allows kit users to output a prompt signal, input a ready signal, and input 8 bits of information.

The programmable I/O hardware uses an LED, a pushbutton, and a bank of eight toggle switches in a manner quite similar to that used for the Microtutor. The input side of the CDS byte I/O card is assumed to be available for gating-in the state of the eight switches; the output side of the byte I/O card can be used to drive a two-digit LED hex display directly using the bit output lines.[5] There is also an audio alarm available, which can be turned on/off via software.

The status and control panel is a general-purpose interface.

This interface displays eight bits of status information and reads any one of eight pushbuttons, each with a unique three-bit code. The code could be used to index a table, perform a vectored branch, select an interface, turn a device on or off, or any other function requiring a simple one-of-eight choice.

The digital-multimeter interface allows a 4½-digit DMM with a digital output option to be used as an input device to the Development System.

Interface programming for the DMM, a Fluke 8600A, is accomplished with a single output instruction. A wide variety of options is available to the user, including a mode that, independent of CPU processing, automatically updates DMM measurement data. Listed below are the relevant features of the DMM interface:

— Complete status of DMM front-panel settings available to CPU, including seventeen bits of data, three bits indicating scale, one sign bit, and one overflow bit.

— Bank of input latches allows the DMM's output to be sampled and stored every machine cycle, completely invisible to the main program with appropriate choice of options.

— Autoranging setting of DMM enables parameter being read to be anywhere within the entire range of the 8600-A for all parameters except current.

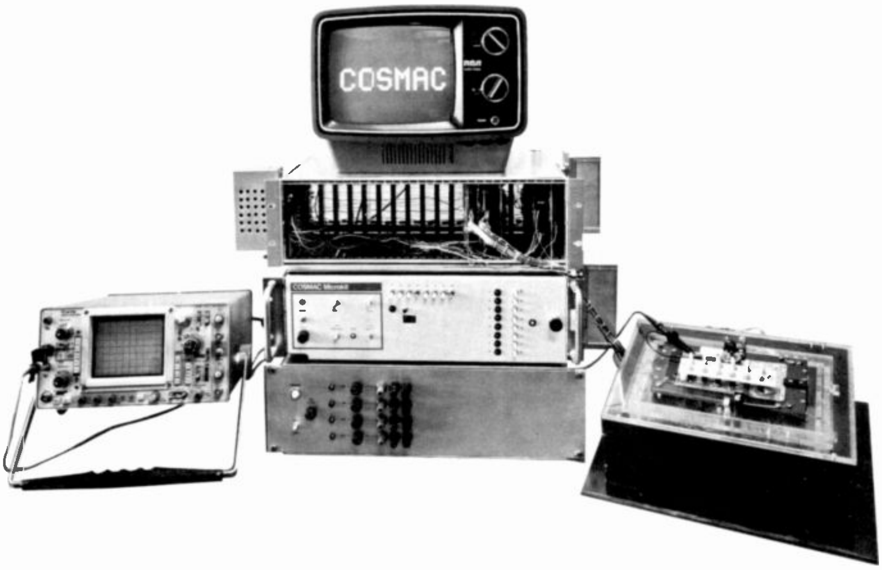


Fig. 2
Developmental hardware. Top to bottom: tv monitor, card nest, COSMAC Development System, and power supply. Scope is at left and "bed of nails" interface at right.

The counter interface, together with a Data Precision 5470 counter, enables the CDS to make frequency, period, event, and time measurements.

This interface[*] allows the CDS software to control functions of reset, run, and hold for a Data Precision 5470 counter. The specific mode of measurement is selected manually.

Ongoing work is aimed at developing additional suitable automation-oriented interfaces for the CDS.

Of particular interest is the development of an IEEE Standard Instrumentation Bus Interface for the CDS system, which would enable it to communicate with a host of standard instruments and hence permit system designers to build up various test/control systems rapidly.

Additional interfaces under development include new A/D, D/A, and printer interfaces, made possible by the rapid advances in those technologies.

Specific applications currently being investigated include closed-loop feedback-control systems and automatic testing of printed-circuit boards.

Conclusions

The basic advantages of microprocessor-based test and control systems over custom-made hard-wired fixtures is the ease with which systems can be reconfigured at the hardware level (modular) and at the function level (software changes). Since the bulk of the hardware will be identical for each test/control system, maintenance is easier, development costs are reduced, and changes are readily made. Indeed, this flexibility is the most valuable aspect of the use of microprocessors in test and control applications.

Fig. 2 presents a close-up view of the developmental hardware.

Acknowledgment

The authors gratefully acknowledge the efforts of T.F. Lenihan for his excellent construction of the prototype hardware.

References

1. *COSMAC Development System Operator's Manual*, MPM-103, RCA Solid State Division, Somerville, N.J.

2. "Adding two-level I O interface capability to RCA COSMAC development systems," Application Note ICAN-6486, (May 1976) RCA Solid State Division, Somerville, N.J..

3. Marcantonio, A.R.; "Interrupt priority resolution circuit for use with RCA COSMAC development system," Application Note ICAN-6485, (Mar 1976) RCA Solid State Division, Somerville, N.J.

4. Marcantonio, A.R.; "Programmable interval timer and counter for use with RCA COSMAC development systems," Application Note ICAN-6482 (Mar 1976) RCA Solid State Division, Somerville, N.J..

5. Lippman, M.D.; Marcantonio, A.R.; and Russo, P.M.; "Hexadecimal display for use with RCA COSMAC development system." Application Note ICAN-6487, (Mar 1976) RCA Solid State Division, Somerville, N.J..

6. Weisbecker, J.A.; "A practical, low-cost, home school microprocessor system," *IEEE Computer*, Aug 1974.

7. Russo, P.M.; "An interface for multi-microcomputer systems," *Proc. Fall* '76 *COMPCON*, Washington, D.C., Sep 1976.

8. Russo, P.M. "Architectural trade-offs in the design of microcomputer systems," *RCA Engineer*, this issue.

[*] Developed by C.C. Wang, RCA Laboratories, Princeton, N.J.

Studio II—using COSMAC in a home entertainment product

Studio II—RCA's new home tv programmer—uses the COSMAC microprocessor, along with resident and plug-in memory, to provide a versatile game and educational format.

J.D. Callaghan| W.M. Stobbe| W.M. Stonaker

RCA recently entered the consumer video game market with a tv programmer called "Studio II," which is currently being marketed on a selected basis, with national distribution scheduled for mid-year.

RCA's new single-chip version of the COSMAC microprocessor, the CDP1802, provides central computer control for the various game, educational, and other entertainment programs offered with Studio II. Thus, a very high level of program "sophistication" can be achieved, limited only by the solid-state memory capacity. The product remains flexible and updatable through the simple expedient of plug-in program cards.

The Studio II logic is based on the FRED system—developed by Joe Weisbecker of RCA Laboratories. Weisbecker and P.K. Baltzer, also from RCA Laboratories, developed the initial software programs for Studio II. In January 1976, the Distributor and Special Products Division engineers were given responsibility for product design of the modified FRED system.

The result was Studio II—a game that connects to the vhf antenna terminals of a standard television receiver. Two ten-digit keyboards, located on the main housing, let the players control the display and select the "game" to be played.

Studio II consists of three major pieces:

Main-control console, which contains the keyboards and electronics for selecting and processing the programs.

Selector-switch box, which controls power to the control console and connects either the tv antenna or the rf output cable (from Studio II) to the receiver antenna terminals.

Power supply, which plugs into the wall ac outlet and supplies 9 Vdc to the

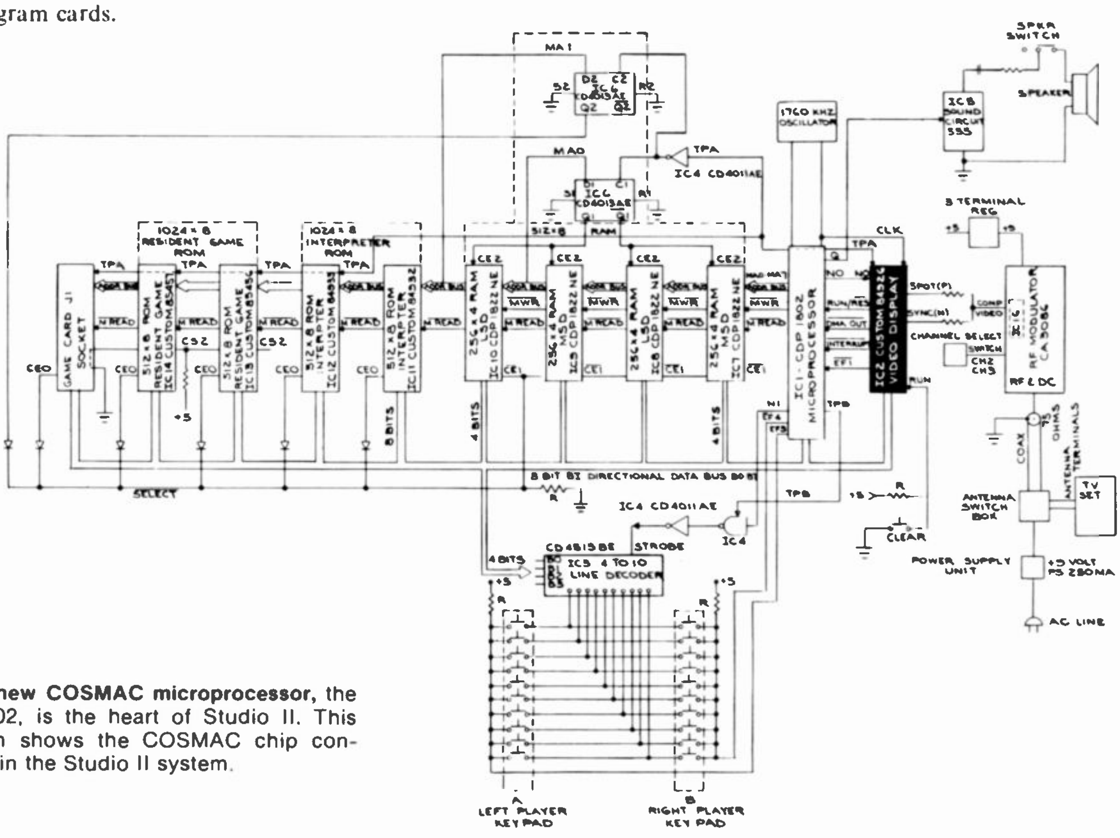


Fig. 1
RCA's new COSMAC microprocessor, the CDP1802, is the heart of Studio II. This diagram shows the COSMAC chip connected in the Studio II system.

control console via the selector-switch box and rf output cable.

Supplementing the programs already available in the control console's fixed read-only memory (ROM), several plug-in cartridges have been programmed with additional entertainment formats. The games that have been developed to date are described in Table I. These games were devised by RCA Laboratories; new games are being developed under the direction of D&SPD product planning.

System operation

The system (Fig. 1) can be divided into eight subfunctions:

1) Microprocessor,

2) Memory system consisting of read-only memories (ROMs) and random-access memories (RAMs),

3) Player/keyboard interface,

4) Video display generator,

5) Sound circuit,

6) RF oscillator/modulator section,

7) Selector-switch box, and

8) DC power supply.

The heart of Studio II is RCA's new COSMAC microprocessor, the CDP1802.

The CDP1802 is an LSI CMOS, 8-bit, register-oriented CPU. For this application, it is operated at 5 Vdc and at a clock frequency of 1.760 MHz.

When the *clear* button is pressed, the microprocessor registers are reset. A game is selected from the ROMs by entering the appropriate codes via the keyboard, as outlined in the instruction sheet supplied with Studio II. The game starts as the microprocessor executes the program instructions contained in the ROMs.

The memory has 2048 bytes of ROM and 512 bytes of RAM.

For programs contained in the resident or external ROM memory devices, 1024 bytes of ROM are used as an "interpreter" language. The interpreter ROM provides common game display patterns, scorekeeping, subroutines, alphanumerics, etc. The remaining 1024 bytes of ROM are mask-programmed resident game formats.

The RAMs are used for tv refresh, stack, and variable storage. Data bytes can be written into the RAM (memory write, MWR) or read out of the RAM (memory read, MRD) on command of the microprocessor.

When an external entertainment cartridge is plugged into the control socket, the resident-memory (ROM) is deselected and operation proceeds using the cartridge-memory (ROM).

Two 10-key pads provide the player/keyboard interface.

The keyboards were developed specifically for Studio II by the Deptford mechanical engineering group. The keys are arranged in the standard telephone-type format. Keyboards were selected in lieu of other means of player-interface because they provide the flexibility of permitting specific numerical entries as well as motion and direction entries.

The decision to make, rather than buy, the keyboard was motivated by several design objectives:

1) Proper button size and spacing for this type of product,

2) Ease of installation and assembly,

3) Good reliability, with a life test in excess of a million cycles per button,

4) All the above at a low cost commensurate with total product cost.

Also, tests showed that available low-cost keyboards could not withstand the pressures developed during the excitement of game-playing.

The keyboards are bused to a latched 4-bit-to-16-line decoder integrated circuit which is under program control for switch

Table I
Five resident game formats and four additional plug-in cartridge programs have been developed. Additional programs are currently being written and new ones will continually be developed.

The five resident programs included in the instrument are:

Doodle transforms the tv into an electronic blackboard.

Patterns can be programmed to design literally millions of interesting and attractive patterns on the tv screen.

Computer bowling starts with a bowling alley on the tv screen. The bowling ball moves up and down, until the player presses a key that sends the ball down the alley to strike the bowling pins. Pressing different keys will hook the ball in either direction or send it straight.

Freeway puts two racing cars on the screen. One racing car is controlled by the computer. The player controls the other car, and steers by keyboard control, to avoid collisions with the computer-controlled car. Accidents slow the player's car down and lower the score. This is a race against the clock, in a test of reaction time and coordination.

Addition allows one or two players to use either or both keyboards. A three-digit number appears on the screen. Each player must quickly add the three numbers, then enter the total on the keyboard. The faster the entry, the higher the score.

Plug-in home games presently completed are:

Space War—This module provides two games:

Horizontal intercept is for one player at a time who has available twenty steerable rockets to shoot at a fleet of fifty enemy spaceships. Hitting the small, fast-moving targets gives a better score.

Vertical intercept allows two players to attempt hitting a vertically moving target by adjusting the trajectory of their missiles.

Fun With Numbers—This module provides three games:

Guess the number puts a single player against the computer, which selects a random three-digit number. As the player guesses what the number might be, the computer gives clues as to how close the guess is. If the player can't guess the numbers within twenty tries, the computer-generated number is displayed and the player loses.

Guess the number allows each player to load a secret number into the computer for the other player to guess. Clues are given as in the one-player game by the computer, and the first player to guess the other's number wins. A good test of logical thinking.

Reverse displays the digits 1 through 9 in random order. The object is to unscramble the numbers in as few "moves" as possible. A "move" consists of reversing groups of numbers according to the key number pressed. The game allows thirty "moves" before it declares the player a loser.

TV School House I (yellow series/orange series)—Each series contains both Social Studies and Mathematics sections. The *yellow* series is elementary in skill level while the *orange* series is more advanced. Comprehensive handbooks are included. This series was developed with the guidance and help of education specialists.

sampling. A second microprocessor output instruction is used to sample any switch by connecting it to the "flag" lines (EF3 or EF4). Programs are written so as to sample any combination of keyboard A or keyboard B switches in any order.

The video display generator is a custom IC, CDP1861.

The video display generator, RCA CDP-1861, is a customized integrated circuit designed to work with the CDP1802 microprocessor. Direct Memory Access (DMA) and Interrupt signals from the display generator to the microprocessor provide a continuous 32×64-dot display matrix.

A transistor-array IC, CA3086, is used as the oscillator/modulator.

Two separate oscillators are used to generate the rf carriers for operation on either channel 2 or 3, to avoid possible interference from local broadcast stations.

The entire assembly is very carefully laid out and shielded to meet the very stringent radiation requirement of 15 μV/m at 1 meter. This assures that no interference will be generated on nearby tv sets.

The modulation percentage is limited to 55% so that the tv receiver will not see a saturated white level which might cause picture tube burn-in from lack of picture motion during prolonged use.

The selector-switch box is the interface between the tv antenna, tv receiver, the Studio II rf-output cable, and the dc power supply.

The selector-switch box normally mounts on the rear of the tv receiver, close to the antenna terminals. In the "Studio II" position, the switch disconnects the tv antenna from the receiver, connects the rf output into the receiver, applies power to the Studio II via the rf cable, and transforms the unbalanced 75-ohm Studio II output to 300 ohms, balanced for the tv receiver.

This function must be done and still meet the FCC rules, which state that the rf transfer switch must maintain 60-dB isolation between the Class I tv device and the antenna.

The power supply is a UL approved wall plug-in type providing 9Vdc at 250 mA.

The dc output cable plugs into the selector switch box where it is switched into the main Studio II console by the selector switch. The 9Vdc is converted to the 5Vdc needed for the circuitry in Studio II by means of a 3-terminal voltage regulator on the main pc board.

"Beep" sounds add another sensation to the game action.

A 555 timer-oscillator circuit, activated by a microprocessor-controlled signal, generates "beep" sounds at appropriate times during game action. The beeps are not intended to represent any particular sound. A switch is provided so that the sound can be shut off if desired.

List price

Optional retail price of Studio II is $149.95. The *Space War* and *Fun-With-Numbers* cartridges are optionally priced at $14.95 each. *TV School House* will have an optional list price of $19.95, since a comprehensive program manual is included.

Conclusion

In Studio II, RCA is offering a low-cost home video game that can be programmed for an almost unlimited variety of games by inserting new ROM cartridges; new game cartridges are being developed on a continuing basis. This capability should offer the consumer a clear advantage over the subtle variations on the "ball going back and forth" type of game that has been available for the past year or so. The range of Studio II games is limited only by the two dimensions of the tv screen, the solid-state memory capacity, and the imagination of the game-programmers.

Reprint RE-22-5-25| Final manuscript received February 23, 1977.

Bill Stonaker, as a Principal Member of D&SPD's Engineering Staff, is responsible for digital circuit designs, including the Studio II logic design and circuit configuration. He joined D&SPD in 1975, after having been associated with RCA's Equipment Design and Development Engineering Group in Harrison.

Walt Stobbe, as a Principal Member of D&SPD's Engineering Staff, is responsible for various rf and digital projects including the Studio II rf section. Prior to joining D&SPD in 1975, he was responsible for design engineering and development of solid-state test equipment products at RCA in Harrison.

Dave Callaghan, as Manager of Engineering, is responsible for all product development at D&SPD; these include tv accessories, color tv test jigs, CB radio, and automotive audio products. The Studio II home tv programmer is the most recent product.

Authors **Stobbe, Callaghan,** and **Stonaker.**
Contact them at:
Engineering, Distributor and Special Products Div., Deptford, N.J. Ext. PT-531

20-MHz CMOS-on-sapphire COSMAC microprocessor

Silicon-On-Sapphire (SOS) technology has produced a high-speed version of the COSMAC microprocessor three times as fast as its bulk-silicon counterpart.

G.R. Briggs
S.J. Connor
J.O. Sinniger
R.G. Stewart

CMOS technology is well established in low-power-dissipation and high-noise-immunity applications. Implementing CMOS with Silicon-On-Sapphire (SOS) technology, however, provides additional advantages of increased speed and greater circuit density.[1] This paper describes circuit approaches developed to exploit these advantages in the design of a high-speed SOS version of the COSMAC bulk-silicon CMOS microprocessor.[2,3]

The low parasitic capacitance of the SOS structure permits a three-fold increase in maximum clock rate—from 6.4 MHz for the bulk COSMAC microprocessor operating at 10 V to 20 MHz for the SOS design at the same voltage. The higher clock rate allows most COSMAC instructions to be fetched and executed in 0.8 μs. Power dissipation ranges from less than 1 mW in the quiescent state to a maximum of 100 mW when executing instructions with a 20-MHz clock. When operating at 10 V, the speed capability of the SOS microprocessor far exceeds that of other MOS devices and can be matched only by bipolar devices requiring multiple-chip CPUs and far greater static and dynamic power dissipation.

Chip architecture

The simplicity of the RCA COSMAC architecture, combined with a 5-transistor static storage cell, produces a completely static single-chip CPU with only 4,827 transistors.

