CHAPTER 1

Basic Decisions In System Design

INTRODUCTION

Texas Instruments has developed and is manufacturing a family of microprocessor products and systems based on the architecture of its 990 minicomputer. The purpose of this book is to present enough factual information about the 9900 and the family of devices and systems surrounding it to serve not only as a guide for deciding to use the 9900 in an application, but also as the primary reference for design and programming activities. The book is much more than a data book or a collection of application notes. It contains basic concepts, presents methods and techniques, and most important of all, shows how the architecture of the 9900, substantially superior to other microprocessor architectures, can be translated into cost effective applications.

The time investment you make in learning how to use the 9900 will inevitably produce substantial benefits because your designs will be advanced well beyond other microprocessor systems; they will be expandable, flexible, easily upgraded and will not be obsolescent. The capital investment in programming systems will bring powerful computing equipment and software tools to your design team that will have them out-distancing the competition in a very short time.

In reading this book, you will see the 9900 product as more than a single microprocessor. You will find a family of processors, peripherals, boards, minicomputers and systems all based on a single architectural concept called *memory-to-memory architecture.* It is this basic principle which, when fully understood at the fundamental level, will help you understand why and how the 9900 can be used to implement outstanding products. In addition, you will see why Texas Instruments has made the commitment to the continued support of the 9900 family in both hardware and software. New microprocessors and peripheral devices will retain and complement the basic architectural features—the 16-bit word length, the instruction set, the I/O techniques. etc. Texas Instruments software support goes beyond the standard assembler, editor, linker and PROM programmer software. New design tools such as POWER BASIC and PASCAL are now available. These powerful software products bring structured programming disciplines into focus and help you to attain an advanced programming capability.

All in all, the book is a collection of useful factual material which should be of substantial benefit to anyone considering designing with microprocessors. For those who have very limited exposure to designing with semiconductor products, the next few sections will be helpful because the theme of "more functions at lower cost" is demonstrated. These ideas lead to the basic philosophy that designing with *standard hardware* — semiconductor LSI products which are *programmable* — is the most economical procedure, and should be carefully considered for *every* new electronic product.

THE IMPACT OF SEMICONDUCTORS

In the short thirty years since the invention of the transistor (the first semiconductor device to exhibit amplification), there have been more inventions and more scientific and engineering accomplishments than in all time previous. The field of digital electronics (especially computers) has been the greatest contributor of new products for these accomplishments and, therefore, has become one of the most rapidly growing industries. Manufacturers of semiconductor components (transistors, integrated circuits, microprocessors and memories) have been providing the building blocks, and the equipment manufacturers have been taking advantage of the opportunity by developing the most sophisticated systems that are economically feasible.

In his keynote address to the 1977 National Computer Conference, Mark Shepherd, Chairman of the Board of Texas Instruments, made the following points:

> "Until 1971, the semiconductor industry was in the circuits business. Semiconductor circuits, complex though they were, constituted only a fraction of an entire system. The one-chip calculator developed in 1971 was the first significant complete system. Since then many calculators and watches have been developed where the entire system function is accomplished by one or a few semiconductor chips. These were custom chips because the technology did now allow any reserve computing power for other applications.
>
> "The semiconductor industry has now entered an era where the entire system function of an end product can be accomplished by a few semiconductor chips, or a single chip, with enough versatility to permit adaptation to many different applications through the programming.
>
> "This change carries enormous implications for the system designer. 1) The lead time for system implementation is shortened because no special chip development is required. 2) The development cost will be low because it will be limited to software (which may be executed in hardware). 3) The required degree of electronic sophistication on the part of the user is much less. To achieve these advantages the system designer must be prepared to use standard products produced in large volume rather than custom devices.
>
> "The functional equivalent of a medium-scale computer *(Figure 1-1)* cost $30,000 in the early 1960s. Its cost equivalent has now dropped to $4,000 and is projected to be at less than $100 by 1985, penetrating the personal cost threshold. As this is accomplished, greater challenges will be encountered in the cost of sales, service, and maintenance, requiring that we learn to incorporate self-diagnostic and self-repair functions into our systems."

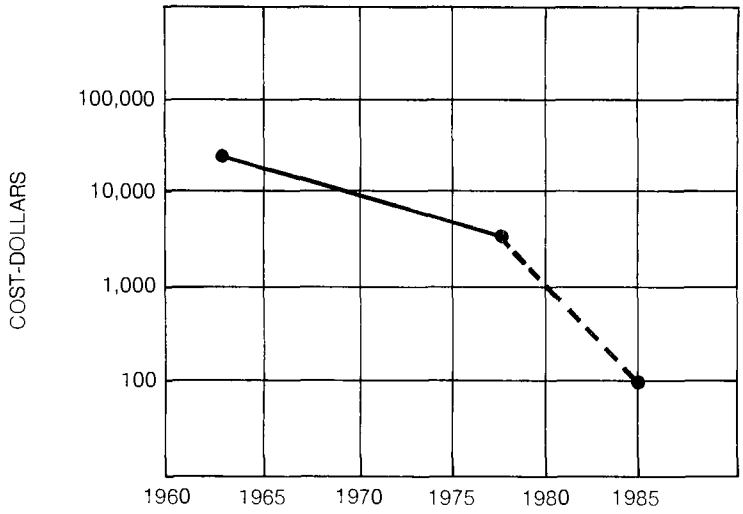


Figure 1-1. *Cost of Medium Scale Computer*
(M. Shepherd, 1977 NCC)

The cost of the hardware components for a typical digital system has been decreasing with time because new and more powerful semiconductor devices have been developed. Equally important is the fact that the development cost for the typical digital system hardware has also been decreasing. *Figure 1-2* illustrates how impressive this cost reduction has been. Contrast the figures of 7-8 *million* dollars in the early fifties with 8-9 *thousand* dollars in the late seventies; digital system development cost has been reduced by a factor of *one thousand* in a period of 25 years! An extension of this trend indicates that typical system hardware development cost will be approximately $1,000 by 1985.

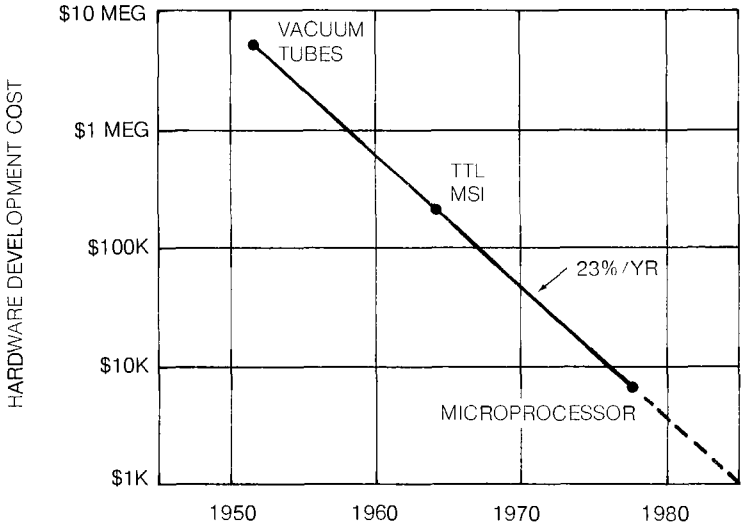


Figure 1-2. *Typical Digital System Hardware Development Cost at OEM Manufacturer*

How has this been accomplished? *Figure 1-3* shows what has been happening. As the number of components per chip of silicon increases, the development cost for each chip also increases. For a semiconductor manufacturer, volume production is required to offset the development cost. Semiconductor devices are therefore being *batched fabricated* — a few hundred, a few thousand per chip — and this means lower cost per *active element group* or AEG. (An AEG is defined as a logic gate, flip-flop, or a memory cell.)

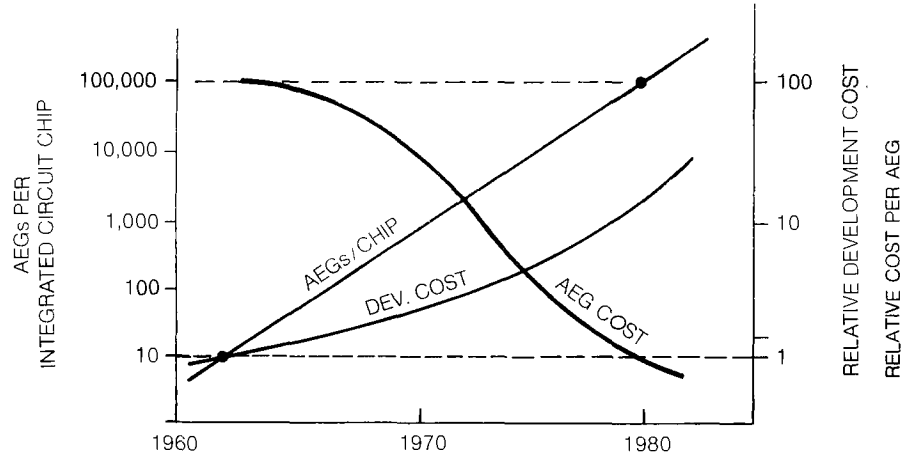


Figure 1-3. *Evolution of Semiconductor Technology*

Figure 1-4 shows the chronology of semiconductor device development. An AEG in the early 1950's consisted of one or two transistors, several resistors, a capacitor or two, and some area of a printed circuit board to hold the parts together as an assembly. Early integrated circuits contained about 10 AEG's. Then medium scale integration achieved up to 100 AEG's per chip and large scale integration reached 1,000 AEG's per chip.