Fig. 1 is a block diagram of the CPU. The principal features of the design are a 16-register scratchpad memory array, an 8-bit serial arithmetic logic unit, and extensive control logic for I/O devices. Each register of the scratchpad memory can be accessed either as two 8-bit data bytes (RHi, RLo), or as one 16-bit external memory address. To provide the I/O control capability and remain within a 40-pin package limitation, the 16-bit address is time-multiplexed on 8 pins (MA output). Incrementing of the 16-bit address is accomplished by two passes through an 8-bit incrementer.

The scratchpad registers use a unique 5-transistor storage cell (Fig. 2) that can be laid out in an area of 10.2 mil^2. Access in and out of the cell is via a single n-device transmission gate, T1, and a single bit line. Two adjacent cell columns contain corresponding-order bits of the RHi and RLo bytes. The column bit lines are multiplexed to the 8-bit incrementer input and memory address (MA) latches. Placement

Reprint RE-22-5-19
Final manuscript received January 25, 1977.

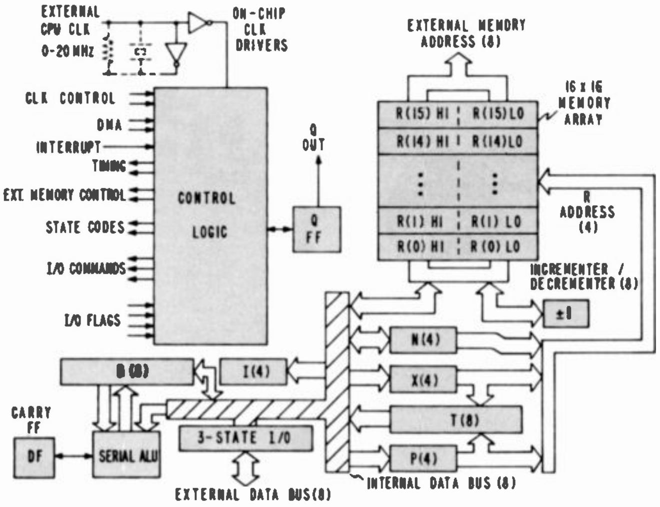


Fig. 1
SOS microprocessor, showing 16 by 16 scratchpad memory array, R, for temporary storage of data and external memory addresses; incrementer and serial ALU for altering register contents; registers N,X,P,T, and D for data manipulation; and logic for microprocessor control.

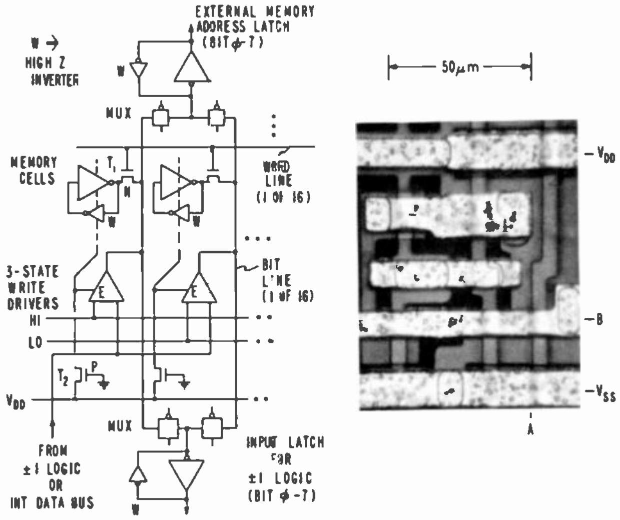


Fig. 2
Scratchpad memory (left), using the 5-transistor static storage cell (right). The supply voltages to the cell columns are reduced via transistors T2 during Write-1 operations to improve cell writing through the source-following gates T1.

of the two latches on opposite sides of the memory array permits access to the MA pins without the need for additional busing (see chip photograph, Fig. 7). Memory operation is described more completely in following sections.

Chip layout

The silicon-gate SOS process used obtains maximum circuit density and highest-speed operation for the chip. Also, a unitized logic cell layout procedure produces a logic layout optimized for minimum wasted chip area.

The wafer fabrication process used for the microprocessor includes two major improvements to the original aluminum-gate SOS process.

First, a self-aligned polysilicon gate obtained denser circuitry, increased transistor G_m, and reduced gate overlap capacitance.

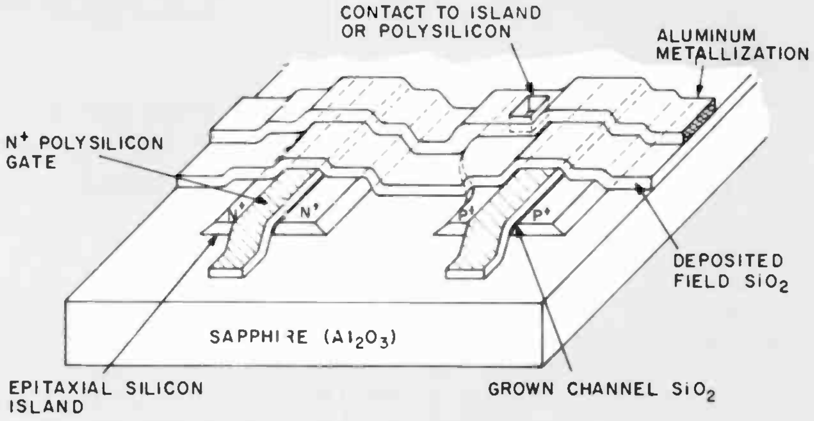


Fig. 3
Structure of CMOS-on-sapphire chip, showing epitaxial silicon islands with self-aligned polysilicon gates forming the n and p transistors. Etched aluminum metallization forms one level of interconnections; the polysilicon forms the second level.

George Briggs contributed to the development of high-speed silicon-on-sapphire counters and frequency synthesizers before working on the SOS microprocessor described here. He has also conducted research on a number of memory devices and systems. Dr. Briggs has received several RCA Achievement Awards and holds 22 U.S. patents.

Contact him at:
LSI Systems Design
RCA Laboratories
Princeton, N.J.
Ext. 2276

R.G. Stewart has been responsible for CMOS and SOS memory circuit design, including static 1024-bit RAMs and high-density CMOS ROM arrays, since joining the SSTC in 1973. Before coming to RCA, he worked on a number of microwave, bipolar, and CMOS devices.

Stephen Connor has been involved in the computer-aided design of the SOS microprocessor and CCL microprocessor interface circuits since coming to the SSTC in 1974. Prior to that, he worked on the initial process development of the thin-film transistor and silicon-on-sapphire technologies at RCA Laboratories.

Joseph Sinniger came to the SSTC in 1973, working on systems for digital tv tuning and the COSMAC microprocessor. Previous to that, he had been working on the analysis and development of communication systems at the RCA Laboratories.

Contact them at:
Solid State Technology Center
RCA Laboratories
Somerville, N.J.

Left to right: **Stewart, Ext. 6346; Sinniger, Ext. 6253;** and **Connor, Ext. 7044.**

Second, n+ doping of the polysilicon reduced the polysilicon sheet resistance so it could more usefully be employed as a second interconnection level. The improved CMOS-On-Sapphire structure is shown in Fig. 3. The substrate material is single-crystal sapphire of (1102) crystallographic orientation. This material is sliced into 18-mil-thick wafers, which are then ground, polished, and cleaned to remove surface contaminants that could cause transistor source-drain leakage.

Single-crystal silicon with the same crystallographic orientation as the sapphire is grown epitaxially on the sapphire surface to a thickness of 5000 Å. The silicon is then defined and etched into islands, which later will become the n and p transistors. A 900-Å channel oxide is next grown on the silicon islands; this is followed by a 5000-Å-thick polysilicon layer doped n+ to minimize its sheet resistance. The polysilicon layer is then etched to form the transistor gates and polysilicon interconnections. The island areas exposed at this step are next doped n+ or p+ to form the self-aligned-gate, complementary transistors. To complete the structure, a field oxide layer is deposited, contact openings are etched to the silicon and polysilicon, and aluminum is deposited and etched to form the metal interconnection level.

In describing the process, the doping of the silicon under the gate has been omitted because several methods for accomplishing this are currently in use: 1) all-n or all-p doping of the initial epitaxial silicon to produce enhancement transistors of one complementary type and deep-depletion enhancement transistors of the opposite type; and 2) selective n or p doping of the silicon islands to produce normal complementary enhancement transistors of both types. The former method produces transistor thresholds of less than 1 V for both p and n devices, while the latter approach typically produces transistors of higher G_m and sharper threshold characteristics.

The SOS microprocessor was designed to take advantage of the low thresholds offered by the deep depletion process, but can be used with either process variation.

With the SOS process, most of the electrical parameters are similar to those for bulk CMOS fabrication except for the wiring-to-substrate capacitance, which is greatly reduced. With SOS, capacitance to the ground plane located under the sapphire substrate is only 0.0004 pF mil for a 5-μm-width conductor; this capacitance is generally negligible. Having nearly eliminated the ground capacitance, one must consider the capacitive coupling between adjacent conductors; this coupling is typically 0.0015 pF/mil for two 5-μm conductors spaced 7.5 μm apart and generally cannot be neglected. This coupling is particularly important when it is into a high-impedance circuit node or between two such nodes.

Table I
Layout design rules (partial listing) for the standard logic cells from which the chip was assembled.

Minimum dimensions for logic cells
Polysilicon gate length—6μm
Silicon island and transistor width—5 μm
Island-island spacing—8 μm (10 μm oppositely doped)
Polysilicon interconnection width—8 μm
Polysilicon spacing—8 μm
Contact opening—5 × 10 μm (7 × 20 μm for n+, p+ split contact)
Metal interconnection width—10 μm
Metal spacing—8 μm

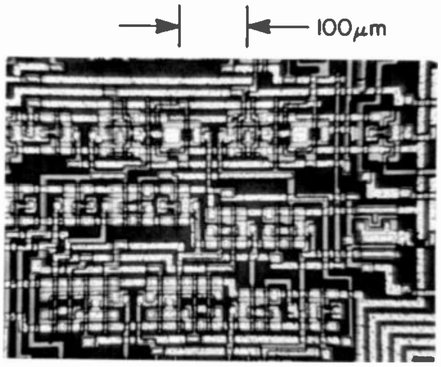


Fig. 4
Section of the control logic, which is constructed of inverter, 2- and 4-input logic gates, and latch cells. The cells are unitized in the x and y dimension consistent with the aluminization spacing to facilitate minimum-area layouts using trial-and-check cell-placement procedures.

The microprocessor chip was assembled from a relatively small group of standard logic cells.

Most of the cells were unitized in both the x and the y dimension, corresponding to the metallization interval of 18 μm (see Table I for an abbreviated list of design rules). Unitizing the cells facilitates rapid trial-and check cell placement procedures that lead to layouts of optimum area efficiency. An example of this approach is the logic layout shown in Fig. 4. The circuitry is assembled from standard cell inverters, latches, and 2- and 4-input logic gates. Each cell of a given type is contained within a rectangle of fixed x and y unit value, the cell blocks are placed on gridded sheets, and the metal and polysilicon interconnects are drawn with the grid spacing. This procedure is repeated using different cell placements until a minimum-area layout is obtained.

Although cell unitization speeds the development of area-efficient chip layouts, each cell must be area-efficient itself to realize a net gain for the approach. For example, 3-input logic gates were found to exhibit poor layout efficiency within the unitization constraints and were therefore not used.

Scratchpad memory design considerations

The 5-transistor storage cell requires less area than a conventional 6-transistor cell, but has an operating disadvantage because the access transistor T1 must operate as a source follower during Read-1 and Write-1 operations. For this reason it was necessary to design the memory carefully and computer-simulate its operation. Fig. 5 shows part of the final computer simulations for Read-0 and Write-1 operations. Protection against memory upset during the read operation is achieved by increasing the Miller-effect capacitance between the output and the internal nodes of the cell. When combined with a high-impedance inverter, W, the internal node is stabilized against upsets caused by transients on the bit line. This effect can be seen in the simulated Read-0 waveforms shown in Fig. 5. To read a 0 state, the capacitance on the bit line is discharged to ground; this must not disturb the logic-0 level stored in the cell. As shown in the figure, the output node in this case is displaced by 2.8 V from its quiescent level, but the interior node is displaced by only 0.9 V, assuring retention of data. Further upset protection is provided by a word-decoder design that limits the voltage risetime at the gate of T1 and also prevents connection of more than one memory cell at a time onto the same bit line.

During Write-1 operations, T1 source-follows, which limits the speed of the write operation, especially when high n-threshold voltage (T1 is an n transistor) is combined with low V_{DD}. To insure reliable writing over a 3- to 15-V range, p-type transistor T2, as shown in Fig. 2, is placed in series with the write driver so that a Write-1 operation reduces the supply voltage applied to the selected column of memory cells. This reduces the supply voltage to the cell during writing to approximately one-half its quiescent value within 10 ns. The reduced supply voltage to the cell allows writing over a V_{DD} range of 2 to 17 V, yet does not lead to storage loss in unselected cells if device thresholds are maintained under 1.0 V, which is generally possible with deep-depletion SOS processing.

The complete scratchpad memory, including write-drivers, decoders, and decoder-buffers, occupies a chip area of 66 by 73 mils. At a supply of 10 V, the read-access time is typically 80 ns, page-mode access time is 8 ns, and minimum write-pulse width is 22 ns.

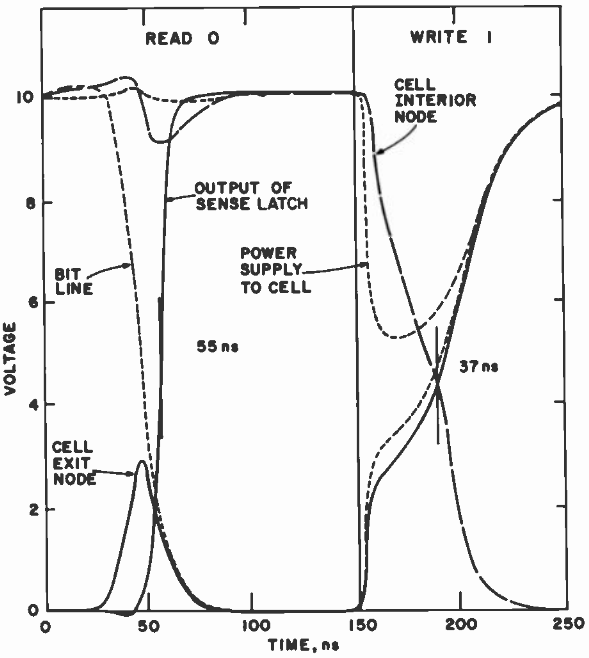


Fig. 5
Computer simulation of the scratchpad memory, indicating 55-ns read-access time and 37-ns minimum write time.

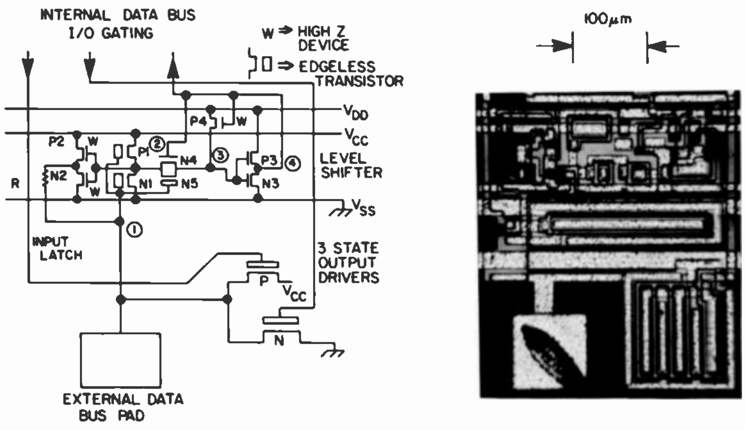


Fig. 6
Input level shifter and associated output drivers for one of the external data bus lines. The level shifter converts TTL logic levels to internal CMOS levels. All gates directly connected to the bonding pad are of an edgeless concentric design to maximize gate breakdown voltage.

High-speed interfaces for use with bipolar logic

The high-speed capability of the SOS microprocessor makes it possible to design fast microcomputer systems using a mix of CMOS/SOS and high-speed bipolar logic. Maximum performance in these systems, however, generally requires the CMOS logic to operate at a higher voltage than the 5-V levels typically powering the bipolar logic. A high-speed level shifter is therefore needed to convert bipolar logic levels to CMOS levels. An additional requirement for low-power-dissipation CMOS compatibility is that no dc current paths may exist between the power supplies of the two types of logic or between either power supply and ground.

Fig. 6 shows a circuit diagram of a high-speed level shifter developed to convert T^2L logic levels to CMOS levels. Transistors P1, N1, P2, and P3, N3, and P4 form two coupled static latches operating from power-supply levels V_{CC} and V_{DD}, respectively. Transistors N4 and N5 provide dc isolation between the two latches, but permit transmission of digital signals between them.

Test results show operation over a wide range of conditions. If a maximum level shift is required, V_{CC} can be operated as low as 3 V, transforming a logic-1 level as small as 1.0 V at the input to 15 V at the output. In a more typical application with $V_{CC} = 5$ V and $V_{DD} = 10$ V, T^2L logic levels of 0.7 V and 2.3 V are converted into CMOS logic levels of 0 V and 10 V, respectively, with a typical delay of 30 ns.

Results

Fig. 7 is a photograph of the SOS microprocessor chip, which measures 5.33 × 5.33 mm (211 × 211 mils), including the 4-mil-wide dicing streets. The scratchpad memory array is located in the upper right corner of the chip. Directly below the memory is the incrementer circuit, then the I, N, X, P, T, and D registers. The internal data bus is distributed through these registers on polysilicon. The external data bus drivers and input level shifters are located at the bottom right, and the control logic and ALU are located on the left side of the chip.

The clocking of the CPU is fully static and uses either an external clock input or an on-chip crystal oscillator. Static clocking permits stopping the clock at any point in the machine cycle. The relationship between instruction time, external-memory access time, chip dynamic power, and clock rate are illustrated by the curves shown in Fig. 8. At V_{DD} = 10 V, a maximum clock rate of 20 MHz is possible. At this clock rate, the required external-memory access time (approximately 4 clock cycles minus MA set-up time) is 150 ns. Preliminary results indicate that 30-MHz operation can be obtained at V_{DD} = 15 V.

The static design of the chip results in negligible (typically less than 1 mW) quiescent power dissipation. Most of the power dissipation is dynamic, given by $P = fCV^2$, where f is the clock rate, C is the total internal capacitance of the active nodes, and V is the supply voltage (V_{DD}). The capacitance is a function of the logical complexity of the instruction being executed. The power curves of Fig. 8 show averaged dynamic power for "typical" instruction strings, not including V_{CC} supply contributions in the output drivers. The power at V_{DD}=5 V and f = 8 MHz is 10 mW, increasing to 70 mW at 10 V and 20 MHz. The V_{CC} supply contributes an additional 30 mW at 20 MHz in driving 100 pF output loads. The total power dissipation and speed/power product are extremely low, even when compared with bulk-silicon CMOS. This is a result of the low capacitance of the internal nodes and also because relatively few of the nodes are active at a given time in the CPU.

Conclusions

Table II summarizes the characteristics of the CMOS-On-Sapphire microprocessor. The 20-MHz clock rate is comparable to that of bipolar logic, such as T²L, but the power dissipation is much less. Also, the complete CPU can be placed on a single chip, in contrast to the several chips required with conventional bipolar logic. The simplified microprocessor architecture and conservative design rules used make the chip suitable for high-volume, low-cost production.

References

1. Meyer, J.E.; Burns, J.R.; and Scott, J.H., Jr.; "High-speed silicon-on-sapphire 50-stage shilt register," 1970 *IEEE ISSCC Digest of Technical Papers*, p. 200.
2. Young, A.W.; "The CDP1802—a powerful CMOS 8-bit microprocessor," *this issue*.
3. Russo, P.M.; "Architectural trade-offs in the design of microcomputer systems," *this issue*.

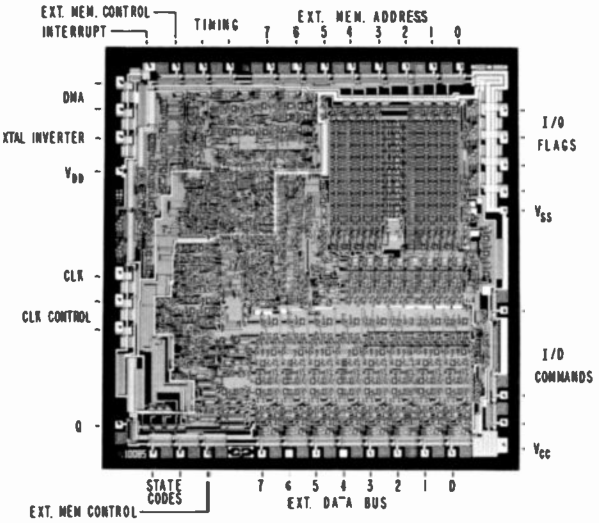


Fig. 7
Microprocesssor chip dimensions are 5.33 mm (211 mils) square; chip is mounted in a 40-pin package.

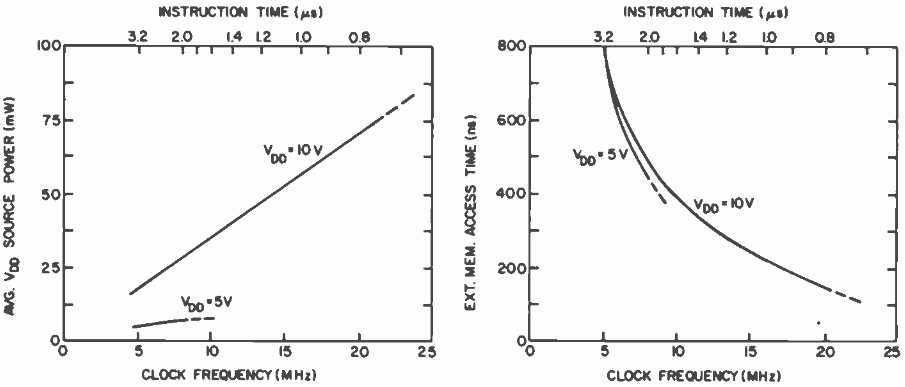


Fig. 8
External-memory access time and power dissipation versus clock frequency; preliminary results indicate operation to 30 MHz is possible at 15V.

Table II
Performance characteristics for the 1802S CMOS-on-sapphire microprocessor. Clock rate is comparable to that of bipolar logic but with much less power dissipation.

Chip size—5.33 × 5.33 mm (211 × 211 mils)
No. transistors—4827
Average chip area per transistor—6×10^{-3} mm²
Inputs—T²L-compatible
Outputs—T²L-compatible
Dynamic performance
Max. clock rate—20 MHz @ V_{DD} = 10 V
Avg. V_{DD} source power at max. clock rate—70 mW @ V_{DD} = 10 V
—10 mW @ V_{DD} = 5 V
Typical quiescent power dissipation—160 μW @ V_{DD} = 10 V
15 μW @ V_{DD} = 5 V
Typical circuit delays at V_{DD} = 10 V:
ALU logic—10 ns
Incrementer 8-stage delay—20 ns
Scratchpad memory access time—80 ns
Scratchpad memory write time—22 ns
Average speed-power product—4.0 pJ
Operating temperature range— −55°C to +125°C
(Max. clock rate—24 MHz @ 30°C, 16 MHz @ + 125°C)

Technology impact on microprocessors

W.A. Clapp| A. Feller

Continuing concurrent advances in solid-state technology and computers are pushing microprocessors toward faster speed, lower power, and higher performance.

The microprocessor—developed over the last five to seven years—merges semiconductor advances with computer-design technology. Advances in each of these independent fields stimulate advances in the other. In this environment, the one unchanging factor is constant change. This paper considers future trends of microprocessors, set against a background of changing technology and computer-aided design.

To see the rate of change of technology in clearer perspective, it is helpful to review the time span from scientific invention until the manufacture of the product for some key past developments. This time lag was about 112 years for photography, 56 years for the telephone, 35 years for the radio, 15 years for radar, 12 years for television, 6 years for the atomic bomb, 5 years for the transistor, 3 years for the integrated circuit, and about 1½ years for the microprocessor. Today we are into the second and third generations of microprocessors.

The result of this change is obvious in the commercial market, where one can see the impact of the personal scientific computer, with its better than 10-to-1 cost reduction in only three years. The digital watch has dropped in price by an order of magnitude, and increased its capability.

Such change is even more sobering in the military market, where it still takes 8 to 10 years from advanced development to fielded equipment. As a result, a manufacturer of military electronics may find himself in the development stage of an equipment when the design approach selected and the parts identified are upstaged by a newly available part which is faster, smaller, and cheaper. The commercial market is faced with product half-lives approaching 1 to 1½ years.

Background of microprocessor technology

In the early 1970s, semiconductor manufacturers continued to increase their capability to place more transistors on a single monolithic chip to the point where they could put a computer on a single chip. Large Scale Integration (LSI) had arrived. This was a simple computer, with perhaps some of it on two or three additional supporting chips, but it was a necessary step to stimulate microprocessor developments.

For years, the semiconductor field had been divided into two camps: one devoted to going faster and faster (the bipolar

Reprint RE-22-5-2
Final manuscript received November 24, 1976.

Al Feller came to RCA in 1951 and joined the Computer Division in 1958. He became active in the development of high-speed circuits and interconnection techniques. He has directed the design of more than 50 LSI devices and several LSI computer systems, the most recent constituting the ATMAC CMOS/SOS microprocessor.

Bill Clapp joined RCA aş a development engineer in 1965 and has been involved with the interrelationship of LSI technology and its impact on digital computers for the past ten years. He has been responsible for several advanced computer developments including the SUMC series, the B-12, and most recently the ATMAC.

Authors Al Feller (left) and Bill Clapp are examining the 16-bit ATMAC microprocessor developed by their group.

Contact them at: **Applied Computer Group, Advanced Technology Laboratories, Camden, N.J., Ext. PC3276 or PC3257**

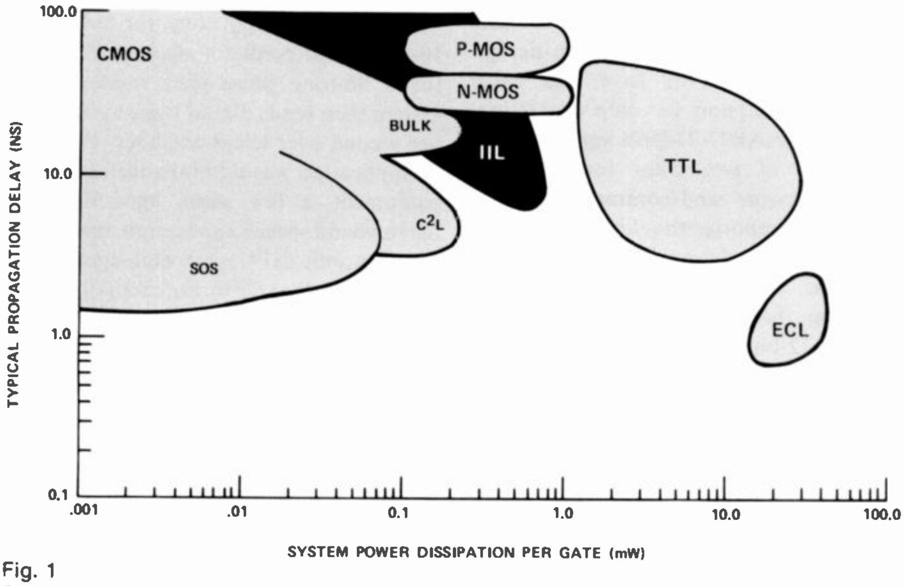


Fig. 1

System speed-power comparison, based on general-purpose logic, such as that in a microprocessor. With this chart, you can calculate the maximum level of integration possible for a given technology running at a given speed.

technology), and the other devoted to putting more and more transistors on a single chip (the MOS technology). Each camp was continually checking on the progress of the other and striving to incorporate the features of the other camp into its own product. As a result, the MOS camp developed technology variants that provided more speed, and the bipolar camp developed technology variants that provided a higher level of integration. These efforts resulted in at least fifty distinct technologies which are present in industry today. The goal of this competition is to provide higher speed and a higher level of integration than those presently available.

A high level of integration must, however, be achieved within a reasonable level of power dissipation; most common packages can handle approximately ½ W. Higher speeds have typically been achieved by higher power dissipation. Thus the ideal technology for LSI and very large scale integration (VLSI) provides the lowest system power dissipation and the fastest speeds. As a result, much attention is now focused on the speed-power product of newer variants of technologies. These speed-power products, however, are usually provided to us based on individual transistor or gate-level data. What is really important is the system-level speed-power product: the speeds achieveable at various power levels for a system implemented on a single chip.

Fig. 1 compares the more common technologies in the system context. This figure illustrates the system speed-power comparisons for general-purpose logic, such as that in a microprocessor. It results from considering duty cycles of gates in a general-purpose logic system where all the logic gates don't have to switch at top speed all the time and, in particular, the duty cycle for CMOS logic gates, which dissipate very little power when not switching. From this type of chart, one can calculate the maximum level of integration possible for a given technology running at a given speed.

The increasingly higher levels of integration result not only in the ability to make better microprocessors, but also total system equipment which has smaller sizes, lower power, higher reliability, and which breeds new equipment concepts not conceivable before LSI. In addition to these advantages, the cost per gate has been steadily declining because fabrication of LSI and VLSI is basically a batch process. Therefore, the more transistors made per step, the cheaper the resulting transistors become.

The benefits cited above are from the point of view of the system builder. On the other hand, semiconductor vendors want high-volume production on as few parts as possible. In fact, engineering is figured as 5 to 10% of sales. So, for each dollar of engineering time in designing a custom LSI circuit, a company needs $10 to $20 of parts business from chip production. The solution to this situation was a universal programmable LSI chip in the form of a microprocessor, adaptable to a wide market. As a result, there are now 30 to 35 unique microprocessors, plus their supporting chips. The microprocessor also created a large demand for another universal chip—the random access memory—which was already on the shelf.

Impact of computer-aided design

Computer-aided design (CAD) will also play a large role in future trends of microprocessors. CAD uses a set of computer programs to automatically lay out and generate the artwork to fabricate a custom-design LSI array. The Army (through ECOM at Fort Monmouth[*]) and RCA during the last several years have advanced CAD for LSI and VLSI complexities. The most successful approach has been the standard-cell CAD concept. Standard cells are basic building blocks which are of standard height and variable width to accommodate a wide variety of cell functions. These cells are designed, checked, and placed on magnetic tape for future reference. The logic designer then partitions his logic design into these standard cells and gives an interconnection list representing the desired logic function to the computer, which proceeds to automatically place and completely interconnect the cells selected to implement the desired logic. Magnetic tapes from CAD programs drive artwork generation equipment to make masks which are then used to fabricate LSI arrays.

In this way, CAD enables the system designer to use LSI without bearing its high non-recurring costs. CAD can reduce the non-recurring chip-development costs by a factor of 3 to 5 times, depending on the logic to be implemented. Custom-designed chips can be used independently of, or in conjunction with, existing off-the-shelf parts, including microprocessors, to acquire the system advantages provided by advances in technology. These advantages include reduced size, increased reliability, and higher performance, all at a lower cost.

Recently, a CMOS/SOS chip—an 8-bit slice of the RCA-developed ATMAC processor—was designed with CAD techniques and fabricated under contract DAAB07-75-C-1314. The 4560 transistors on this single chip yield about 1500 logic

[*] Several contracts have been involved, notably DAAB07-74-C0176.

gates. Many general-purpose computers built in the 1960s used 5000 to 10,000 logic gates and required two racks of hardware. Today these racks can be replaced by four to six LSI chips requiring only 0.001 of the power.

Current microprocessors and microcomputers

There has been much confusion in distinguishing between microprocessors and microcomputers; Fig. 2 clarifies the differences. In practice, the microprocessor may be less than 10% of a total microcomputer system. In both commercial and military applications, the microcomputer system, rather than the microprocessor, is of primary importance, but the microprocessor has been receiving all the attention. The same technology and techniques will continue to be applied to memories, control and timing logic, and peripheral interfaces to reduce the total cost of a microcomputer system.

Custom-design microprocessors may be tailored to an application or class of applications. The advantages of custom-design microprocessors over commercially available microprocessors are 1) an order-of-magnitude improvement in performance and 2) availability of a processor one to two years sooner. In fact, the equivalent may never be available commercially. Only with CAD is this concept feasible for the small-volume applications typical of the military market. The concept is also cost effective for large-volume applications during initial phases where the customer is likely to frequently change his requirements. Then as volume warrants it, a custom-design equivalent can be prepared. During the past four years, RCA has completed several custom-design microprocessors. Two have had some government support for chip processing. Contract DAAB07-73-0098 supported the processing of two chips for the B-12 microprocessor,[1] and contract DAAB07-75-C-1314 supported the chip processing of RCA's most advanced custom microprocessor, the ATMAC. The B-12 processor, fabricated in 1973, is an all-CMOS, 12-bit-wide processor with 47 instructions (including multiply and divide); it has built-in clock generation and bus-driver circuits. Available about two years ahead of comparable commercial microprocessors, the B-12 was incorporated into an Army Radiac demonstration unit (as a microcomputer) which provided both total-radiation-dose and dose-rate readouts with pre-settable alarm levels for each.

A more recent design is the ATMAC, a CMOS/SOS processor optimized for array-type calculations. In a typical signal-processing application, ATMAC provides a throughput of about 8 million instructions per second. This performance is an order of magnitude more than is available in the new AN/UYK-30, a standard military minicomputer. ATMAC is also bit-slice modular in 8-bit slices. A typical 16-bit version is on four chips, provides 189 instructions, and requires only 3/4 W. Fig. 3 shows ATMAC as a component of a generalized microcomputer system. The program memory is ROM storage, the data memory is RAM, and the I/O devices are selected for a particular application. One present application has ATMAC at the heart of a narrowband speech system. Here the ATMAC executes algorithms for not only a 10-pole linear predictor algorithm but also for a 16-tone phase-shift modem. The system then sends digital voice at 2400 bits per second over telephone lines. This type of application was unthinkable for tactical equipment a few years ago. For this narrowband speech application, the special function unit (SFU) is a high-speed multiplier, but other SFUs are useful for other applications to further enhance the capability of ATMAC .

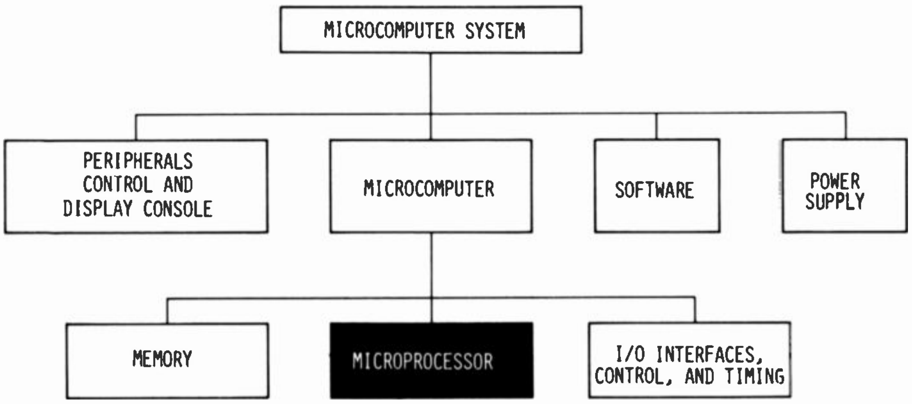

MICROCOMPUTER SYSTEM
PERIPHERALS CONTROL AND DISPLAY CONSOLE
MICROCOMPUTER
SOFTWARE
POWER SUPPLY
MEMORY
MICROPROCESSOR
I/O INTERFACES, CONTROL, AND TIMING

Fig. 2
Typical microcomputer hierarchy. This drawing places the microprocessor in system context with memory, input/output devices, software, and support circuits.

Future trends in microcomputers

Handheld processors today outperform many racks of equipment made just a couple of years ago. The future trends of microcomputers are in technology development and computer architecture. It is reasonable to project continuing advancement in the level of integration possible on a monolithic chip; a growth of five to ten times improvement in the next five to seven years is not unreasonable. This means from 50,000 to 100,000 transistors on a single chip, and underlines the necessity for lower-power technologies to keep the chip dissipation around ½ W—an average of about 10 μW/transistor in the system.

At the microprocessor level, three distinct trends are developing: The first is the move from 8-bit-wide processors to 16-bit-wide processors. A second trend is toward the integration of the processing unit, some read/write memory and read-only memory, and some peripheral control all on the same chip. A third trend is that of developing custom-design microprocessors which are used either alone or in conjunction with other off-the-shelf microprocessors to implement higher-performance or special-purpose systems. Thus, these new technologies and different organizations will allow use of additional logic to make processors faster and more powerful.

For military applications, the search is on for standards in this evolving microprocessor world. Spending more than $100 million annually on semiconductors, the military and prime system contractors are incorporating commercial microcomputers in future equipment. From this effort, *de facto* standards are now surfacing; three of the more often mentioned are the Intel 8080, the RCA 1802, and the Motorola 6800. Most applications to date, however, are either

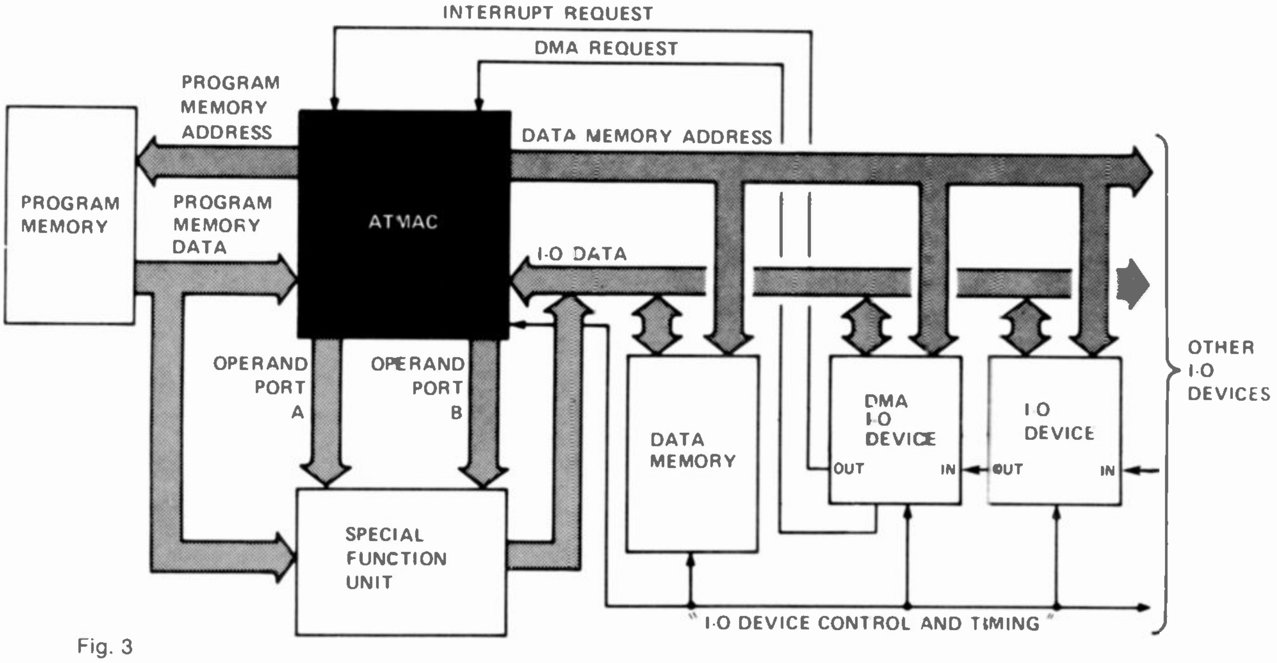


Fig. 3
ATMAC microprocessor as a part of general microcomputer system. At present, ATMAC is RCA's most advanced custom microprocessor.

controllers or dedicated real-time data processors. The military is also emphasizing requirements for second- and third-sourcing, and for full military-specification parts for not only the microprocessors but also memories, I/O devices, and other components.