At this point (the late 1960's), the semiconductor technologists had apparently reached an impasse. If they continued to increase the number of AEG's per chip the high degree of specialization would preclude volume production, and the benefits of LSI would be lost. In fact, the only area in which LSI appeared to be feasible was in memories — primarily read/write memories now called RAM's. Read only memories (ROM's) and programmable read only memories (PROM's) were not required until later (as you will see). But the semiconductor technologists continued their thrust toward greater numbers of AEG's per chip, focusing primarily on memory products.

There was one other product which appeared to be feasible (in 1970) — a single-chip calculator. Here was an opportunity to stretch the imagination to greater degrees of achievement. At the producibility level of about 1,000 AEG's per chip, all of the functions of a microcomputer could be built on one chip — and the application certainly had the required volume potential. So *custom* LSI found a niche in the form of the hand-held calculator.

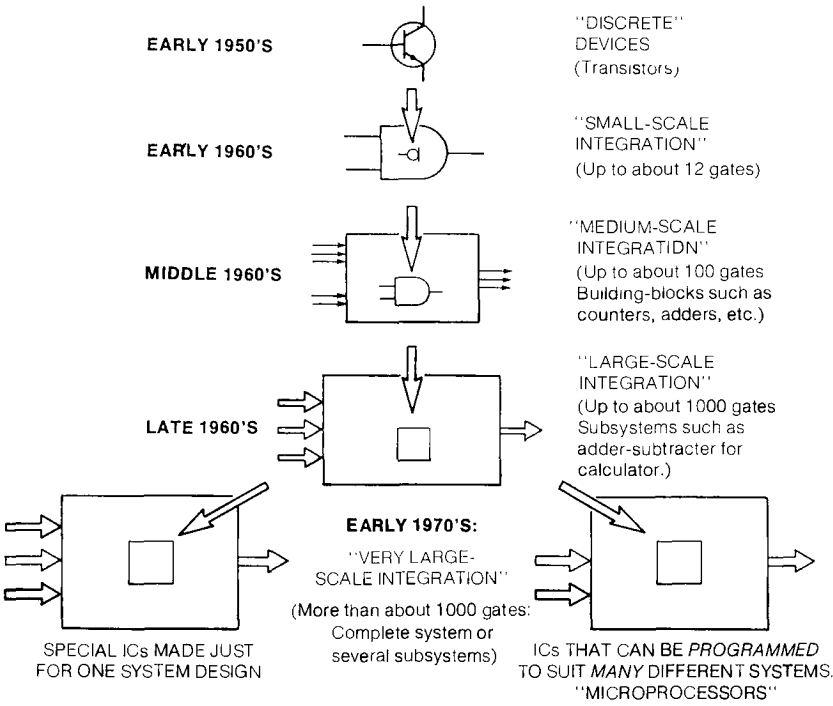


Figure 1-4. *Stages in Evolution of Digital Semiconductor Circuitry.*
(G. McWhorter, Understanding Digital Electronics, Texas Instruments Inc., Dallas, Texas, 1978)

From the very beginning the designers of the single chip microcomputer envisioned new and varied applications of this device, so it was made with a ROM for instructions and RAM for data. It was *programmable,* at least it was "mask programmable." And as we witness the growth of this segment of the semiconductor market, we see a host of dedicated applications for single chip microcomputers such as controllers for microwave ovens, sewing machines, and other appliances.

By designing a "standard" chip that could be *programmed* to do a variety of jobs, semiconductor technologists repeated the step taken by the inventors of the first programmable machine — the first computer — in the late 1940's. The first digital computer was a *stored program* digital calculating machine. Programming provided versatility and variety of applications. Similarly, programmable single chip, LSI semiconductor devices — microcomputers — gave LSI variety of applicability.

The next logical step in the evolution of LSI was the design of the general purpose microprocessor, a computer CPU on a chip. By interfacing the microprocessor to a memory — a set of chips arranged to provide as much storage as needed — one can build larger, more powerful microcomputers which can replace special purpose hardwired logic. In fact, general purpose hardware that is *programmable* provides *multichip* applicability of LSI technology.

With this breakthrough in the concept of LSI application, the semiconductor technologists have been motivated to continue to increase the number of AEG's per device. *Figure 1-5* projects the growth of AEG's per chip to over 10^6 by 1985 — a level sufficient for a single chip 32-bit microcomputer. The 16-bit microprocessor and 4K RAM require about 50,000 AEG's. RAM's of 16K and 64K bits requiring up to 100,000 AEG's are not unrealistic extensions of the trends; they are real products rapidly moving into the marketplace. New advances are being made in semiconductor process technology to achieve the packing densities needed for the future. As *Figure 1-5* indicates, optical techniques for defining regions and interconnections reach a resolution limit at about 10^5 AEG's. E-beam and X-ray technology will be required to further increase component density.

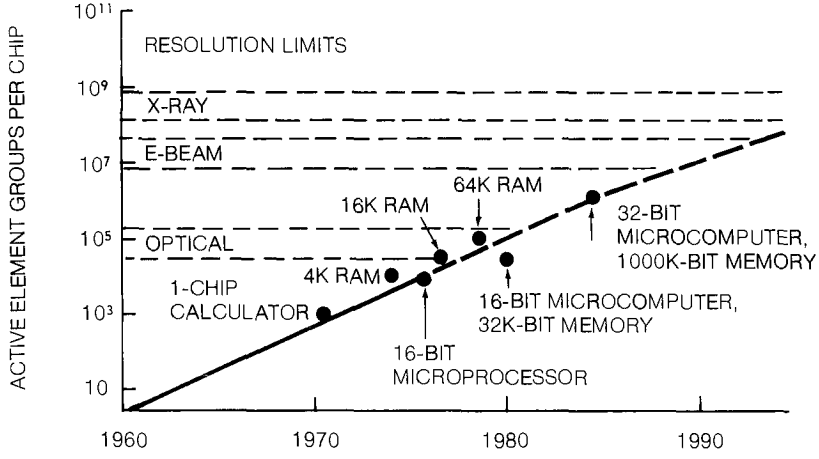


Figure 1-5. *Semiconductor Chip Complexity*
(M. Shepherd, 1977 NCC)

The impact of programmable semiconductor devices is shown in *Figure 1-6.* Prior to 1972, semiconductor devices were designed as *circuits.* Now they are being designed as *systems* or at least subsystems. As the number of AEG's/chip continues its rise, driving down the cost of CPU and memory devices, unlimited opportunity is being created for an unbelievable variety of new products.

Figure 1-7 shows that a dramatic change is anticipated in the rate of AEG cost reduction with time due to the impact of microprocessors. Functions (AEG's) costing $1.00 in 1966 were obtained for around *5 cents* in 1976. In fact, the cost per AEG is projected to be less than *a tenth of a cent* by 1985.

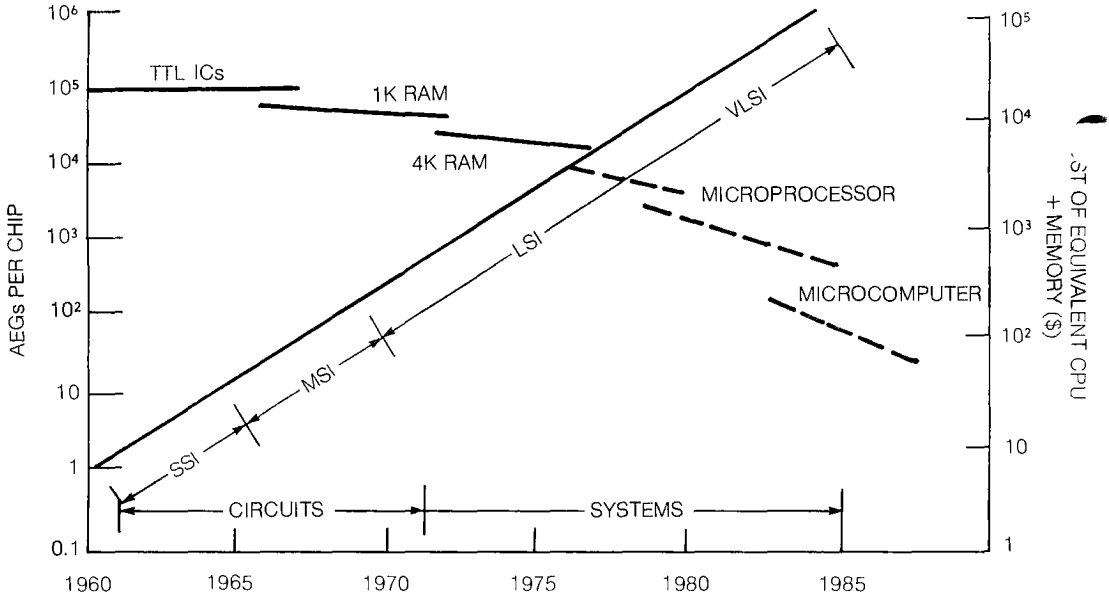


Figure 1-6. Distributed Semiconductor Power
(M. Shepherd, 1977 NCC)