At the system level, there is a trend away from a complex, centralized, data processor and toward microprocessors as controllers for special-purpose black boxes which themselves might use custom-LSI or custom-design microprocessors. This concept leads to distributed processing where the total job is accomplished by a number of small processors, each dedicated to part of the total problem and each working on its own segment of the problem.[2] Typical studies in this field have already been undertaken by the Air Force in its Distributed Processor Memory efforts and by NASA-JPL in its Unified Data System. Older concepts, such as ILLIAC IV, are dwarfed in comparison to what is now possible.

A total system could include anywhere from one to several hundred microcomputers, adding considerably to the reliability and flexibility of the system. In the limit it leads to putting software into hardware where each microcomputer is executing a dedicated segment of the total application software. This trend will also aid the expensive and growing problems of software development costs.

In regard to software, assembly code is most frequently used for dedicated functions. As the inherent speed of microprocessors increases, and the cost of memory decreases, the trend will be toward the use of more standardized higher-level languages such as Fortran, CMS, PL/M, and DOD-1. Compilers for these higher-order languages are under development. However, PL/M compilers already exist for the Intel 8080 and the Motorola 6800.

Summary

The major trends discussed here can be summarized as follows:

- Microprocessors and microcomputers will become the building blocks of future systems.
- The microprocessor market will condense to 8 to 10 types of microprocessors from the 30 to 40 types presently available.
- A move to a 16-bit data path microprocessor is a natural evolution.
- There will be a move to minimize support chips by including such things as clock generation logic, and both read-write and read-only memories on the same monolithic chip.
- Customized microprocessors are available today which can execute 8 million instructions per second, and this performance will increase by a factor of 2 to 4 in the next 5 years.
- Use of multiple microcomputers will increase in two directions—1) dedicated hardware performing dedicated functions, and 2) more general-purpose distributed and array configurations where several hundred microcomputers will be interconnected for a given application.
- Future systems will use off-the-shelf LSI microprocessors and support parts where possible, perhaps supported by custom-design LSI.
- However, for higher performance, future systems will require more custom-design LSI (including custom microprocessors) supported by off-the shelf microprocessors and other parts where their slower speed can be utilized.

References

1. Orga, S.J.: "The B-12 an automotive microprocessor," *RCA Engineer*, Vol. 22, No. 2 (Aug-Sep 1976) p. 43.

2. Russo, P.; "Architectural trade-offs in desigr of microcomputer systems," *this issue*.

3. Clapp, W.A., "LSI computer design—SUMC DV," *RCA Engineer*, Vol. 18, No. 2 (Aug-Sep 1972) pp. 88-94.

4. Feller, A., "LSI computer fabrication SUMC/DV," *RCA Engineer*, Vol. 18, No. 2 (Aug-Sep 1972) pp. 82-87.

5. Feller, A.; "Design automation techniques for custom LSI arrays," AGARD Lecture Series No. 75 (Apr 1975) pp. 7.1-7.15.

6. Merriam, A.; and Zieper, H.S.;"Simulation: methodology for LSI computer design," *RCA Engineer*, Vol. 18, No. 2, (Aug-Sep 1972) pp. 77-81.

7. Feller, A.; "Automatic layout of low-cost quick-turnaround random-logic custom LSI devices," 13th Design Automation Conference, San Francisco, CA (Jun 1976) *Proceedings* pp. 79-85.

8. Feller, A.; and Noto, R.; "Random logic custom LSI and VLSI using the standard cell approach," 1976 GOMAC, Orlando, FL (Nov 1976).

Short course in DIGITAL ELECTRONICS, Part 4

Digital filters

This fourth article of the series delves into the design of finite-impulse-response filters.

L. Shapiro

The preceding article of this series dealt with infinite-impulse-response (IIR) filters; such filters, when activated by an impulse input continue to produce output pulses for an indefinite time period. This article considers finite-impulse-response (FIR) filters which, when activated by an impulse input, produce only a limited number of pulses.

Finite impulse response may be achieved by using a so-called "window" to truncate the output pulse train; this approach will be treated first in the present article. A second method, utilizing certain Fourier transform methods, will then be considered. A third method, involving optimization methods, will be commented upon briefly.

The FIR filter versus the IIR filter

Our present study begins with the difference equation for the IIR filter in which the output depends upon both previous inputs and previous outputs. [Such an equation may take the form given in the second-order equation (Eq. 1) of the preceding article.]

$$y(n) + b_1y(n-1) + b_2y(n-2) = a_0x(n) + a_1x(n-1) + a_2x(n-2) \quad (1)$$

The relationships represented by Eq. 1 can be expressed by the block diagram of Fig. 1 in which each z^{-1} element indicates a delay of one sampling interval T.

Evidently, the IIR digital filter (as represented in Fig. 1) includes feedback loops and, hence, is subject to the instabilities and other peculiarities of feedback systems. A digital filter, however, need not be a feedback system. Thus, if we set all of the b_i coefficients in Eq. 1 equal to zero, we obtain

$$y(n) = a_0x(n) + a_1x(n-1) + a_2x(n-2) \quad (2)$$

The digital filter represented by Eq. 2 may be realized by the block diagram of Fig. 2.

In general, both FIR and IIR filters may be implemented by either recursive or nonrecursive techniques (e.g., feedback or nonfeedback systems). In practice, however, the IIR filter is usually more conveniently realized with a recursive technique while the FIR filter is usually more easily realized with a nonrecursive technique. Alternatively, the FIR filter, as discussed below, may be realized using Fourier transform methods.

An interesting aspect of the nonrecursive difference equation (Eq. 2) is that it inherently represents a finite impulse response system. Thus, if our digital impulse input is represented as

$$x(n) = 1 \quad \text{for } n=0$$
$$x(n) = 0 \quad \text{for } n\neq 0 \quad (3)$$

we come up with the following finite series after iteratively substituting these values in Eq. 2.

$$y(0) = a_0$$
$$y(1) = a_1$$
$$y(2) = a_2$$
$$y(n) = 0 \quad \text{for } n \geqslant 3 \quad (4)$$

We therefore note that when activated by a digital impulse input, our finite-impulse-response filter produces only a fixed number of nonzero pulse outputs. The general form of Eq. 2 may be written as

$$y(n) = \sum_{i=0}^{k} a_ix(n-i) \quad (5)$$

where a digital impulse input can produce, at most, $k+1$ nonzero terms at the output.

Advantages of the FIR filter

1) The FIR filter is readily achieved using the nonrecursive approach of Fig. 2. Hence, it may conveniently be realized with an absolutely stable nonfeedback system.

2) When the FIR filter is realized with a nonfeedback system, errors resulting from quantization or round-off, or errors in coefficients, are more easily predictable and usually less critical than in (IIR) feedback systems.

3) FIR filters may easily be designed with ideal linear phase characteristics. This very important feature is not usually possible with existing IIR design procedures.

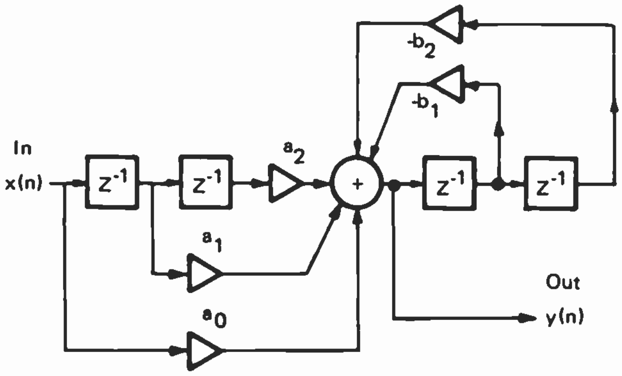


Fig. 1
Recursive (feedback) system showing a possible realization of the digital filter represented by difference equation 1.

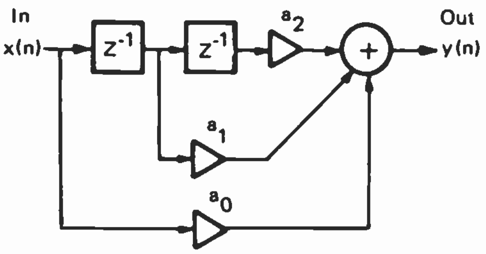


Fig. 2
Nonrecursive system is absolutely stable. It is a possible realization of the digital filter represented by difference equation 2.

4) As shown later in this article, the FIR filter may be achieved using convolution techniques. This enables us to make use of the fast-Fourier transform to provide greater computational flexibility and to help compensate for the inherently greater delay of the FIR approach.

FIR filter disadvantages

1) The FIR filter normally requires a higher order of transfer function (more terms in the difference equation) than does an IIR filter for similar sharpness of amplitude response.

2) Since the time delay depends upon the number of terms, the FIR filter normally requires a greater time delay for its operation than does the IIR filter. This delay may become quite large and possibly unacceptable.

3) The methods available for the design of FIR filters do not lend themselves naturally to meeting direct specifications such as passband and stopband ripple limits. Thus, the design procedure may require a number of iterations before a final acceptable version is obtained.

The ideal phase characteristic

If the FIR filter is realized using the nonrecursive approach of Fig. 2 (or Eq. 5), an ideal phase characteristic may easily be achieved. Thus, the transfer function may be developed from the z-transform formulation as

$$H(z) \rightarrow H(e^{j\omega}) = | H(e^{j\omega}) | \, e^{j\theta(\omega)} \qquad (6)$$

where we have replaced the z variable by $e^{j\omega}$ to reveal the frequency response. The desired linear phase relationship then takes the form

$$\theta(\omega) = -\alpha\omega \qquad -\pi \leqslant \omega \leqslant \pi \qquad (7)$$

The indicated limits for ω follow from the fact that our digital filter is cyclical in its frequency characteristic over these limits due to the original sampling process.

Although we will not burden the reader with the related proof, it may easily be shown[a] that Eq. 7 is satisfied if the impulse response of our FIR filter has the symmetry

$$h(n) = h(N-1-n) \qquad (8)$$

where N corresponds to the total number of terms or pulses resulting from a digital impulse input. Eq. 8 shows that the impulse response (in the time domain) has even symmetry about the $N/2$ point along the time axis. In this case, the coefficient of ω in Eq. 7 becomes

$$\alpha = (N-1)/2 \qquad (9)$$

It becomes important at this point to examine time delay as it applies to time difference between the input and output of a filter. It is customary to distinguish between phase and group (or envelope) delay as follows

$$\textit{Phase delay} = T_p(\omega) = -\,\theta(\omega)/\omega \qquad (10a)$$

$$\textit{Group delay} = T_g(\omega) = -\,d\theta(\omega)/d\omega \qquad (10b)$$

Of the above two types of delays, the group delay $T_g(\omega)$ is the more important since it must be constant over the frequency spectrum occupied by the input signal if the waveshape is not to be distorted. The absolute value of the group delay is often, but not always, of lesser concern.

If we make use of Eqs. 7 and 9 it becomes evident that α represents both the phase and group delay for our ideal phase characteristic and that only for the case of odd N is the filter equal to an integer number of intervals.[b] Examples of impulse responses, both even and odd in number, which satisfy Eq. 8, are shown in Fig. 3. Note that the delay in each case corresponds to the number of sample intervals between the first pulse, $h(0)$, and the center of symmetry. This value is equal to α which is, of course, constant for a given system of this type.

Where only preservation of waveshape is required, the alternate symmetry shown below will also suffice.

$$h(n) = -\,h(N-1-n) \qquad (\text{for } 0 \leqslant n \leqslant N-1) \qquad (11)$$

It may be noted that the symmetry of Eq. 11 is odd about the $N/2$ point on the time axis as compared to the symmetry of Eq. 8, which is even about the same point. In either case, however, the delay, α, follows the relationship of Eq. 9. Examples of impulse responses with both even and odd N which conform to the symmetry requirement of Eq. 11 are shown on Fig. 4. It may be noted that impulse responses of the odd symmetry of Eq. 11 may be valuable for the design of wideband differentiators and Hilbert transformers (see, for example, Rabiner and Gold, p. 164 *et seq.*)

Design techniques for FIR filters

Three basic approaches are used to design FIR filters.

These approaches have been given various names by different authors. However, all deal essentially with the same three methods of attack:

1) *The window method*: We develop the transfer function for the desired filter in the form of a Fourier series which is then truncated and the remaining coefficients are related to the desired impulse response. However, the truncation necessarily distorts the desired response. This distortion can be reduced by modifying the Fourier coefficients using windowing techniques.

2) *Frequency sampling*: We again expand the transfer function, this time making use of both the discrete Fourier transform and z-transform formulations. We then incor-

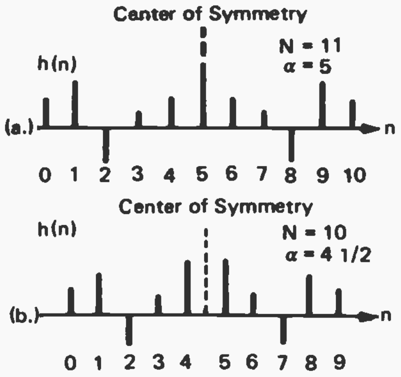


Fig. 3
Impulse responses satisfying the conditions for an ideal phase characteristic. a) **N odd**, the phase delay is an integer number of sample intervals. b) **N even**, the phase is not an integer number of sample intervals.

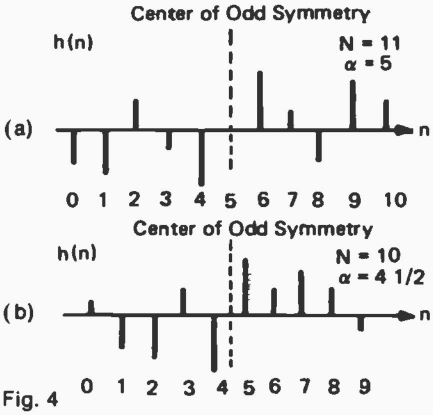


Fig. 4
Impulse responses satisfying conditions for constant group delay. a) **N odd**, group delay is an integer number of sample intervals. b) **N even**, group delay is not an integer number of sample intervals.

[a] See Ref. 2, pages 77-81.

[b] In keeping with customary practice, we have dropped the constant sampling interval T from the arguments of H and h. More strictly speaking, we should write $h(nT)$ and $H(z) \rightarrow H(e^{j\omega T})$. α is then more exactly given by $(N-1)T, 2$ in Eq. 9 where it has the proper units of seconds. T reappears later in our treatment of windows.

porate the desired frequency (rather than impulse) response coefficients directly in the expression for the frequency response in the z-transform domain. We now have a digital filter with exactly the desired response at the sampling points along the *frequency* axis. The remaining problem is to adjust matters so that the frequency response becomes acceptably well behaved in regions between these sampling points.

3) *Optimal filter design methods*: In this method, we define a transition band in addition to pass and stop regions. The frequency response at the sampling points in the transition band are now considered to be the variables, since their value is (deemed to be) of no consequence in the operation of the filter. We then seek to vary these transition-band responses to optimize response in the important passbands and stopbands. A number of methods have been used including the Chebyshev approximation, linear programming, and related algorithms (for computer use).

The windowing technique

We start with a desired frequency response for the digital filter.

A convenient formulation is that for the so-called amplitude response, $A(f)$, which represents the amplitude rather than phase behavior for a given frequency sensitive system. $A(f)$ is obtained from the absolute value of the applicable transfer function whether this be in the Laplace or in the z-transform language. Thus, we may say

$$|H(s)| \rightarrow |H(j\omega)| \rightarrow A(f) \qquad (12a)$$

or

$$|H(z)| \rightarrow |H(e^{j\omega})| \rightarrow A(f) \qquad (12b)$$

where the following two relationhips are understood

$$s = \sigma + j\omega$$
$$z = e^{sT} = e^{(\sigma + j\omega)T} \qquad (13)$$

The (constant) sampling interval T is often considered to be unity and hence does not usually appear in Eq. 12,[b]

The first step in using the windowing approach is to expand the desired frequency response in the form of a conventional Fourier series.

This is possible since the frequency response of our digital filter is inherently *periodic* at the sampling frequency (as a result of the sampling process). We now attach a subscript d to our amplitude response to indicate that this is the theoretically *desired* amplitude response rather than the actual response which we can ultimately obtain from our digital filter. The result of the Fourier expansion is

$$A_d(f) = \frac{\alpha_o}{2} + \sum_{m=1}^{\infty} \alpha_m \cos 2\pi m T f \qquad (14)$$

where

$$\alpha_m = \frac{4}{f_s} \int_0^{f_s/2} A_d(f) \cos 2\pi m T f \, df$$

There are two points of interest to be noted in Eq. 14. First, to represent accurately our desired frequency behavior, $A_d(f)$, an infinite number of terms must be included in the summation, and, second, the variables are reversed as compared to the usual use of the Fourier series. That is, we are expanding a periodic frequency rather than a time function and so obtain a series of *time*, rather than frequency coefficients, namely, the α_m. In other words, we start with a continuous periodic *frequency* function and obtain, thereby, a series of terms with coefficients α_m, each representing the strength of a term located at point mT along the *time* axis.

Two manipulations can now be performed: first, we normalize frequencies to the folding frequency, f_0, which is equal to one-half of the sampling frequency f_s. Second, we place our series in the usual complex exponential Fourier form. Our result is

$$A_d(\nu) = \left| \sum_{m=-\infty}^{\infty} c_m e^{jm\pi\nu} \right| \qquad (15a)$$

where

$$\nu = f/f_0$$
$$f_s = 1/T$$
$$f_0 = \tfrac{1}{2} f_s$$

and

$$c_m = \int_0^1 A_d(\nu) \cos m\pi\nu d\nu \qquad (15b)$$

Eq. 15 relates the desired amplitude response $A_d(\nu)$ to a series of pulses of strength c_m *in the time domain.*

We may now engage in a short excursion into the z-transform. The z-transform of a general transfer function $H(z)$ is defined as follows:

$$H(z) = \sum_{n=0}^{\infty} h(nT) z^{-n} \qquad (16)$$

Let us assume that $H(z)$ is the transfer function of a system being acted upon by a digital impulse $X(z)$. We thus obtain for our output $Y(z)$

$$Y(z) = H(z)X(z) \qquad (17)$$

The z-transform of our impulse input, however, is unity. Hence, Eq. 17 reduces to

$$Y(z) = H(z) \qquad (18)$$

In addition to the obvious fact that our transfer function, $H(z)$, now also represents the output of our system, a very interesting fact is that the time series $h(nT)$ associated with this function (see Eq. 16) also represents the actual impulse response of our system in the time domain. This is entirely in accord with the properties of the z-transform as developed in the Appendix of part 3 of this series.

Following conventional theory, we may now extract the frequency response of the transfer function $H(z)$ of Eq. 16 by replacing our z with $e^{j\omega}$.

$$H(e^{j\omega}) = \sum_{n=0}^{\infty} h(nT) e^{-j\omega nT} \qquad (19)$$

Since ωT is equal to $\pi\nu$,[c] we may normalize Eq. 19 in the manner of Eq. 15, above. In addition, by restating the summation limits of Eq. 19, we may change the sign of the exponent in the exponential factor and obtain the following (using absolute values):

$$|H(e^{j\pi\nu})| = \left| \sum_{n=-\infty}^{0} h(nT) e^{j\pi\nu n} \right| \qquad (20)$$

Comparison of Eq. 20 with Eq. 15 brings out the following key points in our design of the digital filter:

1) The representation of our desired filter in the z-formulation is entirely equivalent, on a term-by-term basis (with appropriate manipulation of indices), to the expansion of its desired frequency response as a Fourier series.[d]

2) In either the Fourier- or z-transform, the coefficients of the various terms represent the *impulse* response of the desired filter.

The desired filter impulse response is immediately available from Eq. 15b. Therefore, we should have no problem in realizing this filter since our system now works out to a simple difference equation of the type of Eq. 2; Eq. 2 may be immediately realized as shown in Fig. 2.

[c] This follows at once from $\omega T = 2\pi f(1/f_s) = \pi f/f_0 = \pi\nu$

[d] The restatement of limits in Eq. 19 to show a more convincing relationship with Eq. 15 is really not necessary since the negative sign in the exponent under the summation does not affect the result of taking the absolute value. The question of limits is quite interesting, however, and is treated below.

There is a fly in the ointment, however. Our summations require an infinite number of terms. This is obviously unacceptable, since each term represents an additional delay of one sampling interval T. We must establish an acceptable number of terms (corresponding to an acceptable number of delay elements in our system such as shown in Fig. 2) and rewrite our Eq. 15a accordingly. The response now will no longer be the one desired, however, since loss of the higher-order terms will necessarily result in distortion.

$$A_d(\nu) \neq A(\nu) = \left| \sum_{m=-M}^{M} c_m e^{jm\pi\nu} \right| \qquad (21)$$

Our expression for the evaluation of the impulse response coefficients c_m (Eq. 15b) is unchanged and is still based on the desired amplitude response $A_d(\nu)$. Furthermore, by use of the cosine function in the integrand, we obtain an even symmetry for the various c_m.[1] The result will be a symmetry of the type of Eq. 8 and Fig. 3, which, in turn, will give us the highly valued ideal linear phase response.

A critical feature of the above development is that the coefficients of the Fourier expansion of the frequency response give us exactly the coefficients needed in the z-transform formulation of our transfer function. We may therefore rewrite Eq. 21 in a more general form utilizing z-transform language as follows:

$$H(z) = \sum_{m=-M}^{M} c_m z^{-m} \qquad (22)$$

A problem of a rather unforeseen nature occurs upon examination of Eq. 22; our impulse response includes pulses with a negative time index. This situation is considered to be "noncausal." It may, however, be corrected and become "causal" by delaying the entire sequence by M sampling time intervals. The first (output) pulse will then occur at time $t = 0$, e.g.,

$$H(z) = z^{-M} \sum_{m=-M}^{M} c_m z^{-m} \qquad (23)$$

The rather awkward appearance of Eq. 23 may be corrected by the following reassignment of indices.

$$H(z) = \sum_{i=0}^{2M} a_i z^{-i} \qquad (24)$$

where $a_i = c_{M-i}$

We now have our transfer function for the desired digital filter except that the problem of truncation remains. The transfer function will not give us exactly the desired frequency response because of the missing higher order terms. Abrupt truncation of a Fourier series will, in general, produce an overshoot of approximately 9% at the point of discontinuity with a series of "ripples" on either side of the discontinuity, see Fig. 5. This figure shows the effect of truncation for the case of the ideal (rectangular) filter.

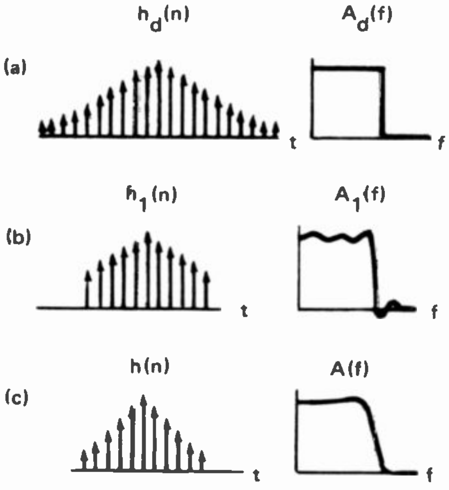


Fig. 5
Effect of truncation on the frequency response of an ideal FIR digital filter.

a) **Impulse response** without truncation reproduces the desired rectangular frequency response.

b) **Impulse response** with truncation leads to a distorted frequency response.

c) **A "window"** applied to the truncated impulse response reduces the distortion.

The windowing operation

The windowing operation may be treated as shown in Fig. 6, in which we apply the window function directly to the infinite series obtained by the Fourier-series expansion. The resulting truncated windowed series can then be inserted directly, (as the coefficients) into our transfer function in the z-domain. It may be noted that our window function automatically becomes truncated by multiplying the undesired higher-order terms by zero and adjusting the amplitudes of the nonvanishing portion of the series by means of appropriate multiplying factors w_m. The term-by-term modification of the amplitudes of the impulse coefficients has the simple form

$$c'_m = c_m \times w_m \qquad (25)$$

In the frequency domain, however, the above multiplication works out to a convolution integral.

$$H(e^{j\omega}) = \frac{1}{2\pi} \int_{-\pi}^{\pi} H_d(e^{j\theta}) \; W(e^{j(\omega-\theta)}) d\theta \quad (26)$$

The convolution process is shown in Fig. 7, where it is clear that the shape of the

a)
Terminated Series
=
Infinite Series
x
Window Function

b) $c'_m = c_m \times w_m$

c) $H(e^{j\omega}) = \underbrace{\frac{1}{2\pi} \int_{-\pi}^{\pi} H_d(e^{j\theta}) \, W(e^{j(\omega-\theta)}) \, d\theta}_{\text{Convolution Integral}}$

Fig. 6
Window function applied to infinite impulse response.

a) **Block** diagram representation.

b) **Direct time-domain** multiplication.

c) **Frequency-domain** convolution.

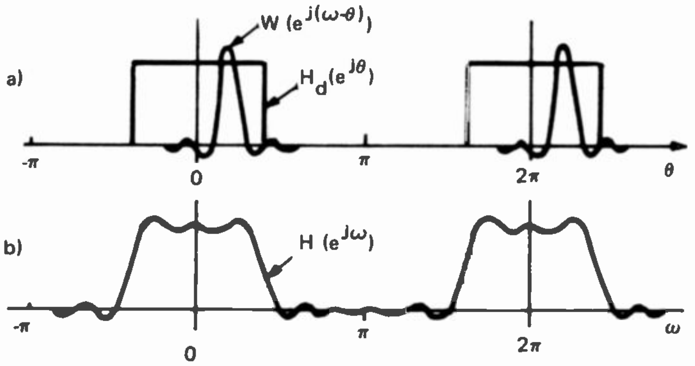


Fig. 7
Frequency response of the digital transfer function developed by convolution of the desired function with a "window." The frequency response repeats at multiples of the sampling frequency. a) **Two convoluting** waveshapes. b) **Result** of the convolution.

[1] The choice of m as the running index in the summation is appropriate even though these terms, in addition, represent the impulse response in the time domain.

window function *in the frequency domain* is all important in determining the final shape of the frequency response of our digital filter. It is evident that the ideal window should have its energy concentrated in a single narrow pulse and have vanishingly small side lobes. Fig. 7 also shows the cyclic repetition of this frequency response at the first harmonic of the sampling frequency.

Design examples

The general problem is to design a digital filter with the following characteristics (insofar as possible):

$$A_d(f)\begin{cases} = 1.0 & 0 \leqslant f \leqslant 125 \text{ Hz} \\ & (\text{or } 0 \leqslant \nu \leqslant 0.25) \\ = 0 & \text{elsewhere} \end{cases}$$

$$f_s = 1 \text{ kHz}$$

Impulse response is to be limited to 20 delays, e.g., $2M = 20$ (see Eqs. 23 and 24).

We have given M the value of 10, although the total number of permissible delays is 20. This follows from the fact that we have assigned M to represent the time delay associated with the displacement of the impulse-response time series as given in Eq. 23 in order to obtain a causal series. The total number of time displacements, however, is $2M$, and the total number of terms, or impulse-response pulses, is 21.

The desired frequency response, normalized to the folding frequency of 500 Hz, is shown on Fig. 8 where, for the sake of completeness, the spurious response at the first harmonic of the sampling frequency is also shown.

Table I
Rectangular windowed digital filter — list of coefficients.

c_m		$a_i = c_{M-i} = c_{10} - \text{i}$
$c_0 =$	0.2500	$= a_{10}$
$c_{\pm1} =$	0.2251	$= a_9 = a_{11}$
$c_{\pm2} =$	0.1592	$= a_8 = a_{12}$
$c_{\pm3} =$	0.0750	$= a_7 = a_{13}$
$c_{\pm4} =$	0.0000	$= a_6 = a_{14}$
$c_{\pm5} =$	−0.0450	$= a_5 = a_{15}$
$c_{\pm6} =$	−0.0531	$= a_4 = a_{16}$
$c_{\pm7} =$	−0.0322	$= a_3 = a_{17}$
$c_{\pm8} =$	0.0000	$= a_2 = a_{18}$
$c_{\pm9} =$	0.0250	$= a_1 = a_{19}$
$c_{\pm10} =$	0.0318	$= a_0 = a_{20}$

If we could tolerate an unlimited number of delays; we could easily obtain the transfer function that would give us this response exactly. Using Eq. 24, for example, we could obtain the a_i values from the c_M values which, in turn, could be obtained directly from Eq. 15b.

We must, however, truncate the series described in Eq. 24 to a limit of 21 terms or 20 delays. Direct calculations of c_m using the value of unity for the desired filter response gives values for the c_m and a_i coefficients as shown in Table I.

This set of values would give a distorted frequency response with over- and undershoots of approximately 9%, as shown in Fig. 5b. We will now examine the effect of a few of the possible window functions that may be applied to our truncated, but unwindowed, transfer function.

Rectangular window

The rectangular window is simply the name given to the process of truncation where there is no modification of the amplitude of any of the nonzero impulse response pulses. It corresponds to a window in which the various nonzero pulses are retained at their original amplitudes while those pulses extending beyond the permissible number of delays are multiplied by zero. Although things seem very elementary at this point, it is interesting to examine the frequency response of this (rectangular) window to get some notion of how it distorts the desired frequency response through the process of convolution in the frequency domain.

The nature of the rectangular window may be considered to be a series of unity amplitude pulses separated by the interval $T = 1/f_s$ and numbering 21 as shown on Fig. 9. The Fourier transform of this window is

$$W(f) = \frac{\sin(\omega M)}{\sin(\omega/2)} = \frac{\sin(10\omega)}{\sin(\omega/2)} \qquad (28)$$

The amplitude (portion of the frequency) response of this window function, which is the absolute value of the above equation, is plotted in Fig. 10. The convolution of the frequency response of Fig. 10 with the specified ideal filter function response of our example is similar to that shown in Fig.

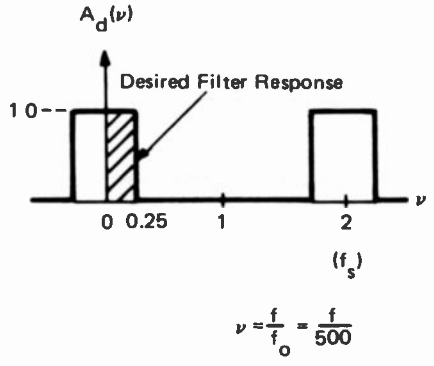


Fig. 8
Desired filter response for sample problem. Spurious response at the first harmonic of the sampling frequency is shown as well as the "negative frequency" portion of the response.

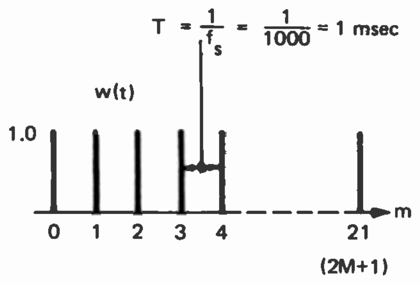


Fig. 9
Rectangular window function in the time domain. There are $2M$+1 pulses of unity amplitude, each separated by a time T from its nearest neighbor(s).

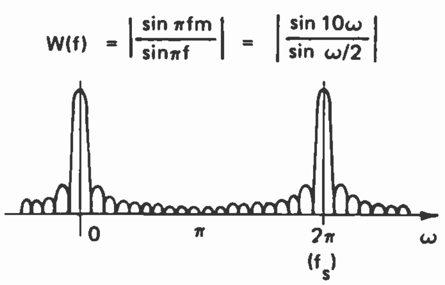


Fig. 10
Frequency response of rectangular window. The convolution of this frequency response with the specified ideal filter function response of the example is similar to that shown in Fig. 7.

7. the result, plotted on a decibel scale and normalized to the folding frequency, f_0, is shown on Fig. 11.

The transfer function of our rectangular windowed filter continues to be given by

$$H(z) = \sum_{i=0}^{2M} a_i z^{-i}$$

where $a_i = c_{M-i} = c_{10-i}$

The corresponding values for the a_i coefficients are those given in Table I.

The frequency response for our rectangular windowed digital filter may also be obtained directly from its transfer function by allowing z to go to $e^{j\omega}$, e.g.,

$$H(z) \rightarrow H(e^{j\pi\nu}) = \sum_{i=0}^{20} a_i e^{-ji\pi\nu}$$

$$= \sum_{i=0}^{20} a_i \cos i\pi\nu - j \sum_{i=0}^{20} a_i \sin i\pi\nu$$

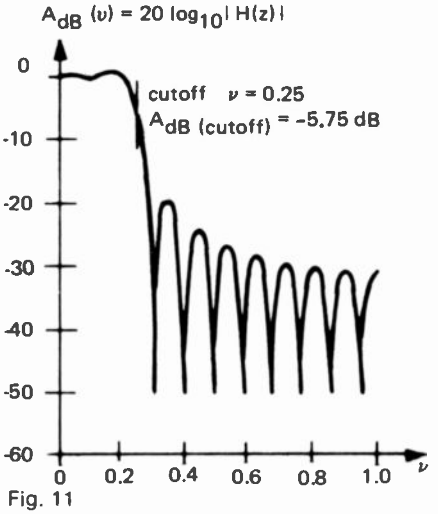


Fig. 11
Response of digital filter with rectangular window.
Note : In each case $a_i + c_{10-i}$ (see Table I).

and

$$|H(e^{j\omega\nu})|^2 = \left[\sum_{i=0}^{20} a_i \cos i\pi\nu\right]^2 + \left[\sum_{i=0}^{20} a_i \sin i\pi\nu\right]^2 \qquad (29)$$

The triangular window

More interesting results are obtained when a window which does modify the impulse coefficients is applied. The simplest window to work with is the triangular window shown on Fig. 12. The analytical expression for this window is

$$w_m = 1 - \frac{|m|}{M} = 1 - \frac{|m|}{10}$$

The resulting values for w_m are given in Table II. These values are then applied to the c_m impulse coefficients to obtain c'_m as in Eq. 25, and the final result is made causal by obtaining the a_i coefficients using the translation $a_i = c'_{M-i} = c_{10-i}$, as in Eq. 24. These values are also given in Table II.

The frequency response of the triangular window filter may be obtained directly from the transfer function as given in Eq. 29, above, after the proper a_i coefficients have been inserted.

It is interesting to note the extent to which the application of the triangular window has altered the frequency response of our FIR digital filter. For one thing, the pass region no longer has a ripple. The falloff, however, is more gradual, making for a wider transition region between passband and stopband. The stopband region, however, is very much better behaved and has very much reduced ripple variation as well as somewhat greater attenuation. A few other types of windows will now be considered.

Table II
Impulse coefficients for frequency response of triangular window.

m	c_m	w_m	c_m'	i
0	0.2500	1.0000	0.2500	10
±1	0.2251	0.9000	0.2026	9,11
±2	0.1592	0.8000	0.1273	8,12
±3	0.0750	0.7000	0.0525	7,13
±4	0.0000	0.6000	0.0000	6,14
±5	−0.0450	0.5000	−0.0225	5,15
±6	−0.0531	0.4000	−0.0212	4,16
±7	−0.0322	0.3000	−0.0096	3,17
±8	0.0000	0.2000	0.0000	2,18
±9	0.0250	0.1000	0.0025	1,19
±10	0.0318	0.0000	0.0000	0,20

Generalized Hamming window

The generalized Hamming window is given by the following expression:

$$w_m = \begin{cases} \alpha + (1-\alpha)\cos(\pi m/M) & |m| \leq M \\ 0 & \text{elsewhere} \end{cases} \qquad (30)$$

The value of the constant, α, is important in determining the result of the windowing action. There are two general classes depending upon the value assigned to this constant. Thus for $\alpha = 0.5$, we obtain the Hanning [not Hamming] window. In this case our expression appears as

$$w_m = \tfrac{1}{2} + \tfrac{1}{2}\cos(\pi m/M) = \cos^2(\pi m/2M) \qquad |m| \leq M \quad (31)$$

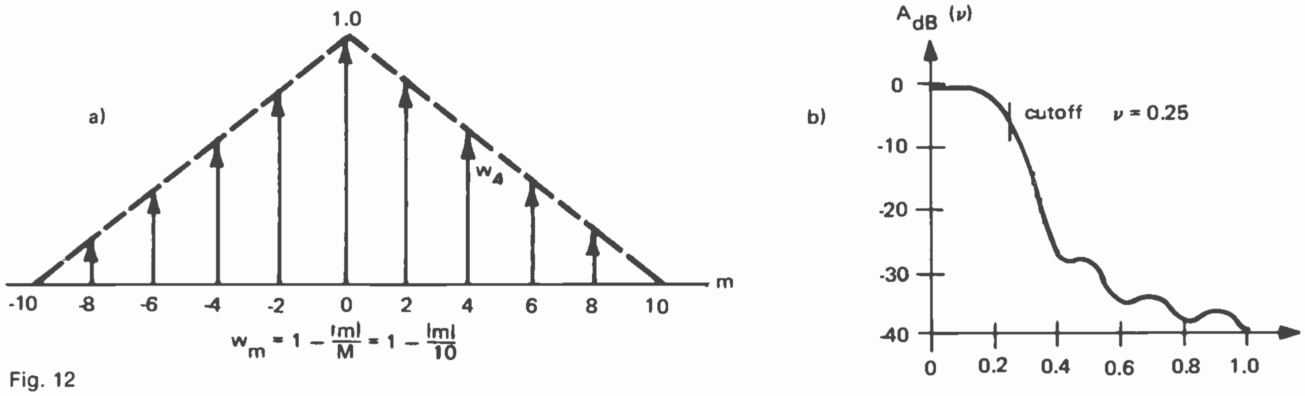


Fig. 12
Triangular window and the resulting frequency response when applied to our ideal filter. a) **Triangular window.** b) **Frequency response** of windowed ideal filter.

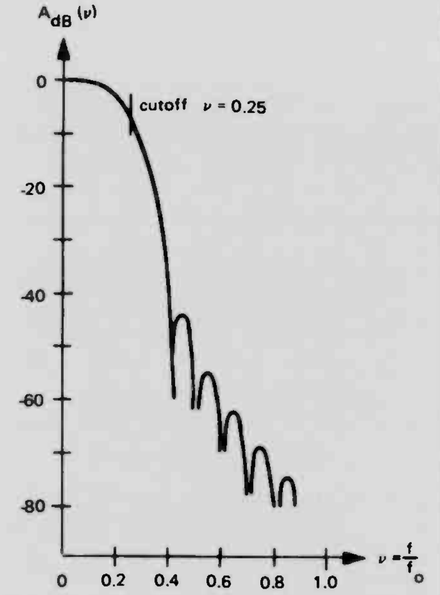


Fig. 13
Frequency response of ideal digital filter with Hanning window. Note that the fall-off is more gradual than that for the triangular window (Fig. 12b).

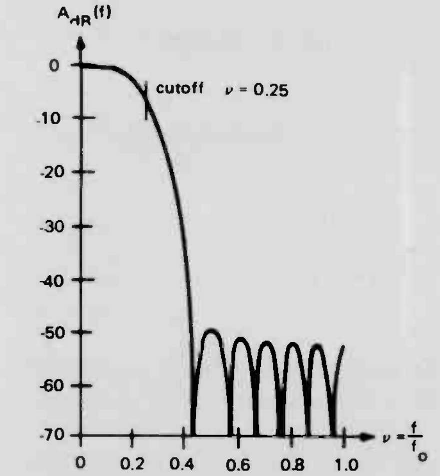


Fig. 14
Frequency response of digital filter with Hamming window. Although the irregularities in the stopband region are more pronounced, behavior is more uniform throughout the region.

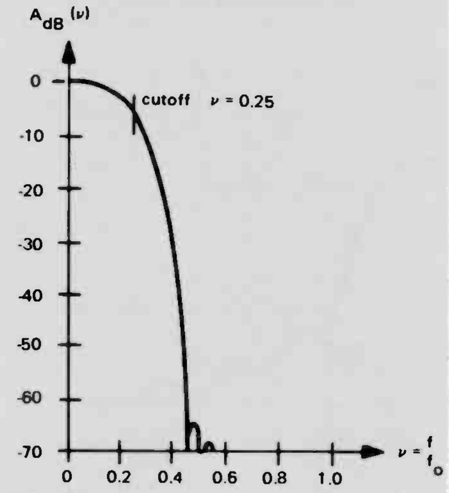


Fig. 15
Frequency response of digital filter with Kaiser window. Although the cut-off is still gradual, the stopband is highly attenuated.