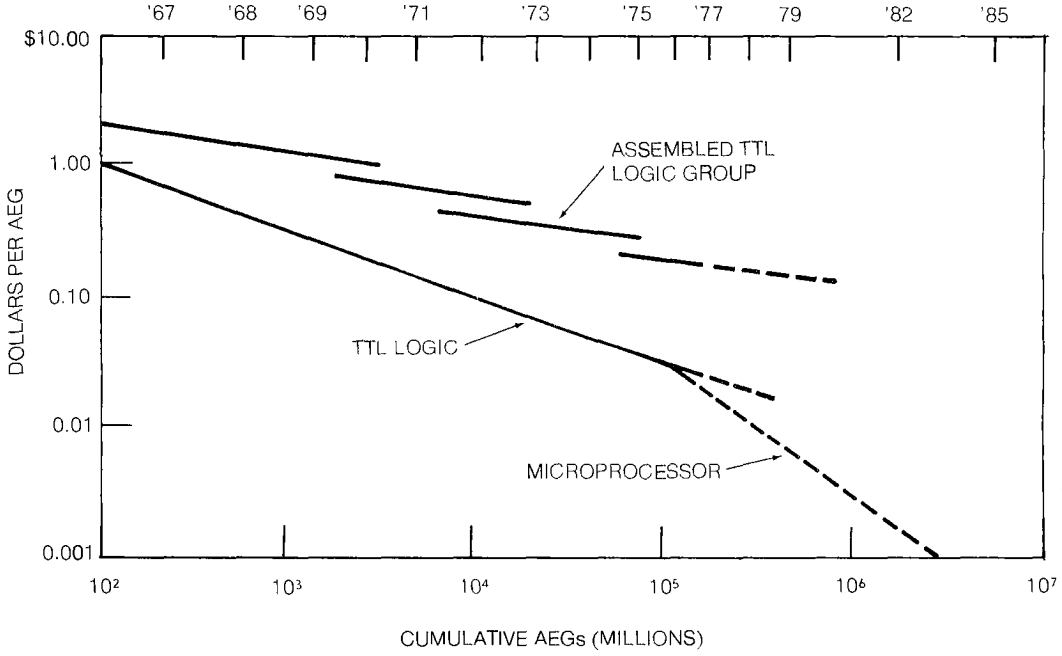


Figure 1-7. Cost Per AEG for TTL and Microprocessor
(M. Shepherd, 1977 NCC)

Memory costs (on a per bit basis) are diminishing, too. Following the projected trends for the cost of AEG's, RAM cost is forecast to be less than .05 cents per bit by 1982 (*Figure 1-8*). The need for various memory types has now been established. Programs for microcomputers are stored in non-volatile memories such as ROM's, PROM's and EPROM's. ROM's are mask programmable by the manufacturer and are best suited for high volume applications. PROM's are programmable after the devices are completely packaged. Either the manufacturer, the distributor or the user may store the desired program (or data) in a PROM. PROM's are suited for medium volume to low volume applications. EPROM's are erasable and so find use during prototyping and development cycles. They are also used in applications where software must be periodically changed, upgraded, or modified in any way. Other memory technologies such as CCD's (charge coupled device) and bubbles will be used for mass storage requirements where speed is not critical.

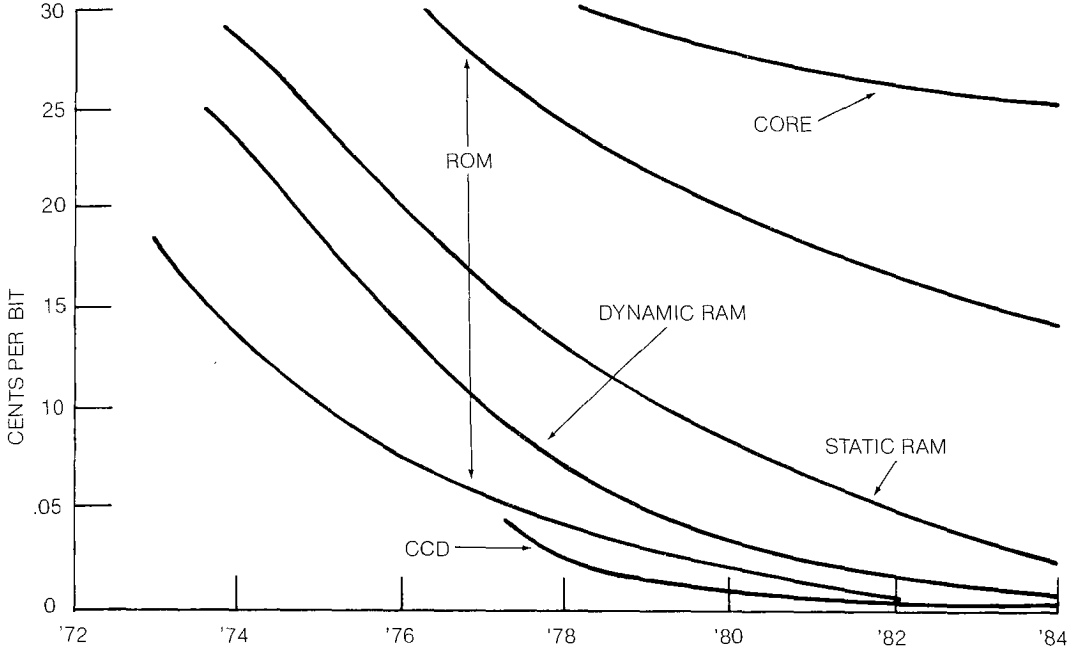


Figure 1-8. *Memory Cost Comparison*

(M. Shepherd, 1977 NCC)

The effect of modern semiconductor technology has been to alter the roles of the component manufacturer and the OEM (original equipment manufacturer). Component manufacturers are continuing to produce batch-fabricated semiconductor products. But the economic benefit — the low cost per AEG — of batch fabricated semiconductor devices with high functional density cannot be realized except through applications which are program controlled. The component manufacturer must therefore provide programming support via PDS's (program development systems) and software products to enable the OEM to develop applications programs. Thus increased development cost of high functional density devices is found not only in improved process technology and in the design of LSI masks, but also in the attendant software support products. And volume production is required to offset these costs.

The role of the OEM is undergoing a corresponding shift. Component costs and the assembly cost of hardware have been sharply reduced. *Table 1-1* demonstrates the evolutionary steps in hardware costs. The cost improvement ratio of each step as compared with the previous step is dramatic: overall, it is 600:1.

Table 1-1. *System Cost Reduction*

EVOLUTIONARY STEP	COMPONENTS TO ASSEMBLE	TOTAL COST	
		COMPONENTS + ASSEMBLY COST	COST IMPROVEMENT RATIO
DISCRETES	20000-30000	6000-9000	—
IC'S (GATES & FLIP FLOPS)	350-500	600-900	10:1
IC'S + MSI	125-150	250-450	2,5:1
MICROPROCESSORS	7-10	120-190	2:1
MICROCOMPUTERS	1	6-12	12:1

While hardware costs are decreasing, the software costs, as a percentage of the overall design effort, are increasing. *Figure 1-9* illustrates the relationship of hardware to software costs in product development and the change in emphasis. In the 1950's computers were only used in large-scale business and scientific applications. OEM's had no opportunity to use computing power in their systems. When minicomputers were introduced in the 1960's, OEM's found applications in process control and small business EDP functions, and therefore had to provide some special programs for their use. With the advent of microprocessors in the 1970's, the software component of the development cost increased further, and this trend can easily be forecast into the 1980's — less than 25% of the development cost of most products will be for hardware.