The frequency response of the Hanning window application to our digital filter is shown on Fig. 13. We may note that the fall-off is somewhat more gradual than that obtained with the triangular window. The stop region, however, is more effectively attenuated.

The Hamming window is represented by the window function with a value of $\alpha = 0.54$.

$$w_m = \begin{cases} 0.54 + 0.46 \cos(\pi m / M) & |m| \leqslant M \\ 0 & \text{elsewhere} \end{cases} \tag{32}$$

The application of the Hamming window to our digital filter results in the frequency response shown in Fig. 14. We note that the "ripples" of the response in the stopband region are more pronounced, but also more regular.

Kaiser windows

Kaiser windows are represented by the following window function:

$$w_m = \frac{I_o\{\theta[1 - (m/M)^2]^{1/2}\}}{I_o\{\theta\}} \tag{33}$$

for $|m| \leqslant M$

This window function consists essentially of the ratio of two zero-order modified Bessel functions of the first kind.[f] In each case, the parameter θ enters into the argument giving us an entire family of such windows. The frequency response of our digital filter with the application of a Kaiser window where θ has a value of 2π is given on Fig. 15. Note that although the cut-off is quite gradual, the stopband has an exceedingly high attenuation.

The frequency-sampling approach

This design technique makes use of the discrete Fourier transform.

Hence, it would be of value to review some important aspects of this method. The usual Fourier transform is one in which a continuous periodic time function is expanded into a Fourier series which, in turn, is discrete and nonperiodic. Both time and frequency series are shown on Fig. 16 for the case of a square wave. The period between successive cycles of the periodic time function is given the notation t_p, while the frequency interval between successive frequency samples (or harmonics) is given the notation F. Fig. 16 gives the formulas for both transform and inverse transform as well as the important reciprocal relationship between t_p and F.

The discrete Fourier transform deals with time and frequency functions in which both are discrete and periodic, as shown on Fig. 17. In this case we have a new parameter, T, which represents the spacing between adjacent pulses along the time axis. The important relationships connecting T with both t_p and F are noted. The transform and inverse transform for the discrete case are

$$x(nT) = \frac{1}{N} \sum_{m=0}^{N-1} X(mF) e^{j2\pi mnFT} \tag{34a}$$

$$X(mF) = \sum_{n=0}^{N-1} x(nT) e^{-j2\pi mnFT} \tag{34b}$$

where $t_p = 1/F$; $f_s = 1/T$

An interesting aspect of the discrete Fourier transform is that time and frequency functions are periodic as well as discrete; perhaps surprisingly, however, the number of pulses (or terms) per cycle in each case, is the same, namely N.

The Z-transform enters

We will begin with our specification for the FIR digital filter, which may be stated as

$$H(m) = A(m) e^{j\beta(m)} \tag{35}$$

for $0 \leqslant m / \leqslant N-1$, where m is the index for the term in the frequency domain. The actual frequencies are, of course, mF (see Eq. 38 below).

In the z formulation, the desired frequency response is represented with z as the variable

$$H(m) \rightarrow H(z) = \sum_{n=0}^{N-1} a_n z^{-n} \tag{36}$$

Upon replacing z with $e^{j\omega}$ to obtain the frequency response of our transfer function, we obtain

$$H(z) \rightarrow H(e^{j\omega T}) = \sum_{n=0}^{N-1} a_n e^{-j\omega nT} \tag{37}$$

We now recognize that the various terms in the summation represent corresponding points along the frequency axis in the

[f] See Pipes, L.A. and Harvill, L.R.: *Applied Mathematics for Engineers and Physicists* (McGraw-Hill; 1970)p. 793-4.

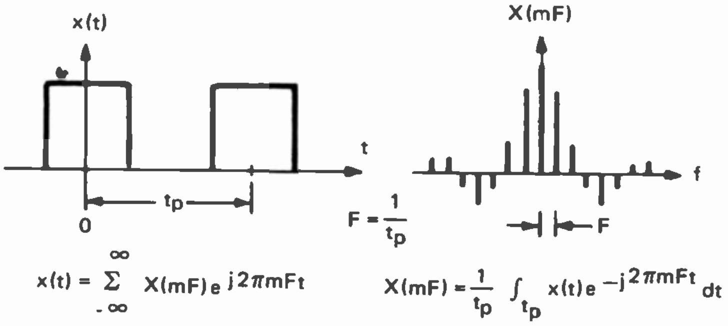

x(t)
t
tp
0
x(t) = Σ X(mF) e j2πmFt (∞, -∞)
X(mF)
f
F = 1/tp
F
X(mF) = 1/tp ∫tp x(t) e -j2πmFt dt

Fig. 16
Common form of the Fourier transform is the Fourier series in which a continuous periodic time function is expanded into a discrete nonperiodic frequency function.

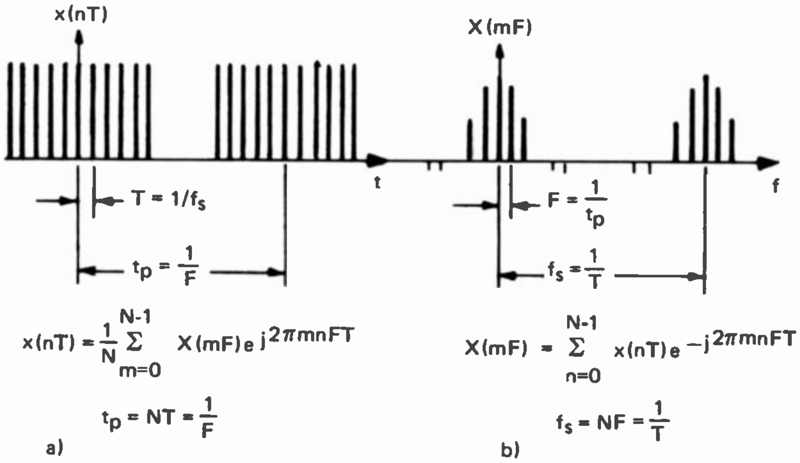

x(nT)
t
T = 1/fs
tp = 1/F
x(nT) = 1/N Σ X(mF) e j2πmnFT (N-1, m=0)
tp = NT = 1/F
a)
X(mF)
f
F = 1/tp
fs = 1/T
X(mF) = Σ x(nT) e -j2πmnFT (N-1, n=0)
fs = NF = 1/T
b)

Fig. 17
The discrete Fourier transform. Both time and freqency functions are periodic and discrete. a) **Discrete periodic time function.** b) **Discrete periodic frequency function.**

frequency domain and that these points may be specified by the mF product. Upon making this substitution for the frequency, we obtain

$$H(e^{j\omega T}) = \sum_{n=0}^{N-1} a_n e^{-j2\pi mnFT} \quad (38)$$

Comparison of Eq. 38 with Eq. 34b immediately identifies our constant a_n (in Eq. 38) as equivalent to $x(nT)$ of Eq. 34b. This enables us to evaluate a_n using the discrete Fourier transform

$$a_n = \frac{1}{N} \sum_{m=0}^{N-1} H(mF) \; e^{j2\pi mnFT} \quad (39)$$

We now make matters a bit more convenient by simplifying the argument $x(nT)$ to $x(n)$ since T is a constant and does not contribute to the identification of the n^{th} term. Likewise we shorten the argument of $H(mF)$ to $H(m)$ since here, likewise, the constant F does not contribute to the identification of the n^{th} frequency term. We make the substitution indicated below and obtain expressions containing the commonly used notation W_n.

$$FT = 1/N \quad (\text{for } F \rightarrow 1.0)$$

$$W_N = e^{-j2\pi/N}$$

$$H(e^{j\omega}) = \sum_{n=0}^{N-1} a_n W_N^{mn} \quad (40)$$

$$a_n = \frac{1}{N} \sum H(m) W_N^{-mn}$$

We make the substitution for a_n in the expression for our transfer function in Eq. 33. This gives

$$H(z) = \sum_{n=0}^{N-1} a_n z^{-n} \quad (41)$$

$$= \sum_{n=0}^{N-1} [\frac{1}{N} \sum H(m) \, W_N^{-mn}] z^{-n}$$

$$= \frac{1}{N} \sum_{m=0}^{N-1} H(m) \sum_{n=0}^{N-1} W_N^{-nm} \, z^{-n}$$

It turns out that the quantity under the second summation sign in Eq. 41 may be represented in closed form as follows (where we use the identity $W^{-Nm} = e^{(-j2\pi/N)(-Nm)} = 1$):[g]

$$\sum_{n=0}^{N-1} W^{-nm} z^{-n} = \frac{1 - z^{-N}}{1 - W^{-m} z^{-1}} \quad (42)$$

The final result is

$$H(z) = \frac{1}{N} \sum_{m=0}^{N-1} H(m) \frac{1 - z^{-N}}{1 - W^{-m} z^{-1}}, \quad (43)$$

which may be rewritten as

$$H(z) = \frac{1 - z^{-N}}{N} \sum_{m=0}^{N-1} \frac{H(m)}{1 - W^{-m} z^{-1}} \quad (44)$$

It can be shown rigorously that the response of the above transfer function reduces exactly to $H(m)$ at the frequency values mF.[h] This allows us to specify the response at these frequencies by appropriate choice of the $H(m)$ values. The behavior of $H(z)$ at the in-between values, however, is another matter entirely and will be studied in the following section dealing with the realization of this filter.

Realization of the filter

The digital filter represented by the transfer function of Eq. 44 may be realized either recursively (using the approach of Fig. 1, for example) or by the discrete Fourier transform. In the latter case, an algorithm for speedy calculation of the discrete Fourier transform is developed and applied directly to the input signal. We then operate directly on the resulting frequency components. The final step is to transform the result back to the time domain to obtain the filter output as a series of pulses along the time axis. In this article, we will discuss the recursive realization of the transfer function of Eq. 44.

Inspection of Eq. 44 reveals that it contains N zeros and N poles. The zeros occur in the coefficient multiplying the summation while the poles occur, singly, in each of the terms of the summation. We will consider the zeros first.

The zeros are obtained by setting the coefficient of the summation equal to zero. By this process we obtain, not only the zeros, but the effect of this portion of the transfer function on the frequency

[g] See Cadzow, J.A.; *Discrete Time Systems* (Prentice Hall, Inc., 1973) p. 26-7.

[h] The solution to this problem is given at the end of this article.

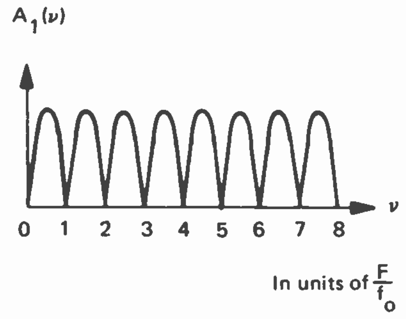


Fig. 18
Effect of Eq. 45 is that of a comb filter.

response. Our final result below is obtained after a modest amount of manipulation:

$$\frac{1 - Z^{-N}}{N} = 0 = \frac{1 - e^{-j\pi\nu N}}{N}$$

$$0 = \frac{2j}{N} e^{-j\pi\nu N/2} \sin(\pi\nu N/2)$$

$$0 = \frac{2}{N} \sin \frac{\pi\nu N}{2} \angle -\frac{\pi\nu N}{2} + \frac{\pi}{2}$$

or

$$A_1(\nu) = \frac{2}{N} \sin \frac{\pi\nu N}{2} \tag{45}$$

Zeros occur at $\nu = 2m/N$, for m=0,1,2, $\cdots$, $N - 1$, which may be rewritten

$$\nu = \frac{2mF}{NF} = \frac{mF}{f_o}$$

The important quantity in the above development is the expression which we have identified as $A_1(\nu)$. Since this is an absolute magnitude, it is always positive and may be plotted as shown on Fig. 18. Such a response corresponds to that of a comb filter and is fairly common in work of this type.

An interesting situation occurs with the summation portion of the transfer function of Eq. 44. Here, each term, individually, has it own pole. These poles occur at

$$z = W^{-m} = e^{j(2\pi m/N)}$$

but

$$\frac{2\pi}{N} m = \frac{\pi mF}{NF/2} = \frac{\pi f_m}{f_o} = \pi\nu_m$$

and also:

$$z = e^{j\omega T} = e^{j\pi\nu}$$

Hence

$$e^{j\pi\nu} = e^{j\pi\nu_m}$$

and poles occur at $\nu = \nu_m = mF/f_o$

$$\tag{46}$$

Comparing the location of the zeros of Eq. 45 and the poles of Eq. 46 shows that they occur at exactly the same frequency positions mF. The poles, however, occur singly in the individual terms of the summation. This corresponds to separate outputs from the system—one for each value of m. Consequently, we have a situation wherein all of the zeros in the coefficient of the summation are combined successively with each of the individual poles. The result of one such operation is shown on Fig. 19 for the case of the $m = 0$ pole. From an analytical standpoint, the pole in this term cancels the zero at the same ($m = 0$) frequency with the rigorous result that the final response at $m = 0$ becomes exactly equal to the $H(m)$ coefficient for $m = 0$. Descriptively speaking, the comb develops a "hump" at $m = 0$.

If all of the outputs of this filter, corresponding to all of the terms in the summation, are summed into a single output, the resulting frequency response at the various mF frequencies corresponds exactly to the specified $H(m)$ coefficients but tends to move in a ripple-like fashion between these points. The problem then resolves itself to an adjustment of $H(m)$ coefficient values, or a change in the number of terms, to reduce these ripples to acceptable values. This represents the basic design problem which has to be solved in this method.

The physical realization of the filter may be shown as in Fig. 20. The coefficient of the summation is shown on the left, where it affects the response for all values of frequency. The various terms in the summation are shown as separate outputs on the right. Each output is adjusted in amplitude by the value of its particular $H(m)$ coefficient and then modified by the recursive (feedback) circuit shown which introduces the pole.

The various outputs may then be utilized singly or summed as a group to obtain the frequency response of the filter system.

The realization of Fig. 20 is recursive because of the feedback loops on the right-hand side of the diagram. Hence, there may be some stability problems connected with this realization. These problems however,

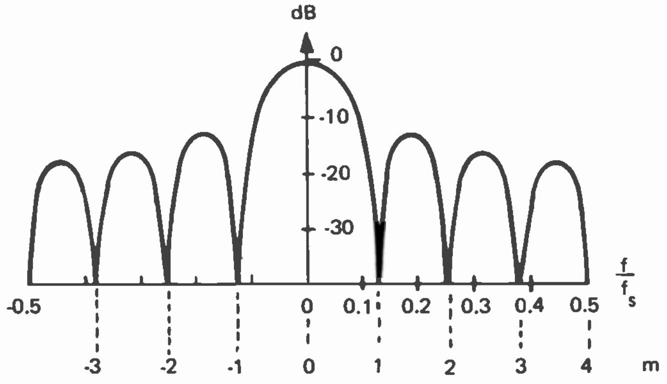
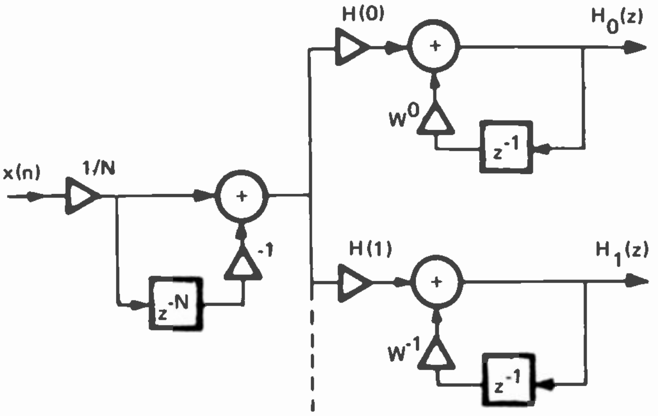


Fig. 19
In Eq. 44, the m = 0 term cancels the zero at the same value of frequency and leads to a frequency response exactly equal to the desired value of $H(m)$ [= H (0)] at that frequency.



$$H_m(z) = \frac{1 - z^{-N}}{N} H(m) \frac{1}{1 - z^{-1} W^{-m}}$$

Fig. 20
Filter bank recursive realization of FIR filter. The final output is the summation of the various $H(z)$ outputs.

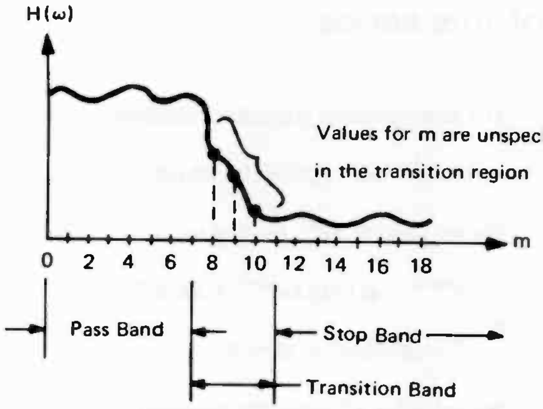


Fig. 21
Basis for the optimization method. The frequency response in the transition region is unspecified so that the frequency [amplitude $A(f)$] values in this region may be varied to minimize ripple excursions in the pass- and/or stop-bands.

are not usually severe. On the other hand, in certain cases, a good many of the $H(m)$ coefficients are zero, thus reducing the number of circuits involved. This can be important since each circuit involves a time delay (memory). Thus, it is not possible to generalize as to whether a recursive or discrete Fourier transform realization of a particular FIR filter is more advantageous. Each case requires separate study.

As a final note, the recursive design approach outlined above does not preclude the additional use of windowing techniques to modify the response shown on Fig. 19 to a more acceptable configuration. An additional technique which is sometimes used involves developing a weighted average of frequency values about the frequency point being examined rather than an unquestioning acceptance of the value at that particular point.

Optimization techniques

The basis for this method is shown on Fig. 21. We assume that we have an initial design for a digital filter in which the amplitude response consists of a passband, a stopband, and a transition band. To minimize the spurious frequency response excursions in the passband and, perhaps, establish a maximum acceptable response in the stopband, we manipulate the frequency response in the transition band, where this response is considered to be of no concern. In brief, then, the problem resolves itself into developing the overall desired frequency response, $H(j\omega)$, by varying the response in the transition band.

A fairly extended treatment of this approach is given in Chapter 3 of Rabiner and Gold,[2] in which various aspects of the design are considered, including determination of the minimum number of terms required to obtain a given response, as well as design based on a simultaneous time-response constraint. A fairly sophisticated level of mathematics is involved, with one of the major approaches being linear programming. Readers interested in this aspect of digital filter design may do well to study linear programming since it provides a good introduction into the optimization process in the presence of constraints.

The linear programming approach was developed to a large extent by mathematicians and economists who found this technique valuable in problems such as maximizing profit under certain production, warehousing, and transportation constraints. I recommend any of the three sources listed below for a good, readable introduction to this technique.

> *Scientific Programming in Business and Industry*, Andrew Vazsonyi, (Wiley, 1958)
>
> *Economic Theory and Operations Analysis*, William J. Baumol, (Prentice-Hall, 1972).
>
> *Operations Research*, Hillier and Lieberman (Holden-Day, Inc., 1974)

A more general approach into the various more classical methods for obtaining optimization under constraints may be found in:

> *Foundations of Optimization*, D.J. Wilde and C.S. Beightler (Prentice-Hall, 1967)

Additional readings in the optimal design of digital filters, including references, may be found in *Digital Filtering and Signal Processing*, Donald Childers and Allen Durling (West Publishing Co., 1975) and the paper by G.C. Brown in *Introduction to Digital Filtering*, edited by R.E. Bogner and A.G. Constantinides (John Wiley & Sons, 1975). However, in all fairness, entrance into this method of digital filter design requires a fairly high degree of mathematical maturity on the part of the circuit designer.

References

1. Stanley, W.D.; *Digital signal processing* (Reston Publishing Co., Inc.; 1975)
2. Rabiner, L.R. and Gold, B.; *Theory and application of digital signal processing* (Prentice-Hall; 1975)
3. Oppenheim, A.V. and Schaefer, R.W.; *Digital signal processing* (Prentice-Hall; 1975)

Additional useful discussions of FIR digital filter design may be found in

4. Childers, D. and Durling, A.; *Digital filtering and signal processing* (West Publishing Co.; 1975)
5. Bogner, R.E. and Constantinides, A.G. (Editors); *Introduction to digital filtering* (John Wiley and Sons; 1975)

Reprints of key papers dealing with FIR digital filter design may be found in the following collections:

6. Liu (Editor); *Digital filters and the fast Fourier transform* (Halstead Publishing Co.; 1975) This reference contains papers on the optimization approach.
7. Rabiner and Rader (Editors); *Digital signal processing* (IEEE Press; 1972)
8. Selected papers in *Digital Signal Processing*, edited by the Digital Signal Processing Committee, IEEE Acoustics, Speech, and Signal Processing Society. (IEEE Press 1975) This reference contains papers on optimization techniques.

A listing of available literature in the overall field of digital signal processing may be found in *Literature in Digital Signal Processing*, edited by Helms, Kaiser, and Rabiner (IEEE Press, 1975).

A final problem to chew on

In connection with our key result for the frequency sampling approach, Eq. 45, we emphasized that this transfer function will give us exactly the desired responses at the frequencies defined by mF, where these responses are defined by the $H(m)$ coefficients. Prove this assertion rigorously by making the proper substitutions in Eq. 45 and applying L'Hospital's rule. See if you can do it—it might really be fun!

Answer to review question from Part 3 of this series

We are given

$$y(k) = u(k) + c(k) + i(k) \tag{1s}$$

We reduce all quantities on the right to either a constant or to functions of y as follows:

Given,

$$u(k) = u_0$$
$$c(k) = ay(k-1) \tag{2s}$$
$$i(k) = b[c(k) - c(k-1)] \tag{3s}$$

We rewrite Eq. (2s) as

$$c(k-1) = ay(k-2),$$

and substitute in Eq. (3s) to obtain

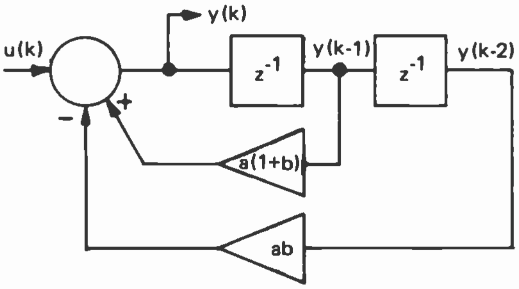
$$i(k) = b[ay(k-1) - ay(k-2)].$$

We now rewrite Eq. (1s) as

$$y(k) = u_0 + ay(k-1) + b\,[ay(k-1) - ay(k-2)]$$

or

$$y(k) = u_0 + a(1+b)y(k-1) - aby(k-2) \tag{4s}$$



In this model of the national economy, z^{-1} represents a delay of one time period, e.g., three months.

Referring to the above model, we note that the circular summation symbol has three inputs and one output (from which $y(k)$ is obtained). These three inputs consist of

$u(k)$, which is constant and corresponds to the first term on the right of our final equation,

$y(k-1)$ multiplied by $a(1+b)$, which is added to $u(k)$ and corresponds to the second term on the right,

$y(k-2)$, which is multiplied by ab and then subtracted from the above two terms and corresponds to the third term on the right.

$y(k)$ is obtained at that point in the model where all three of these inputs have been summed into a single signal.

We easily solve Eq. (4s) using classical methods. We begin by rewriting the equation as

$$y(k+2) - a(1+b)y(k+1) + aby(k) = u_0 \tag{5s}$$

The homogeneous (transient) solution is first obtained

$$y(k+2) - a(1+b)y(k+1) + aby(k) = 0$$

We let $y_{tr}(k) = Ae^{mt}$, leading to

$$Ae^{m(t+2)} - a(1+b)Ae^{m(t+1)} + abAe^{mt} = 0$$

or

$$e^{2m} - a(1+b)e^{m} + ab = 0$$

We solve for e^m using the binomial theorem

$$e^m = \frac{a(1+b)}{2} \pm \sqrt{\frac{a^2(1+b)^2}{4} - ab} \tag{6s}$$

We make our calculations more convenient by using the notation

$$e^m = \alpha + \gamma$$

It turns out that for reasonable values of a and b, both α and γ will be real with $\alpha > \gamma$ and with $0 < (\alpha + \gamma) < 1.0$

Our transient solution is

$$y_{tr}(k) = A\,(\alpha + \gamma)^t + B(\alpha - \gamma)^t$$

The inhomogeneous (steady state) solution is easily obtained by noting that the right hand side of Eq. (5s) is a constant. Hence, $y_{ss}(k)$ = constant for all k. We can then write

$$y_{ss} - a(1+b)y_{ss} + aby_{ss} = u_0$$

$$y_{ss} = \frac{u_o}{1 - a(1+b) + ab} = \frac{u_o}{1-a} \tag{6s}$$

Our complete solution is

$$y(k) = y_{tr}(k) + y_{ss}(k)$$

$$y(k) = A(\alpha + \gamma)^t + B\,(\alpha - \gamma)^t + u_o/(1-a) \tag{7s}$$

Our two arbitrary constants are evaluated by given or known values of $y(0)$ and $y(1)$, e.g.,

$$y_o = A + B + u_o/(1 - a)$$
$$y_1 = A(\alpha + \gamma) + B(\alpha - \gamma) + u_0/(1-a)$$

Solving this pair of equations, we obtain

$$A = \frac{y_1 - (\alpha - \gamma)y_o - [u_o/(1 - a)](1 - \alpha+\gamma)}{2\gamma} \tag{8s}$$

$$B = \frac{y_1 - (\alpha+\gamma)\,y_o - [u_o/(1-a)](1-\alpha-\gamma)}{-2\gamma}$$

Although it may not be immediately evident, and although it would require a certain amount of tedious calculations, the formal solution [Eq. (7s)] does in fact reduce to y_o and y_1 (on the right hand side) when values of $t = 0$ and $t=1$ are substituted. (t may be used interchangeably with k in this problem.)

An examination of Eq. (7s) reveals that the ultimate steady state of the national income depends entirely upon the term $u_o/(1-a)$ e.g., the ratio of government expenditures to the fraction of the national income which is *saved* $(1 - a)$ by the consumer. The moral of this story is to get the public to spend as much as possible of their income (do not save!) and work on congress to maximize government expenditures.

The violence of the transient fluctuation of the national income prior to achievement of the steady state depends primarily on the rate of consumer expenditures and induced investments.

Caution! This is a very simplified model and the results should not be taken at their face value. For example, the model does not predict economic cycles, inflation, recession, etc., etc.

Answer to the review question in this article

Our transfer function may be written as

$$H(z) = \frac{1 - z^{-N}}{N} \sum_{m=0}^{N-1} \frac{H(m)}{1 - z^{-1}\, e^{j2\pi m/N}} \qquad (1s)$$

where we have used the equivalence

$$W_N = e^{-j2\pi/N}$$

To obtain the frequency response of $H(z)$, we let

$$z \rightarrow e^{jwT} = e^{j2\pi f/2f_o} = e^{j\pi\nu}$$

and obtain

$$H(e^{j\pi\nu}) = \frac{1 - e^{-j\pi\nu N}}{N} \sum_{m=0}^{N-1} \frac{H(m)}{1 - e^{-j\pi\nu}\, e^{j2\pi m/N}}$$

We note that the frequency values mF may be written, in our normalized system, as

$$mF \rightarrow mF/f_o = \nu_m$$

but $f_s = NF$ and $f_o = ½\ NF$

hence,

$$\nu_m = \frac{mF}{½\ NF} = \frac{2m}{N}$$

Our zeros occur at exactly the mF, or ν_m, frequencies, e.g.,

$$\frac{1 - e^{-j\pi\nu N}}{N} \rightarrow \frac{1 - e^{-j\pi(2m/N)N}}{N} = \frac{1 - e^{-j2\pi m}}{N} = \frac{1 - 1}{N} = 0$$

since m is an integer having values from zero to $N - 1$. Hence a zero exists for each value of m.

Consider the N terms of the summation. Here, again, poles occur at the ν_m frequencies. We recognize that

$$e^{j2\pi m/N} = e^{j\pi\nu_m}$$

and write our summation as

$$\sum_{m=0}^{N-1} \frac{H(m)}{1 - e^{-j\pi[\nu - \nu_m]}}$$

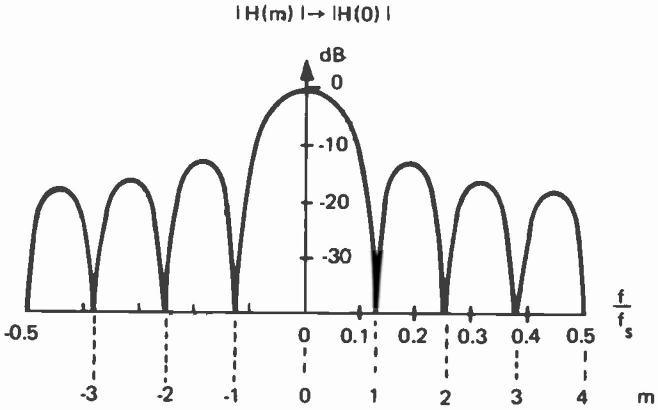


Fig. 1s
Response of the m=0 term in transfer function of Eq. 1s.

giving us a pole for the m^{th} *term* at $\nu=\nu_m$. (This works for no other value of m for that particular term since $\nu_m = (2m/N)$ and $\nu_m - \nu$ cannot reach the value of 2; the greatest value of m is $N - 1$.)

Hence, the product of our zero-producing coefficient and the N terms of the summation causes all terms to vanish at the ν_m frequencies *except the m^{th} term* where a zero and pole exist simultaneously and the value of the transfer function is indeterminate, e.g.,

$$\frac{1 - e^{-j\pi\nu N}}{N} \; \frac{H(m)}{1 - e^{-j\pi[\nu - \nu_m]}} \rightarrow \frac{0}{0}$$

We differentiate numerator and denominator with respect to ν and evaluate at $\nu = \nu_m$.

$$\left. \frac{j\pi N\, e^{-j\pi\nu N}}{N} \; \frac{H(m)}{j\pi e^{-j\pi[\nu - \nu_m]}} \right|_{\nu \rightarrow \nu_m} = H(m)$$

Hence, for each value of mF, the transfer function exactly equals $H(m)$. However, we should still consider the response of each term at other frequencies. These responses all have the same configuration. The response for the $m = 0$ term is shown in Fig. 1s.

We see that the response consists of a central region and sidelobes. The zeros, however, occur precisely at those points where the other $H(m)$ terms in the summation are centered. Hence, at exactly the mF frequencies only the m^{th} term contributes and we can rigorously say that the transfer function of Eq. 1s has exactly the value of $H(m)$ at that frequency.

Pen and Podium

Recent RCA technical papers and presentations

To obtain copies of papers, check your library or contact the author or his divisional Technical Publications Administrator (listed on back cover) for a reprint. For additional assistance in locating RCA technical literature, contact RCA Technical Communications, Bldg. 204-2, Cherry Hill, N.J., extension PY-4256.

Advanced Technology Laboratory

W.A. Clapp| A. Feller
G. Hrivnak| R. Reitmeyer
Future trends of microprocessors and microcomputers—Microcomputer Seminar, Assoc. of the US Army, Ft. Monmouth, NJ (11/10/76)

A. Feller| R. Noto
Low-cost, quick-turnaround random-logic custom LSI devices using automatic placement and routing program—1976 Government Microcircuit Conf., Orlando, FL (11/9-11/76)

P.F. Joy| B. Shelpuk
Solar heating thermal storage feasibility—*Proc.* ASME Winter Meeting, NYC, NY (12/3/76)

Automated Systems

T. Kupfrian
Commercial integrated circuits in selected military application—IEEE Boston Section Reliability Chapter, RCA, Burlington, MA (1/12/77)

W.S. Radcliffe
Laser beam divergence in short interferometers—*Optical Engineering* (1/77)

E.M. Stockton
Management problems—identification and correction—IEEE Management Meeting, Boston, MA (1/11/77)

Government Communications Systems

H. Barton
Spares allocation for cost-effective availability achievement—*Spectrum,* Soc. Logistics Engineers (Winter 76/77)

K. Bodzioch| B. Patrusky
Evaluation performance of a unified node with simulated network—Bicen. Winter Simulation Conf., MD (12/6-8/76)

J. Guzy| P. Boehm| R. Paetzold
Communications system simulation for a network displaying circuit switching, message switching—Bicen. Winter Simulation Conf., MD (12/6-8/76)

Government Engineering

W.W. Thomas
Industry views of government specifications and standards—Defense Specifications Management Course, Army Logistics Management Center, Ft. Lee, VA (11/16/76)

Laboratories

B. Abeles| J.I. Gittleman
Composite material films: optical properties and applications—*Appl. Optics,* Vol. 15, No. 10 (10/76) p. 2328

M.S. Abrahams| C.J. Buiocchi| R.T. Smith
J.F. Corboy| J. Blanc| G.W. Cullen
Early growth of silicon on sapphire-I: transmission electron microscopy—6th European Congress on Electron Microscopy, Jerusalem (9/76) and *J. Appl. Physics,* Vol. 47, No. 12 (12/76) pp. 5139-5150

G. W. Beakley
Television to small earth stations—*Trans. on Broadcasting,* Vol. 22, No. 3 (9/76) p. 96-100

A. Bloom| R.A. Bartolini
P.L.K. Hung| D.L. Ross
A non-polymeric organic host for recording volume phase holograms—*Appl. Phys. Lett.,* Vol. 29, No. 8 (10/15/76) p. 483

C.R. Carlson
Two-dimensional contrast sensitivity measurements with sinusoidal luminance gratings—ARVO, Sarasota, FL (4/26/76)

R.S. Crandall
Properties of surface state electrons on liquid helium—*Surface Science,* Vol. 58 (1976) p. 266

M. Ettenberg
Stress and strain in heteroepitaxial layers—Materials Research Soc. Mtg. (11/76)

M. Ettenberg| H. Kressel| J.P. Wittke
Very high radiance edge-emitting LED—*IEEE J. Quantum Electronics,* Vol. QE-12, No. 6 (6/76) pp. 360-364

M. Ettenberg| H. Kressel
Very-low-threshold double-heterojunction AlGaAs/GaAs laser diodes: theory and experiment—*J. Appl. Phys.,* Vol. 47, No. 8 (Aug 1976) pp. 3533-3537

M. Ettenberg| G.H. Olsen| C.J. Nuese
The effect of gas phase stoichiometry on the minority carrier diffusion—*Appl. Phys. Lett.,* Vol. 29, No. 3 (8/76) pp. 141-142

B. Goldstein| D.J. Szostak| V.S. Ban
Langmuir evaporation from the (100), (111A), and (111B) faces of GaAs—*Surface Science, Vol. 57 (1976) p. 733*

L.A. Goodman| D. Meyerhofer
S. DiGiovanni
The effect of surface orientation on the operation of multiplexed twisted-nematic devices—*IEEE Trans. Electron Devices,* Vol. 23, No. 10 (10/76) p. 1176

P.D. Griffis| J. Shefer
Kinescope spot size as it relates to picture resolution—*Trans.* IEEE Fall Conf. on Consumer Electronics, Vol. CE-23, No. 1, Chicago, IL (2/77)

W.E. Ham
The measurement and interpretation of the electrical properties of silicon on sapphire—*Extended Abstracts,* Vol. 76-2, Electrochemical Soc., Las Vegas, NV (10/17-22/76) pp. 462-4

W.E. Ham| S. Eaton
Anomalous electrical gate conduction in self-aligned MDS structures—*Proc.* IEDM, Washington, DC (12/76) pp. 323-327

E.W. Herold
A history of color television display—*Proc. IEEE,* Vol. 64, No. 9 (9/76) p. 1331

A.C. Ipri| D. W. Flatley
Radiation tolerant silicon gate CMOS/SOS using ion implantation—*IEEE Trans. Electron Devices* (9/76) p. 1110

R.U. Martinelli
The temperature dependence of the dc base and collector currents in silicon bipolar transistors—*Trans. Electron Devices,* Vol. 23, No. 11 (11/76) p. 1218

J. H. McCusker| S.S. Perlman| H.S. Veloric
Microsonic pulse filters—replacements for traditional Buterworth designs—*RCA Review,* Vol. 37, No. 3 (9/76) p. 389

D. Meyerhofer
A new technique of aligning liquid crystals on surfaces—*Appl. Phys. Lett.,* Vol. 29, No. 11 (12/1/76) p. 691

D.O. North
Theory of modal character, field structure, and losses for semiconductor laser—*IEEE J. Quantum Electronics,* Vol. 12, No. 10 (10/76) p. 616

J.I. Pankove| D.E. Carlson
Electroluminescence in amorphous silicon—*Appl. Phys. Lett.,* Vol. 29, No. 9 (11/1/76) p. 620

J. I. Pankove
Phenomena useful for displays—*Tech. Digest* 1976 IEDM, Washington, DC (12/8/76) p. 621

A. Pierrefeu| B. Dorner| E.F. Steigmeier
Inelastic neutron scattering in SbSI near the ferroelectric phase transformation—*Ferroelectrics, Vol.* 12 (1976) p. 125

R.J. Powell
Photoconductive processes in Al_2O_3 films—*J. Appl. Phys.,* Vol. 47, No. 3 (10/76) p. 4604

W. Rehwald
Ultrasonic studies of ferroelectric phase transitions—*Ferroelectrics*, Vol. 12 (1976) p. 105

K.M. Schleiser| J.M. Shaw| C.W. Benyon, Jr.
Al_2O_3 as a radiation-tolerant CMOS dielectric—*RCA Review*, Vol. 37, No. 3 (9/76) p. 359

P. Sheng
Phase transitions in surface-aligned nematic films—*Phys. Rev. Lett.*, Vol. 37, No. 16 (10/18/76) p. 1059

P. Sheng| P.J. Wojtowicz
Constant coupling theory of nematic liquid crystals—*Phys. Review A*, Vol. 14, No. 5 (11/76) p. 1883

E.K. Sichel| R.E. Miller
Sputtering of reactive metals for composite materials: erbium and Al_2O_3—*Thin Solid Films*, Vol. 37 (1976) p. L19

E.F. Steigmeier| R. Loudon
G. Harbeke| H. Auderset
Raman scattering in $K_2Pt(C)_4Br_{0.3}V\ 3H_2O$—*Ferroelectrics*, Vol. 13 (1976) p. 549

L.C. Upadhyayula
Trigger sensitivity of transferred electron logic devices—*IEEE Trans. Electron Devices*, Vol. 23, No. 9 (9/76) p. 1049

A.E. Widmer| R.Fehlmann| H.P. Kleinknecht
Liquid-phase epitaxial growth of multiple (AlGa)P-GaP heterojunction structures—*J. Crystal Growth*, Vol. 35, No. 1 (8/76) p. 89

C.R. Wronski| D.E. Carlson| R.E. Daniel
Schottky barrier characteristics of metal-amorphous silicon diodes—*Appl. Phys. Lett.*, Vol. 29, No. 9 (11/1/76) p. 602

C.P. Wu| E.C. Douglas| C.W. Mueller
Redistribution of ion-implanted inpurities in silicon during diffusion in oxidizing ambients—*IEEE Trans. Electron Devices*, Vol. ED-23, No. 9 (9/76) p. 1095

Missile and Surface Radar

M.W. Buckley
Project management—planning scheduling and control—Co-chairman, AMA Seminar, Atlanta, GA (12/12-16/76)

R.W. Howery
AEGIS combat system computer program management approach—Software Management Conf., Washington, DC (1/27-18/77)

S.M. Sherman| J.C. Pracklin
Tests of complex-angle monopulse in the TRADEX radar—DARPA Low-Angle Tracking Symp., Washington, DC (12/1-2/76)

Dates and Deadlines

Upcoming meetings

Ed. Note: Meetings are listed chronologically. Listed after the meeting title (in bold type) are the sponsor(s), the location, and the person to contact for more information.