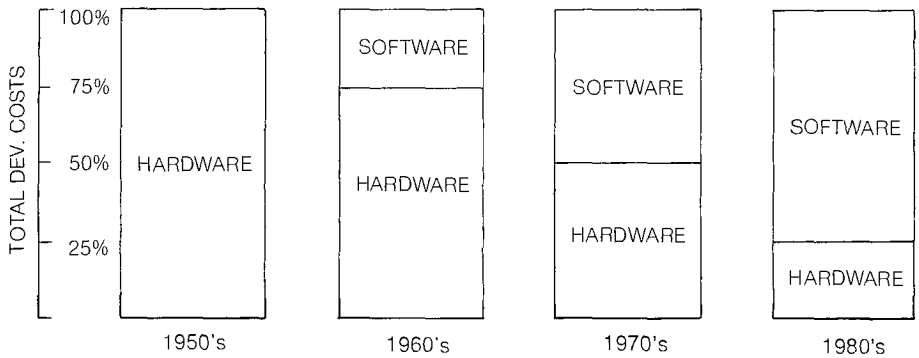


Figure 1-9. *Increasing % of Software Development Cost*

Development costs are changing — becoming more software oriented — and this has a strong impact on overall product cost. In any product design, the development cost is amortized over some production quantity, and this affects the price of the product. But developing *software* to achieve any design goal is less expensive than developing *hardware* to do the same thing. Therefore, the *total* development cost for "equivalent systems" is decreasing (perhaps by as much as 15-20% per year).

The development of *programmable semiconductors* has been a major accomplishment equivalent in importance to the inventions of the transistor, the integrated circuit, and the stored program computer.

The trends appear to be well established. The number of AEG's per chip will be increasing by at least *75% per year* for at least another two decades. As a result, AEG cost will decline by about *50% per year* and RAM cost per bit will decline by about *20% per year.* The computing power of LSI devices will increase while the price will continue to decrease. The impact will be felt in all walks of life.

APPLICATIONS OF PROGRAMMABLE SEMICONDUCTORS

The application of programmable semiconductors can be considered as an extension of the application of computers. All applications of LSI semiconductor devices are as computers because microprocessors, microcomputers and programmable LSI peripheral chips are *programmed* to perform the special functions required for each application. All the elements of a computer — ALU, control, memory and I/O — are present.

As the price of computing power decreases, the number of applications increases. The number of computers of any given type being applied is inversely proportional to the cost (*Figure 1-10*). As of 1976 there were relatively few systems in the $100-$10000 range. But microcomputers are changing this. Applications are being found in new designs of digital electronic systems, in products previously using electro-mechanical devices, and in new products which previously were not economically feasible.

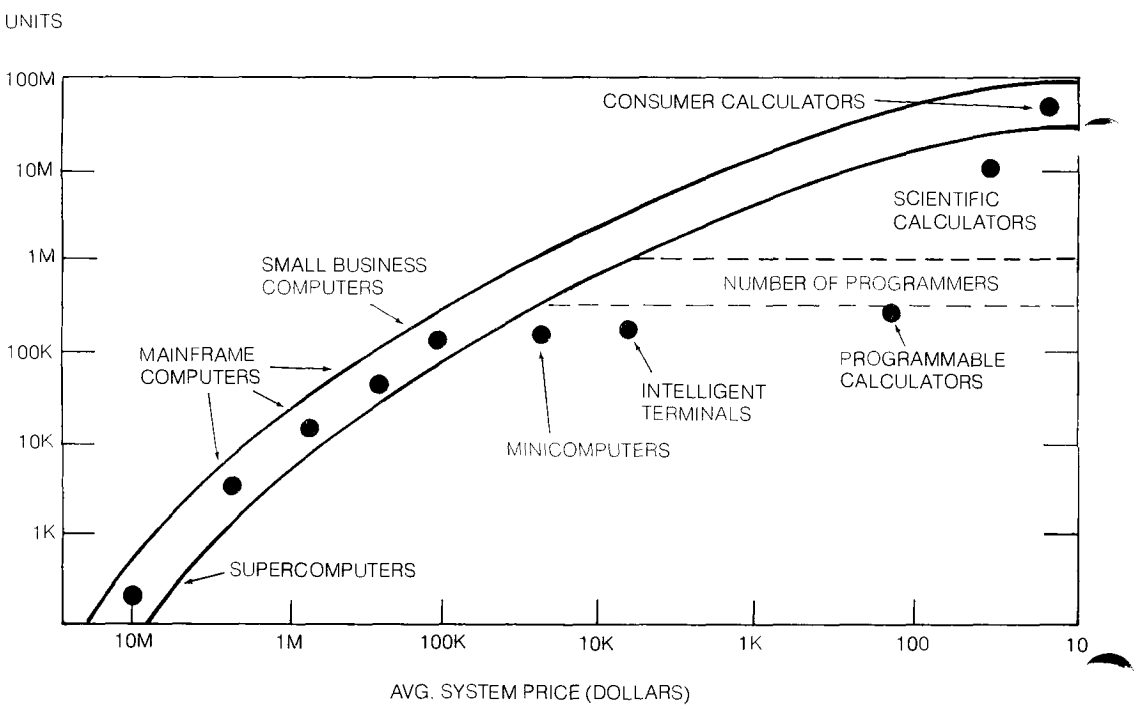


Figure 1-10. U.S. Installed Computer Base—1976
(M. Shepherd, 1977 NCC)

While some people may feel that the number of computers cannot exceed the number of "programmers" (approximately one million according to *Figure 1-10*), it is evident that all designers of products which use microcomputers will acquire the necessary programming skills to achieve the desired end product results.

Single Chip Microcomputer Applications

Single chip microcomputers are being used in the small, dedicated, high volume applications such as calculators, microwave ovens, and general appliance controllers. As the computing power of single chip devices increases, the range of applications will obviously expand. Early devices contained about 1K bytes of memory. New devices with 2K bytes of ROM for instructions and small amounts (256 bytes) of RAM for data have been built and designed into more complex applications. One example is a terminal controller using the TMS 9940 microcomputer with one support chip; this is described in Chapter 9.

Multi-Chip Microcomputer Applications

The application areas which involve the greatest number of designers and programmers by far are those using a multi-chip approach — a microprocessor, memory sized to the application, and peripheral interface devices. Limitations are much less in multi-chip systems than for single chip microcomputers. Designs can be accomplished using the general purpose microcomputer boards which have been designed to be applied to a variety of end products. Or the designer can start with individual LSI devices and build a special microcomputer for each application.

The list of applications for microprocessors is long and continues to grow. But a few of the representative areas are these:

Instrumentation
Test Equipment
Industrial Process Control
Point-of-sale Terminals
Cash Registers
Typewriter/Word Processing Equipment
CRT Terminals
Vending Machines
TV Games
Automobile Engine Ignition Controllers
General Automotive Products
CB Equipment
Communications Controllers
Educational Toys
Personal Computers
Special Dedicated EDP Functions

In each application standard programmable semiconductor LSI devices are used to sense input information, process the information according to special procedures (algorithms), and send information to external devices for display, printing, physical control devices, etc. Obviously the need for interface circuits is great. They cover specific functions such as A/D converters, D/A converters, transducers, and special display drivers, etc., as well as standard digital circuits for buffering, multiplexing, latching, etc. *Figure 1-11* shows conceptually how the elements of the microcomputer are arranged for any application.

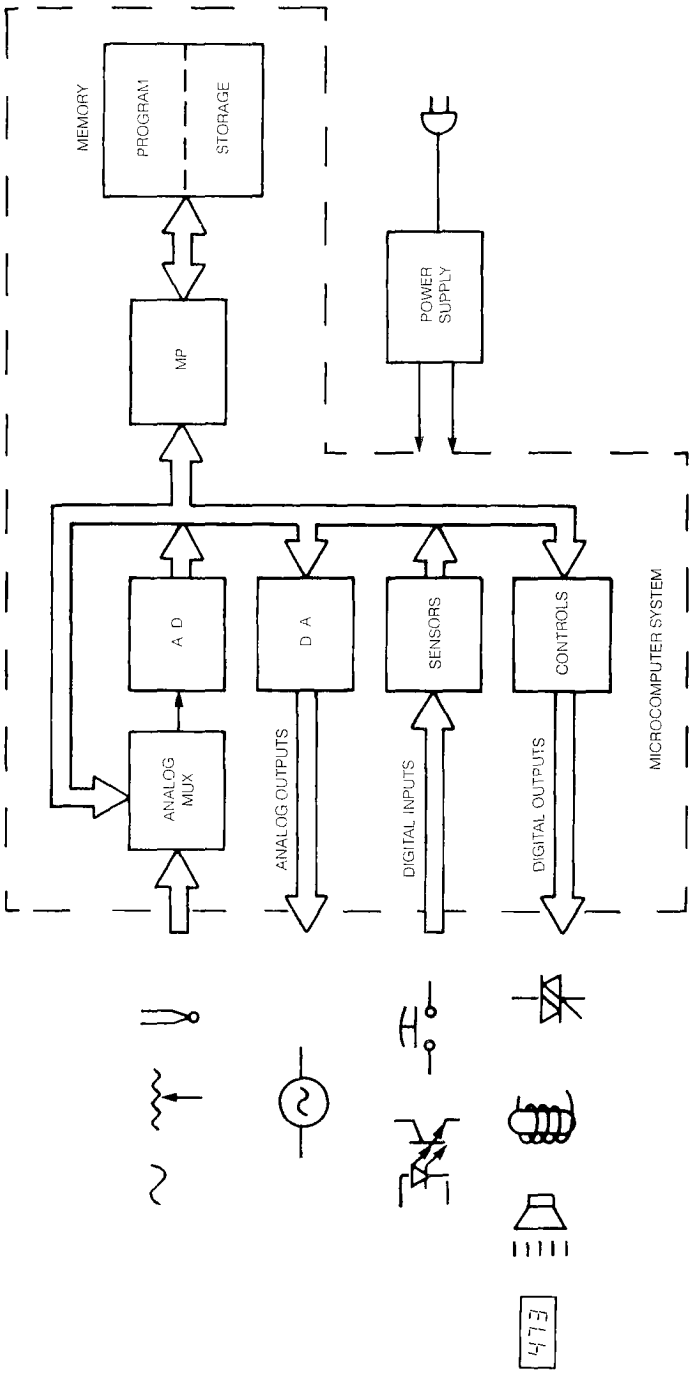


Figure 1-11. Microprocessor Applications in Process Control Systems

BUILDING A MICROPROCESSOR BASED SYSTEM

Given an application idea, how does one proceed toward designing a product in which a microprocessor is the central control device? The design steps may be diagrammed in great detail, but the most important steps are these.

1. System Specifications — The system requirements include electrical specifications for each input and output, timing details, and overall performance logic.

2. Division into small subsystems — By defining small, easily managed tasks, the hardware and software requirements can be measured, and design can be scheduled.

3. Decisions for hardware and software — This is the appropriate design point at which the tradeoff between hardware and software solutions for each task is evaluated. For economy, the best solution may appear to be software, but there may be a penalty in performance.

4. Hardware and software design — Here the two design activities may be carried out in parallel. The microcomputer parts are assembled on one or more breadboards and tested for signal flow. Software is developed using a software development system (a computer with appropriate peripherals and programs). Software testing may be done to provide algorithm functionality.

5. System integration — Ultimately, the hardware must be tested under program control. At this point the programs must be loaded into the system memory (usually PROM or EPROM) for testing. Often special logic analyzers and other computer based diagnostic tools are needed to debug the complete system (see the description of the AMPL system in Chapters 2 and 7).

It is clear from the foregoing list of steps that a thorough understanding of the hardware components and a knowledge of programming is required to design with microprocessors and microcomputers. But this is not difficult to acquire. By learning the names of standard building blocks and software packages, you will have taken a major step toward understanding what you read about them.

Basic Hardware Components

Since the hardware for digital systems is being standardized, the basic elements and their functions can easily be studied. Comparisons of similar devices from various manufacturers must be made and design tradeoffs evaluated. Here are the basic building blocks, what they do, and how they are used.

Microprocessor or CPU

This fundamental chip contains the Arithmetic and Logic Unit (ALU) which basically performs addition and comparisons between two numbers. Temporary storage registers are available to hold numbers (called operands) and addresses (memory location numbers) which identify or point to instructions and data. Sometimes the ALU is used to calculate addresses by arithmetic operations on certain register contents. The microprocessor must also contain timing and control circuitry to direct all activities in an orderly step-by-step procedure. The actual control functions are determined by decoding and executing instructions. Instruction execution is a special type of operation on information which comes from memory. The memory stores numerical values which may be interpreted by the processor in one of two ways. Either the number is an instruction, which will direct the sequence of operations over the next few clock cycles, or it is data to be operated upon either arithmetically or logically.

Memory

The main memory of a microcomputer holds the program and data for the system. Because semiconductor RAM devices are volatile (that is, all data is lost when power is removed), it is desirable to use ROM devices (Real Only Memory or non-volatile memory) for the program and RAM for data. ROM devices are programmed (information stored in the cells) by means of a metalization pattern or mask at the time of chip fabrication. Programmable read only memories (PROM's) may be programmed by the manufacturer or the user because information is stored by burning small metalic fuse links via the application of electric current. Programming is performed on the device after it has been packaged. EPROM's are non-volatile read-only memories which may be *erased,* usually via the application of ultraviolet light. These devices are especially useful in prototyping and system development during which program changes are frequent.

Memory devices are designed for cascading so that any size memory may be obtained by adding more devices. Capacities of 4K bits per chip are common; devices with 64K bits per chip are not far away.

Input/Output

For the input and output function — interfacing the microprocessor-memory combination to the "outside world" — usually consists of a variety of devices including programmable LSI devices. Examples of interface requirements are as follows:

1. For communication of digital information over a pair of wires, conversion from 8-bit bytes (parallel) to single bits sent in sequence (serial) is required. The I/O device must receive a number of bits, hold them in a register and then shift them serially to a transmission line. The reverse procedure, serial to parallel conversion, must be performed for receiving information from the transmission line. Since the clock rates,

start and stop characters, and "handshaking" requirements can be complex in communications networks, the protocol is designed into the TMS 9902 and TMS 9903 programmable communications controllers (see Chapter 8 for details).

2. Man-machine interfacing may consist of arrays of switches and indicators or may be performed via a terminal such as a teletype (TTY) or a video display terminal (VDT). Arrays of switches are connected to microcomputers via multiplexers. The address bus may be used to select one of the switches for sampling at any given moment. Addressable latches are useful in supplying on-off data to arrays of indicators. The address bus is again used to select one specific display device (a single lamp) to be turned on (or off) in a given computer cycle. Terminal interfacing can be accomplished via a serial data interface such as the TMS 9902 (see Chapter 9 — example using the TM 990/100M board).

3. The broad category of analog (continuously variable) inputs and outputs requires converters (A/D and D/A) to obtain digital information on the computer side of the interface. Input signals from transducers or output signals to actuators (positioners) require this type of conversion.

Connecting the I/O devices to the CPU and addressing them may present problems in some microcomputer systems. The 9900 solves the problem by providing two types of general purpose I/O. Memory mapped I/O allows a set of memory addresses to identify the I/O devices (as though they were words of memory), while the communications register unit (CRU) provides a separate I/O port specially designed to interface single bit devices, communications devices, standard computer peripherals, etc. Unique to the 9900 architecture, the CRU interface is a powerful and versatile I/O technique; it is easily utilized via the special LSI peripheral supports circuits (such as the TMS 9901, 2, and 3, and the TIM 9905 and 6).

The rules for interconnecting the various elements of the microcomputer include loading specifications and signal level limitations. In observing these rules the designer will occasionally use a few standard devices to reduce loading or perform level shifting. Generally, the addition of such devices is an insignificant part of the overall design. (Details for hardware interfacing are given in Chapter 4.)

Programming for Microcomputers

The writing of programs — often called software development — is the companion activity to hardware breadboarding and testing in computer systems. But software is substantially more flexible than hardware because it consists primarily of *ideas,* documented in strings of characters on a page, or in 1's and 0's in a memory. In fact, until a program is actually loaded into a memory, it is truly a set of ideas on paper, hence the contrasting name, *software.*

In developing the individual hardware components of a microcomputer, designers usually subdivide the activities into small, easily managed tasks. These tasks are performed sequentially by one designer or simultaneously by several members of a design team. The same is true of software design. Small, easily defined and understood sub-programs are given as individual assignments to the programmers on the design team.

The disciplines for programming are set up so that each sub-program stands alone, yet couples to the other sub-programs in a harmonious manner. But the overall plan begins at the top (a program to handle all sub-programs) and expands to several lower levels (a "Christmas tree" of programs). This is known as "top-down programming", and it is a form of structured programming.

The term *structured programming* means that discipline in programming in which each program module implements an algorithm with a single entry point, a single exit point and a definitive result for each possible input. Each module must contain its own buffer area so that it cannot alter procedures or data of other modules. (In some cases common buffers are allowed, but complex rules for their use are needed.)

How is programming done? What equipment is needed? And what support can you get from a microcomputer manufacturer? First, there is a preparation phase in which the designer and/or programmer must become familiar with the instruction set and the architectural elements of the microcomputer selected for the design. The second phase involves writing selected short program segments to gain insight into the memory requirements and the execution speed of various sub-programs. Then the main body of the program may be developed.

Writing programs means writing code; writing program steps which must be executed in sequence. Usually these steps are written in a mnemonic language which uses one to four letters as operation codes, and strings of other characters to designate the operand (the number to be operated upon). These program steps must be translated and "assembled" into a set of 1's and 0's — the machine language executable by the microcomputer — by a special program development computer.

The programmer writes the program on paper. Then he enters the program steps via a keyboard into the program development system (PDS), and directs the PDS to "assemble" the instruction into machine code. The output from the PDS is a set of numbers which represent the program steps, and a listing of the input and output codes.

Obviously the PDS uses some special programs (software) for performing the tasks outlined. The programmer writes *source code* statements, submits them to the PDS via a program called the *editor*, then uses the *assembler* program to produce *object code* — the machine code used by the microcomputer. Errors in the program statements are printed along with the object code listing. Errors are corrected by editing the source code (via the editor) and resubmitting it to the assembler.

After a number of program modules are complete, a set of two or more may be "linked" together as a single program. This is done by submitting object code programs to the *linker*. The output from the linker is a single program which may be loaded into the microcomputer. 1◀

The list of support software is just beginning.