APR 23-28, 1977—**American Ceramic Soc., Electronics Div., 179th Annual Mtg. & Exposition,** Conrad Hilton Hotel, Chicago, IL **Prog Info:** Dr. Richard M. Rosenberg, E.I. duPont de Nemours & Co., Inc., Photo Products, Bldg. 428, Buffalo Ave., Niagara Falls, NY 14302

APR 25-27, 1977—**Circuits & Systems Intl. Symp.** (IEEE) Del Webb's Towne House, Phoenix, AZ **Prog Info:** W.G. Howard, Motorola Integrated Circuits Center, Mail Stop MR, POB 20906, Phoenix, AZ 85036

MAY 2-5, 1977—**Offshore Technology Conf.** (IEEE et al) Astrodome, Albert Thomas Convention Ctr., Houston, TX **Prog Info:** OTC, 6200 N. Central Expressway, Dallas, TX 75206

MAY 3-6, 1977 —**EUROCON 77** (IEEE et al) Venice, Italy **Prog Info:** Alberto Vandind Buti, c/o AEI, Viale Monza 259, 20126 Milan, Italy

MAY 9-11, 1977—**Acoustics, Speech and Signal Processing Intl.** (ASSP) Sheraton Hartford, Hartford, CT **Prog Info:** Harvey F. Silverman, Thomas J. Watson Research Center, POB 218, Yorktown Hts., NY 10598

MAY 16-18, 1977 —**Electronic Components Conf.** (IEEE) Stouffer's Natl. Center Inn, Arlington, VA **Prog Info:** Charles M. Tapp, Sandia Labs, Dept. 2150, Albuquerque, NM 87115

MAY 17-19, 1977 — **NAECON—National Aerospace & Electronics Conf.** (AES, IEEE, et al) Dayton Conv. Ctr., Dayton, OH **Prog Info:** NAECON, 140 E. Monument Ave., Dayton, OH 45402

MAY 19, 1977—**Trends and Applications 1977: Computer Security and Integrity** (NBS, IEEE) Natl. Bureau of Standards, Gaithersburg, MD **Prog Info:** Steven Tsakos, Applied Physics Lab., Johns Hopkins Rd., Laurel, MD 20810

MAY 24-27, 1977—**Multiple-Valued Logic (7th)** (IEEE, UNC, et al) Univ. NC, Charlotte, NC **Prog Info:** C. Michael Allen, Univ. of NC, UNC Sta., Charlotte, NC 28223

MAY 24-27, 1977—**PICA-Power Industry Computer Applications** (IEEE) Royal York, Toronto, Ont. **Prog Info:** Lynn M. Gordon, 24 Wimbleton Rd., Islington, Ontario M9A, 3R8

MAY 25-27, 1977—**Electron, Ion and Photon Beam Technology** (ED, AVS) Rickey's Hyatt House, Palo Alto, CA **Prog Info:** T.E. Everhart, U. of Cal., Berkeley, CA 94720

JUN 1-3, 1977—**Conf. on Laser Engineering and Applications** (IEEE/OSA) Wash., DC **Prog Info:** Conf. Mgr.: Anne J. Morandiere, Courtesy Assoc., Suite 700, 1629 K Street, N.W., Wash., DC 20006

JUN 6-8, 1977—**Pattern Recognition & Image Processing** (IEEE et al) Rensselaer Poly. Inst., Troy, NY **Prog Info:** H. Freeman, Elec. & Systems Eng. Dept., Rensselaer Poly. Inst., Troy, NY 12181

JUN 6-10, 1977—**INTERMAG-International Magnetics Conf.** (IEEE) L.A. Hilton, L.A., CA **Prog Info:** Geoffrey Bate, IBM Corp., POB 1900, Boulder, CO 80302

JUN 13-15, 1977—**Intl. Conf. on Communications** (IEEE) O'Hare Inn, Chicago, IL **Prog Info:** Edward J. Glenner, Systems Research, GTE Aut. Elect. Labs., POB 17, Northlake, IL 60164

JUN 13-16, 1977—**Nat. Computer Conf.** (IEEE) Dallas, TX **Prog Info:** Portia Isaacson, U. of Tex., POB 688, Richardson, TX 75080

JUN 20-22, 1977—**Design Automation** (ACM, IEEE) New Orleans, LA **Prog Info:** Harry Hayman, Design Automation Conf., POB 639, Silver Spring, MD 20901

JUN 20-24, 1977—**Intl. IEEE/AP Symp. & USNC/URSI Meeting** (IEEE/USNC/URSI) Palo Alto, CA **Prog Info:** J.B. Damonte, 1716 Hillman Ave., Belmont, CA 94002

JUN 21-23 1977—**Intl. Microwave Symp.** (IEEE) Sheraton Harbor Island Hotel, San Diego, CA **Prog Info:** David Rubin, 3528 Quimby St., San Diego, CA 92106

JUN 21-25, 1977—**World Electrotechnical Congress** (USSR Acad. of Sci., IEC, IEEE, et al) Moscow, USSR **Prog Info:** A.K. Antonov, Organizing Comm., WEC, Kalinin Prospect 19, Moscow G-19, USSR

JUN 22-24, 1977—**Jt. Automatic Control** (IEEE, et al) Hyatt Regency, S.F., CA **Prog Info:** J.S. Meditch, U. of Cal., Irvine, CA 92717

Engineering News and Highlights

David Sarnoff Awards Announced

The 1977 David Sarnoff Awards for Outstanding Technical Achievement, RCA's top technical honors, have been conferred upon 17 members of the scientific and engineering staff. Recipients and their citations are as follows:

James L. Sullivan, Missile and Surface Radar, Moorestown, N.J., received an award *"for conceptual contributions and technical leadership in the development and implementation of advanced signal processing systems."*

Leopold A. Harwood, Consumer Electronics, Somerville, N.J., was honored *"for the invention, development and application of chroma processing integrated circuits used in color television."*

Lucas J. Bazin, Sidney L. Bendell, John J. Clarke, Donald C. Herrmann, Cydney A. Johnson, A.H. Lind, Mark R. Nelson, Dennis M. Schneider, Alexis G. Shukalski, and **Harry G. Wright,** Broadcast Systems, Camden, N.J. were cited *"for team effort leading to the highly successful TK-76 electronic news gathering TV camera."*

Bennie L. Borman, Larry J. Byers, Eduard Luedicke, Larry A. Olson, and **Larry M. Turpin,** were honored *"for team effort in the design, construction, and installation of automated test equipment used in the assembly of color TV chassis."* Dr. Luedicke is with RCA Laboratories, Princeton, N.J. Messrs. Borman, Byers, Olson, and Turpin are with Consumer Electronics, Indianapolis, Ind.

More about the award and winners in the next issue.

Forty-five RCA scientists honored for research achievement in 1976

William M. Webster, Vice President, RCA Laboratories, announced that 45 scientists have been given RCA Laboratories Outstanding Achievement Awards for contributions to electronics research and engineering during 1976. Recipients of the awards are:

David E. Carlson, for development of amorphous silicon material and devices.

Frank J. Marlowe, for designing and developing unique integrated circuits for broadcast equipment.

Richard W. Nosker, for leadership and technical contributions in significant improvements in VideoDisc performance.

Richard E. Novak, for advancement in edge-defined film-fed technology for growth of sapphire ribbons.

W. Ronald Roach, for invention and development of optical instrumentation for the rapid analysis of VideoDisc groove and signal geometries.

Roger C. Alig, David A. de Wolf, and **Angelo Pellos,** for contributions to a team effort in theoretical analysis of the electron optics of television display systems.

Richard A. Auerbach, Henry S. Baird, Allen J. Korenjak, and **Lawrence M. Rosenberg,** for contributions to a team effort in developing advanced techniques for integrated-circuit design verification.

Howard R. Beelitz, Harold Blatter, John G.N. Henderson, Charles M. Wine, Robert M. Rast, and **Juri Tults,** for contributions to a team effort leading to a new digital television tuning system employing frequency synthesis.

Joseph Dresner and **Bernard Goldstein,** for contributions to a team effort in developing and understanding stable, high-gain secondary-emission surfaces.

William L. Harrington, and **Charles W. Magee,** for contributions to a team effort in developing a superior ion probe and applying it, together with ion-scattering spectrometry, to many important thin-film problems.

William R. Lile, John A. Olmstead, Anthony D. Robbi, Joseph O. Sinniger and **James W. Tuska,** for contributions to a team effort in the innovative application of microprocessor technology to the control of automobile engines.

Louis S. Napoli, and **Walter F. Reichert,** for contributions to a team effort in pioneering the development of gallium arsenide microwave field-effect power transistors.

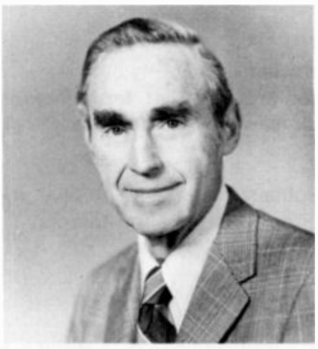
Adolph Presser and **Franco N. Sechi,** for contributions to a team effort in developing high-efficiency linear transistor amplifiers for satellite communications.

James E. Carnes, Robert H. Dawson, Peter A. Levine, and **Donal J. Sauer,** for contributions to a team effort in developing and demonstrating feasibility of charge-coupled delay-line techniques for video-signal processing applications.

Charles B. Carroll, Arthur H. Firester, Macy E. Heller, John P. Russell, and **Wilber C. Stewart,** for contributions to a team effort in the invention and development of optical recording and reading techniques compatible with VideoDisc format.

Richard E. Chamberlain, Wieslaw W. Siekanowicz, and **Frank E. Vaccaro,** for contributions to a team effort in the design and realization of low-loss electron-beam-confinement structures.

Hans W. Lehmann and **Roland Widmer,** for contributions to a team effort in the preparation of very fine patterns of high accuracy and sharpness in various materials by reactive sputter etching.



IEEE honors Avins

Jack Avins received two awards from the Institute of Electrical and Electronics Engineers (IEEE) in recognition of his substantial achievements in television engineering and related fields.

Mr. Avins received the IEEE Consumer Electronics Outstanding Contributions Award as well as a special award from the IEEE Consumer Electronics Group for "his outstanding contribution and loyalty to the Group particularly in the editing and promulgating of standards." An RCA employee since 1946, Mr. Avins retired last year from RCA Laboratories in Princeton, N.J., where he was Staff Advisor to the Director of the Systems Research Laboratory.

Mr. Avins has received two RCA Laboratories Achievement Awards and in 1971 was given RCA's highest technical honor, the David Sarnoff Outstanding Achievement Award, for the development of integrated circuits for TV. He holds more than 50 U.S. patents and has published a number of technical papers.

Zworykin inducted into inventors Hall of Fame

Dr. Vladimir K. Zworykin, Honorary Vice President of RCA who pioneered in the development of television, has been inducted into the National Inventors Hall of Fame along with Edwin H. Land (Polaroid camera), and, posthumously, Lee de Forest (electronic tubes), George Eastman (Kodak camera), and Charles Steinmetz (electric power and transmission).

Dr. Zworykin, who retired from RCA Laboratories in 1954, is the holder of more than 120 U.S. patents on developments ranging from television to gunnery controls to electrically controlled missiles and automobiles. He was involved in many tv developments, including the iconoscope—a tv camera tube that made possible practical picture transmission—and the kinescope or television picture tube.

A successful Vehicle Monitoring System study contract led to a Phase II award for RCA and to a Technical Excellence Award for ten engineers at Automated Systems, Burlington, Mass. Left to right are **Tony Amato**, Manager of Products Engineering; team members **Bill Stewich, Dick Hanson, Tony Muzi, Steve Hadden, Eldon Sutphin,** and **Jim Lynch**; Chief Engineer **Gene Stockton**; and **Harry Woll**, Division Vice President and General Manager. (Missing when the photo was taken were team members **Jim Bardis, Gus Fortin, Ernie Heyl,** and **Dave French**.)

Top TE award for 1976: Don Thomson (center) receives the Missile and Surface Radar, Moorestown, N.J., 1976 Annual Technical Excellence Award. With Don are **Max Lehrer** (right), Div. VP and General Manager, and **Joe Volpe,** Chief Engineer. Don received the award for his long-term performance in systems technology and for his ability to develop solutions to complex problems in the form of practical system definitions. Of particular note were his efforts on the HR-76 radar.

Annual Technical Excellence Awards Dinner for 22 engineers at Missile and Surface Radar in Moorestown. In the photo, seated: **Max Lehrer**, Division Vice President and General Manager (left) and **Joe Volpe**, Chief Engineer (right) flank **Don Thomson**, the 1976 Annual Award Winner. Standing (left to right) are 1976 TE award winners **Mike Stoll, John Douglas, Alex Riell, Marty Herold, Maurice Timken, Bob Socci, Bob Ottinger, Al Eisenberg, Steve Behnen, Ralph Udicious, Dick Smith, Pete Bronecke, Vic Mangulis, Norm Landry, Ron Kolc, Dee Lewis, Irv Kruger, Bob Kooperstein,** and **Ralph Pschunder**. Missing when photo was taken were **Al Schwarzman** and **Jerry Wonderlich**.

The Maxi-Decoy—a small glider containing an ECM payload that is dropped by a high-performance aircraft for ground radar deception—won a Technical Excellence Award for four Automated Systems engineers: **John Furnstahl, Demitrios Lambroupoulos, Frank Tartaro, and Ed Wirtz**. In the photo (left to right) are **Bill Hannan**, Manager of Radiation Systems Engineering; **Tartaro; Lambropoulos; Wirtz; Gene Galton**, Manager of Radiation Programs; **Furnstahl; Gene Stockton,** Chief Engineer; and **Harry Woll,** Division Vice President and General Manager.

Twenty at CCSD get Team of Tigers Award

Twenty employees of Commercial Communication Systems Division, Camden, have received the Team of Tigers award for outstanding achievement in the design, manufacture and marketing of the TK-76, a new portable color tv camera for electronic newsgathering. The award, presented periodically to an individual or group for distinguished performance on the job, was presented by **Irving K. Kessler**, RCA Group Vice President, who initiated the Tiger Award program in 1969.

The award winners are **John C. Adison, Lucas J. Bazin, Sidney L. Bendell, Joseph J. Bulinkis, Louis D. Ciarrocchi, John J. Clarke, Maurice J. Gallagher, Donald C. Herrmann, Robert C. Johnson, Henry H. Klerx, George H. Laning, Anthony H. Lind, James, J. McCormac, Miles G. Moon, Mark Nelson, Vincent R. Renna, Dennis M. Schneider, Aleyis G. Shukalski, Dale, C. Smith,** and **Harry G. Wright.**

Cited by NASA for a new technology disclosure: **Ed Nossen,** and **Eugene Starner**, right, both of Government Communications Systems, Camden, N.J. **J.B. Howe, Sr.**, Chief Engineer, presents the award. The certificate and cash award were given for "Digital Voltage Controlled Oscillator Doppler Extraction Technique." This technique permits the measurement of an unknown frequency with precision limited only by the reference oscillator quality and the signal-to-noise ratio of the unknown signal. The method can readily measure the doppler component of a received signal to within a millihertz.

Authors and inventors honored at Moorestown

Sixty-one individuals were honored on February 24 at Missile and Surface Radar's Authors' Reception held in Moorestown, N.J. This reception, hosted by Joseph Volpe, Chief Engineer, was the tenth in a series to honor people who have presented or published papers or received patents.

In congratulating the MSR honorees, Mr. Volpe called attention to the efforts involved in professional achievements:

"I'm aware of the amount of time and effort involved in generating and reworking a technical paper draft or patent disclosure. It's a real sacrifice, especially when it's added onto an already imposing workload. My congratulations to the individuals being honored for their contributions, and also my sincere thanks for a special effort."

Eta Kappa Nu honorable mention for John Henderson

John G.N. Henderson, a Member of the Technical Staff of RCA Laboratories, has received an Honorable Mention Award as part of the **Eta** Kappa Nu Outstanding Young Electrical Engineer Awards.

An honorary national engineering society, Eta Kappa Nu each year cites outstanding electrical engineers who are less than 36 years old.

Mr. Henderson, who is 31, received a Bachelor's degree, cum laude, in Electrical Engineering from the University of Pennsylvania in 1967 and a Master's degree from Princeton University in 1969. He has worked at RCA Laboratories since graduating from Penn in 1967 and has twice won RCA Laboratories Outstanding Achievement Awards for his research on television receivers.

Anderson and Dingwall named Fellows

Bill Webster, Vice President, RCA Laboratories, Princeton, N.J., appointed **Charles H. Anderson** and **Andrew Dingwall** Fellows of the Technical Staff, in recognition of their outstanding contributions. The designation of Fellow is comparable to the same title used by universities and technical societies. It is given in recognition of a record of sustained technical contribution in the past and of anticipated continued technical contribution in the future.

Upcoming issues

Our next issue (Apr/May) covers **electro-optics systems—image** tracking, the hand-held laser rangefinder, CCTV surveillance, and laser inspection systems, among others.

Our anniversary issue usually highlights **the most important technological advances** that have taken place throughout RCA during the last year. Our Jun/Jul issue this year is no exception; it highlights the PRICE system, the TK-76 camera, the microprocessor-controlled safety automobile, the ANIK B satellite, the Xtended-life color tv chassis, and more.

Further ahead, look for issues on **advanced communications, hybrids, radar,** and **SelectaVision.**

Editorial Representatives

Contact your Editorial Representative to schedule
technical papers and announce your professional activities.

Commercial Communications Systems Division		
	Broadcast Systems	**BILL SEPICH*** Camden, N.J. **KRISHNA PRABA** Gibbsboro, N.J. **ANDREW BILLIE** Meadow Lands, Pa.
	Mobile Communications Systems	**FRED BARTON*** Meadow Lands, Pa.
	Avionics Systems	**STEWART METCHETTE*** Van Nuys, Cal. **JOHN McDONOUGH** Van Nuys, Cal.
Government Systems Division		
	Astro-Electronics	**ED GOLDBERG*** Hightstown, N.J.
	Automated Systems	**KEN PALM*** Burlington, Mass. **AL SKAVICUS** Burlington, Mass. **LARRY SMITH** Burlington, Mass.
	Government Communications Systems	**DAN TANNENBAUM*** Camden, N.J. **HARRY KETCHAM** Camden, N.J.
	Government Engineering	**MERLE PIETZ*** Camden, N.J.
	Missile and Surface Radar	**DON HIGGS*** Moorestown, N.J. **JACK FRIEDMAN** Moorestown, N.J.
Solid State Division		**JOHN SCHOEN*** Somerville, N.J.
	Power Devices	**HAROLD RONAN** Mountaintop, Pa. **SY SILVERSTEIN** Somerville, N.J.
	Integrated Circuits	**FRED FOERSTER** Somerville, N.J. **JOHN YOUNG** Findlay, Ohio
	Electro-Optics and Devices	**RALPH ENGSTROM** Lancaster, Pa.
Consumer Electronics		**CLYDE HOYT*** Indianapolis, Ind. **RON BUTH** Indianapolis, Ind. **PAUL CROOKSHANKS** Indianapolis, Ind. **CORT HILL** Indianapolis, Ind.
SelectaVision Project		**FRANCIS HOLT** Indianapolis, Ind.
RCA Service Company		**JOE STECGER*** Cherry Hill, N.J. **RAY MacWILLIAMS** Cherry Hill, N.J. **DICK DOMBROSKY** Cherry Hill, N.J.
Distributor and Special Products Division		**CHARLES REARICK*** Deptford, N.J.
Picture Tube Division		**ED MADENFORD*** Lancaster, Pa. **NICK MEENA** Circleville, Ohio **JACK NUBANI** Scranton, Pa. **CLINT BELL** Marion, Ind.
RCA Communications	**Alascom**	**PETE WEST*** Anchorage, Alaska
	Americom	**DON LUNDGREN*** Kingsbridge Campus, N.J. **MAUCIE MILLER** Kingsbridge Campus, N.J.
	Globcom	**WALT LEIS*** New York, N.Y.
	RCA Records	**JOSEPH WELLS*** Indianapolis, Ind.
NBC		**BILL HOWARD*** New York, N.Y.
Patent Operations		**JOSEPH TRIPOLI** Princeton, N.J.
Electronic Industrial Engineering		**JOHN OVNICK*** N. Hollywood, Cal.
Research and Engineering		
	Corporate Engineering	**HANS JENNY*** Cherry Hill, N.J.
	Laboratories	**CHET SALL*** Princeton, N.J. **LESLIE SCHMIDT** Somerville, N.J.

*Technical Publications Administrators are responsible for review and approval of papers and presentations.

RCA Engineer

A TECHNICAL JOURNAL PUBLISHED BY CORPORATE TECHNICAL COMMUNICATIONS
"BY AND FOR THE RCA ENGINEER"

Printed in U.S.A.