The following outline of software products describes the program development cycle further.

- Program development software
  - Editor — for entering and changing source code
  - Assembler — for conversion from symbols and mnemonics into machine code
  - Linker — for connecting several programs into one
  - PROM programmer — for loading numbers (programs) into PROMs
- Program testing software
  - Debug routines — for testing programs
  - AMPL system software — for testing programs and the interaction with the hardware
- Software available for use with user programs
  - Monitor — for checking status of all program modules
  - Executive — for overall control
  - Operating system — for operating peripheral devices
  - Library (utility) programs — for performing special mathematical conversions and calculations
  - High level language software for program development
    - PASCAL — for structured programming
    - POWER-BASIC — for ease of programming in BASIC language
    - FORTRAN — for general computer problem solving

This partial list of software is intended as a categorical outline which should indicate the level of support one finds in the areas of software development. To comprehend the value of any or all of these software products, you must work with them and develop a few programs for microcomputers.

The obvious difficulty with software evaluation is that few designers can afford the capital investment for a large PDS to properly evaluate each of the alternative paths for software development. But Texas Instruments has developed a variety of program development systems. Some of these are very economical and readily available. They were designed specifically for product and programming evaluation and for initial design.

You will find descriptions and approximate prices for each PDS in Chapter 2.

Which Microprocessor or Microcomputer to Use

You may be convinced that designing with programmable semiconductors is the best design philosophy, and you may be attempting to evaluate the various products on the market. But a significant decision point has been reached: "which microprocessor or microcomputer is best for my application?" The selection of the proper device is based on many factors, some of which are not related to architecture or instruction execution speed.

Selection of a microcomputer or microprocessor usually means selection of one primary vendor (and sometimes one or more second sources) who manufactures the product and the compatible peripheral devices. It also means the purchase of a program development system designed especially for the specific microprocessor. Selection of one device means a commitment to using that device for future designs. Changing to another microprocessor is costly both in hardware and in the development of programming skills.

Selection criteria for a microprocessor may be summarized as follows:

1. The microprocessor must be versatile so that it can be used in many applications.
2. The vendor must provide a comprehensive set of support and peripheral circuits.
3. One PDS should serve the programming activity for a significant period of time.
4. The cost of the devices and the PDS must be economically attractive.
5. The performance of the microprocessor must be sufficient to meet the design goals.

Texas Instruments 9900 family of components and software systems clearly meets these selection criteria. Careful evaluation of the price/performance tradeoffs between the various microprocessor products on the market will reveal superior adaptability of the 9900 family to any product design. The selection criteria applied to the 9900 family may be summarized thus.

1. Versatility has been achieved by providing a *family* of processors using one basic 16-bit architecture. Both 16- and 8-bit versions are available as well as a single chip microcomputer. Instead of trying to apply a single microcomputer to a broad scope of applications, the designer may select from the 9900 family the most appropriate microprocessor for each application. Programming and software support is the same for all.

2. Numerous support devices are available from Texas Instruments, and new products are introduced regularly. Chapter 8 contains detailed data sheets for many of them.

3. A single PDS from Texas Instruments can provide programming support for all of the processors in the 9900 family. The powerful support software streamlines program development. And programs written for one microcomputer may be used on another. Software compatibility of the processor family is one of the primary benefits of designing with the 9900.

4. Product pricing of the microprocessors, peripheral devices and PDS's is economically attractive. Designing multiple applications from the same components and using the same development tools means even greater economic benefit.

5. The 16-bit architecture — especially the bus width and the register size — enables the 9900 family of processors to achieve outstanding performance. Performance is measured by throughput and computing power, not by clock speed alone.

A complete evaluation of the 9900 in each of the above categories is not possible in so few words. But one specific feature of the product family should be included as a part of the evaluation. Memory-to-memory architecture, a unique computer concept developed by Texas Instruments for the 9900 minicomputer, is an outstanding feature because it enables the 9900 to achieve the most cost effective product development. The story of the evolution of this architecture will help you understand its importance.

EVOLUTION OF MEMORY-TO-MEMORY ARCHITECTURE

All things change with time, and computers are no exception. An evolutionary process has been at work in computer design since the beginning. Early machines were designed around a single accumulator which served as the focal point of most of the instructions. Steps such as load the accumulator (LDA), add to the accumulator (ADD), and store the accumulator (STA) were common in programs written for such machines. (The instruction mnemonics used here are simply illustrative and are not intended to be identified with any specific computer or microprocessor.) But there was a fundamental limitation—the bottleneck effect of forcing all transactions to be performed via a single accumulator *(Figure 1-12).*

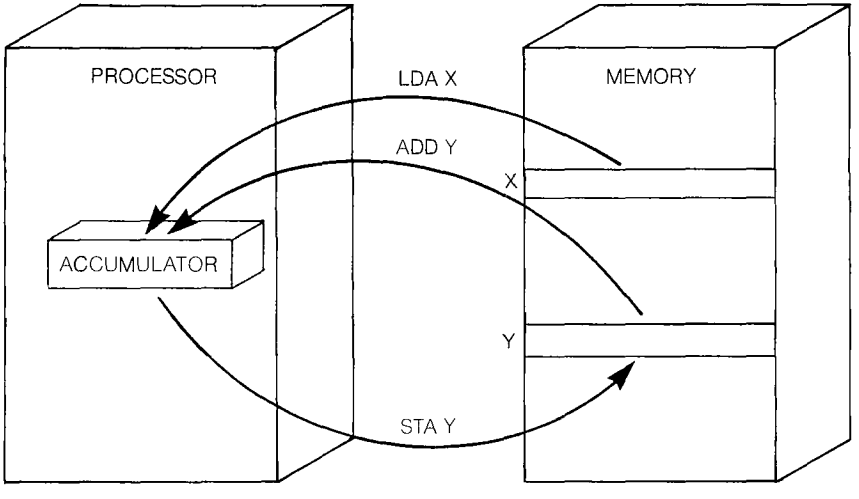


Figure 1-12. *Single Accumulator Architecture*

EVOLUTION OF MEMORY-TO-MEMORY ARCHITECTURE

As circuit elements became less expensive, especially through the introduction of integrated circuits, multiple accumulator architectures emerged *(Figure 1-13)*. A and B accumulators were the focal points of expanded instruction sets which allowed loading either accumulator (LDA, LDB), adding to either accumulator (ADA, ADB), and storing either accumulator (STA, STB). With this design came the increased use of an accumulator for holding the address of an operand, adding flexibility and power to the instruction set and to the architecture.

It should be clear at this point that the instructions, the dictionary of words used by a computer to implement the ideas of the programmer, are as much a part of the architectural fabric as the registers, the control unit or the bus structure. In fact, by implementing instructions as strings of microinstructions stored in an on-chip control ROM, microprocessor designers have created the opportunity for increasing instruction set power through microprogramming.

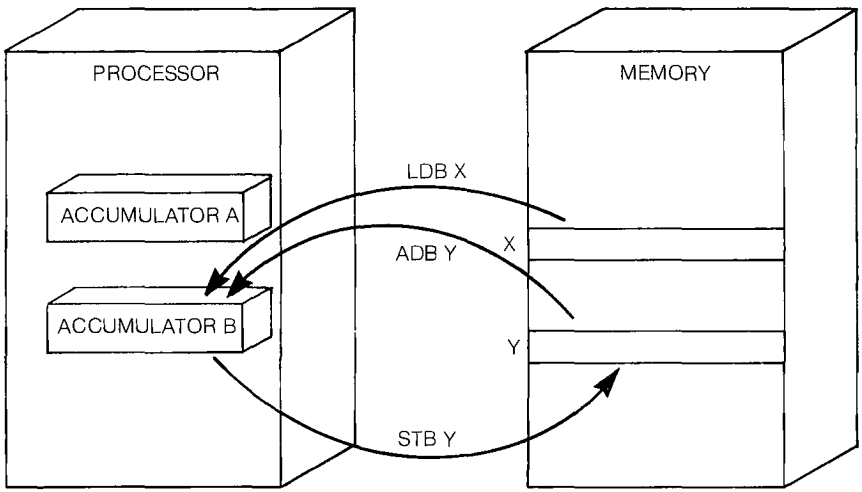


Figure 1-13. Multiple Accumulator Architecture

The next major step in the architectural evolution was the design of machines based on a set of general registers which could be used as accumulators for numerical operations or for storage of operand addresses *(Figure 1-14)*. The expanded capabilities allowed increased flexibility not only in arithmetic functions but also, and more importantly, in the generation of operand addresses via indirect addressing, and indexed addressing.

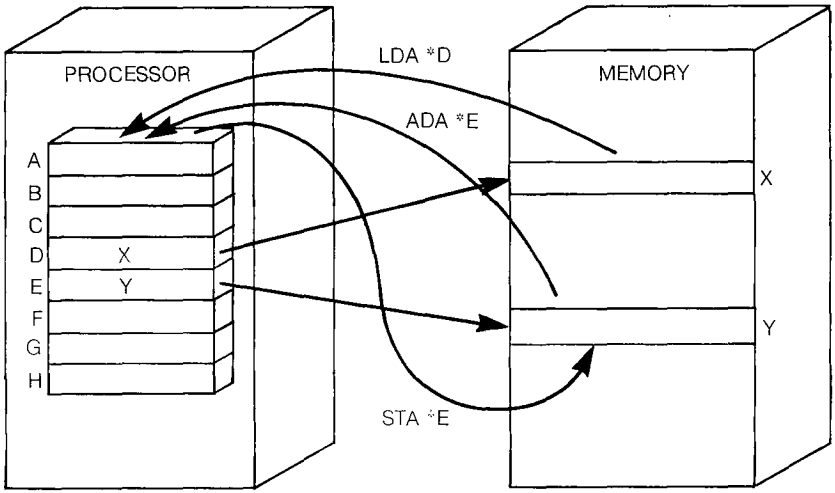
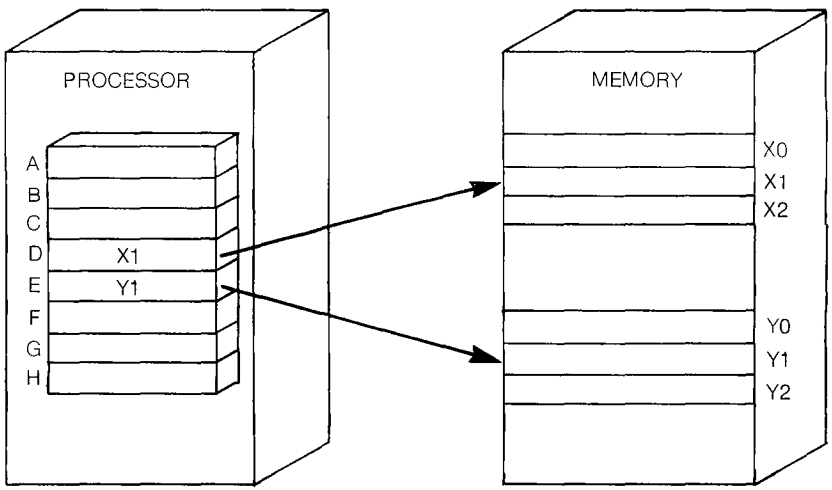


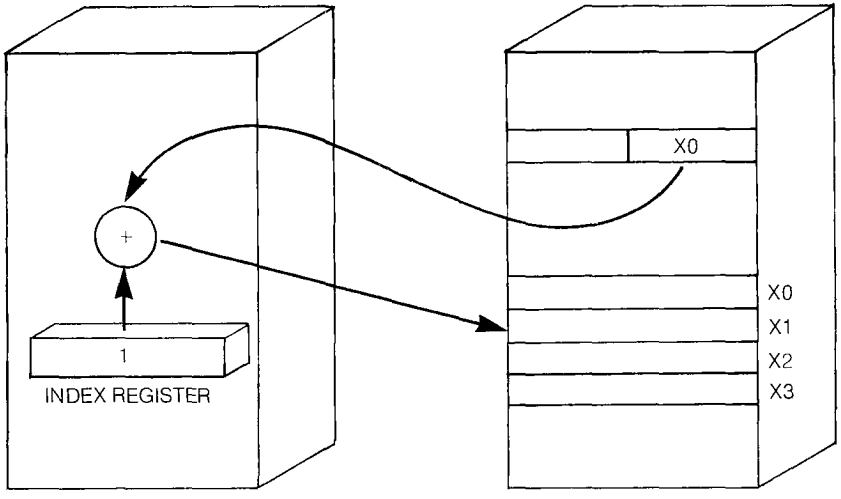
Figure 1-14. General Register Architecture

Perhaps it is well to digress for a moment and explain these terms. Indirect addressing allows a register to serve as a pointer to identify specific elements in a table or an array of data *(Figure 1-15)*. Instructions for an arithmetic operation may be used over and over, with the pointer (or pointers) being adjusted to access different values for each pass. In indexed addressing, the instruction contains a base value while an index register holds the displacement value *(Figure 1-16)*. The base value locates the table, and the index register contains the number of the element in the table (one, two, three, etc.). The base value must be added to the contents of the index register to obtain the actual memory address.



Registers D and E contain the addresses of operands X1 and Y1. D and E may be incremented to address sequential elements in tables.

Figure 1-15. Indirect Addressing



X0 is the address of the first element in the table.

X1 is obtained by adding X0 to the 1 in the index register.

The index register may be incremented to address sequential entries in the table.

Figure 1-16. Indexed Addressing

The general register architectures were made economically feasible by the expanded capabilities of integrated circuits through the technological advancements of Medium scale integration (MSI) and large scale integration (LSI) *(Figure 1-17)*. As more and more circuits were implemented on a chip, it became feasible to expand from two accumulators, to a general register file, to the general register file on a single LSI microprocessor chip.

In discussing LSI, one must not fail to recognize that the single most important impact of LSI is in the development of memories. More bits per unit area of silicon means higher capacity and lower cost, generally without sacrificing speed. The advent of microprocessors was the natural evolutionary step in the utilization of memory for a greater variety of logic and control applications.

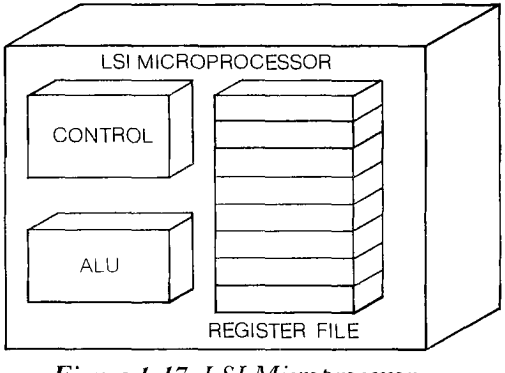


Figure 1-17. LSI Microprocessor

In looking toward the future of memories and microprocessors, the technologists see the implementation of an ever increasing number of memory cells and microprocessor calculation and control functions on an ever-shrinking area of silicon *(Figure 1-18).* Registers and memory cells are virtually identical in their implementation at this point, so the words *register* and *memory* no longer connote high speed and low speed storage. In fact, memory speed is rapidly approaching register speed *(Figure 1-19).*

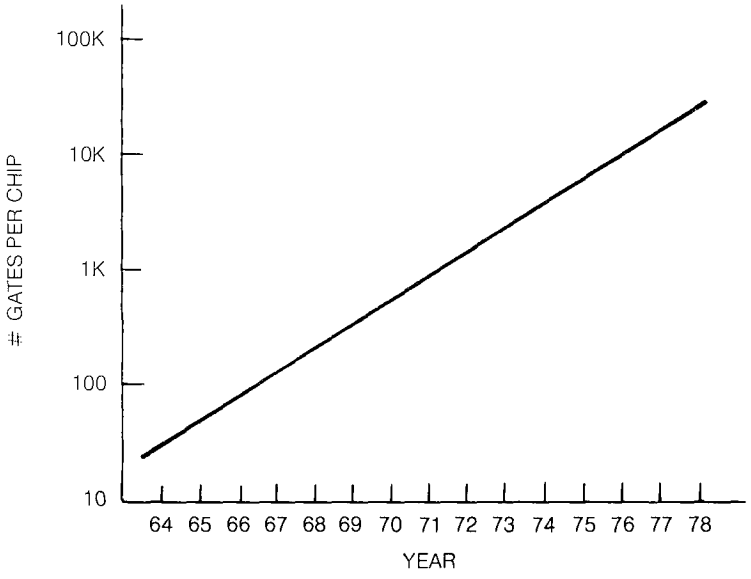


Figure 1-18. *Trend in Gates per Chip*

In view of this convergence of memory speed and register speed, the architects of the 990 minicomputer (from which the 9900 is derived) envisioned an architecture in which the instructions are written with respect to *memory words* rather than *registers.* The architectural concept, called memory-to-memory architecture, was the basis for a new computer design in which all memory reference instructions operate on one or two words of memory and store the result before going on to the next instruction.

There were actually two major reasons for developing such an architecture. First, since all instructions would reference words of memory and complete their cycles by placing results in memory, there would be no requirement for register-save operations in a multitask or interrupt processing environment. Second, while this approach might at first be slightly slower in some cases, the architects envisioned that the technological evolution would continue to narrow the gap between register speed and memory speed, and in the long run this minor disadvantage would vanish.

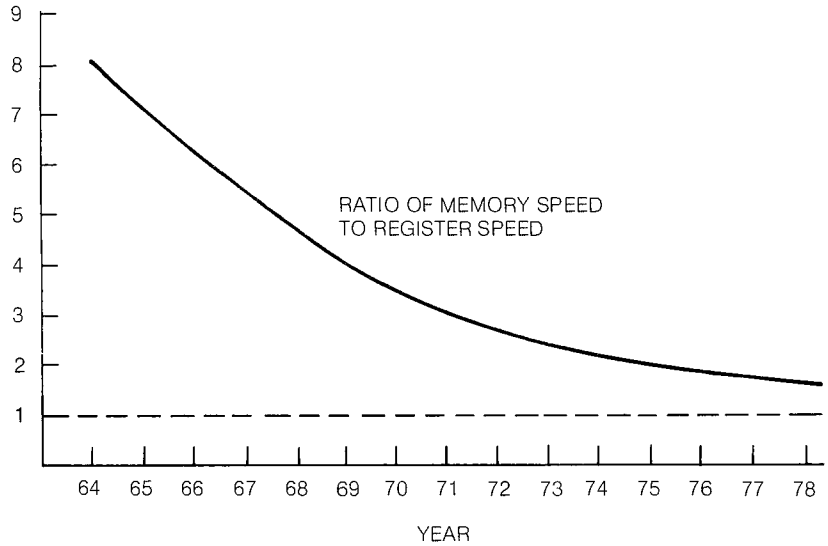


Figure 1-19. Ratio of Memory Speed to Register Speed

Another important advantage of this new architecture, often overlooked, appears to be an even stronger and more important justification for the development of this radical departure from conventional computer architectures. When one instruction can identify two memory words or operands, perform an operation, and store the result in memory, it will replace common sequences such as LDA, ADA, STA found in the instruction sequences of all accumulator-based machines. Furthermore, a single instruction can access additional memory words for use in addressing operands and can even increment pointers and employ index registers all as a part of its execution sequence. If a single instruction can do all this, then the writing of instruction sequences, programming, must be substantially easier. Fewer lines of code are required. (In data manipulation and address computation sequences, the reduction is typically 3:1.) Support software, such as monitors, executives, and operating systems can be smaller, easier to use and understand, and will consume less memory.

For these reasons, benchmarks written to compare the 9900 architecture with conventional register file based microprocessors show the advantage of the 9900's memory-to-memory architecture in three important categories: the number of bytes of memory required, execution speed, and the number of instructions written to accomplish a given task *(Figure 1-20)*. The 9900 comes out ahead in all three categories.

	Program memory requirements (bytes)				Assembler statements				Execution time (microseconds)			
	9900	A	B	C	9900	A	B	C	9900	A	B	C
Input/output handler	24	38	28	17	9	14	17	7	71	154	79	49
Character search	22	24	20	18	8	10	9	8	661	1636	760	808
Computer go to	12	12	17	14	5	5	11	8	98	352	145	145
Vector addition: $A_N \leftarrow B_N = C_N(16)$	20	30	29	46	5	14	20	22	537	2098	1098	1866
Vector addition: $A_N \leftarrow B_N = C_N(8)$	20	32	23	40	5	15	14	22	537	2108	738	936
Shift right 5 bits	10	6	19	20	3	3	12	9	22	56	137	81
Move block	14	18	16	34	4	9	9	16	537	1750	1262	2246
Totals	122	160	152	189	39	70	92	92	2464	8154	4219	6131

Figure 1-20. Benchmark Comparison of 9900 vs. Other Microprocessors

One final note about architecture. Memory-to-memory architecture and instructions in the 9900 do not sacrifice the concept of "registers" as they are conceived in the architectures with general register organizations. The general "register file" is conceptually retained as a block of sixteen words of memory *(Figure 1-21).* Over two thirds of the instructions in the 9900 refer in one way or another to this "register file" in much the same way as prior architectures referenced the general register file in the CPU. Thus, base addresses, subroutine linkage and interrupt save operations can all be accomplished via the "register-file-in-memory" concept.

By using memory for the register file, the advanced memory-to-memory architecture allows new programming flexibility. There is a way to identify *multiple register files* in 9900 based systems *(Figure 1-22).* Each basic process can have its own set of "registers." *There is no limit (except memory size) to the number of "registers" available* for use in programming the functions of a particular application.

The memory-to-memory architecture of the 990 and 9900 products is clearly revolutionary and innovative. Programming effort for the 9900 is typically less than half that for any other microprocessor currently available because the instructions operate on words of memory and store results automatically. This means not only that programs consume less memory, but execution speed (for the 16-bit processors) is faster than that of other processors.

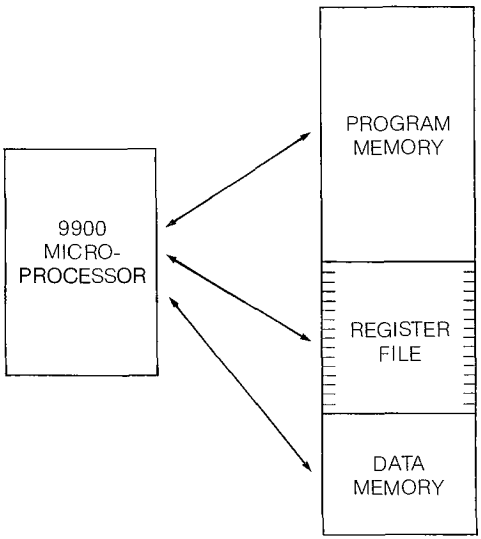


Figure 1-21. Register File in Memory

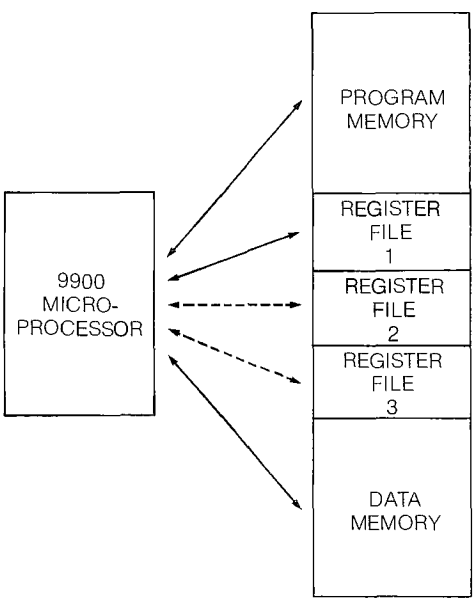


Figure 1-22. Multiple Register Files in Memory

GETTING UP TO SPEED ON MICROPROCESSORS

By now you may be convinced that this book contains a great amount of information about microprocessors and microcomputers, but you may feel that you are not as well prepared to understand it as you would like to be. This section has the answer. Here are the steps you should take to learn about microprocessors and microcomputers. The knowledge gained will help you in all new designs and will be especially helpful in designing with the 9900 family of processors and peripherals.

Few people have had the opportunity to learn about microcomputers in college. In fact, schools and colleges exist primarily to teach you *how to learn*, and not to teach you everything you need to know to do a particular job. Your effectiveness in performing any job is directly related to your willingness to acquire new specialized knowledge in your particular field. This book will serve as one source of specialized knowledge in the field of microcomputers, but it is focused on the 9900 family. And you may require additional education in this field before achieving a full understanding of the material presented.

It may be that knowledge of MOS and I^2L technologies is needed for a clearer understanding of interfacing techniques. Basic computer fundamentals, such as storage of data and programs and the sequential operations may be an area you would like to study. It could be that you feel a need to improve your understanding of programming and the concepts of building programs via the modular approach. The list of specialized areas within the field of microcomputer technology can be quite long.

Technology advances so rapidly today that it seems virtually impossible to keep up, much less catch up. But you can do both, and without spending an inordinate amount of time. To acquire specialized knowledge in any field, you should devote 30 minutes a day to reading books or periodicals which contain the information you need. Advising you on the implementation of such a program is not the intent of this section. You know where you are and where you are going. What you need is a clear path or plan of action to achieve the goal: the acquisition of specialized knowledge about microprocessors and microcomputers.

The first step is to find authoritative texts on the various subjects in the field. This chapter contains a bibliography of texts and periodicals from which to begin your search for new information. Get your hands on these books and articles. Review them for general content and readability, then decide which ones are best suited to your needs. Set up a plan to read one or more of these books in a definite period of time, devoting a *scheduled*, uninterrupted period of 30 minutes a day to this program. Take notes while you are reading and (if the book belongs to you) underline the information which is especially important to you.

BIBLIOGRAPHY

As you are getting up to speed, you will become aware of certain periodicals that contain articles most directly suited to your background and experience. Subscribe to one or more of these or be sure to obtain each issue as it is published so that you are not only reading about fundamentals, but current topics, the latest improvements in devices and systems.

Set up the goal, the plan of action; and then, above all, form the habit of reading for 30 minutes a day. Few people can set up such a plan, and fewer still can continue to execute it for long periods of time. But if you persist, you can learn, not just one, but *all* facets of design with microprocessors and microcomputers, and in time you will achieve the success you desire.

BIBLIOGRAPHY

Books

Altman, L., *Microprocessors*, Electronics Book Series, McGraw-Hill, 1975

Bartree, T. C., I. L. Lebow, and I. S. Reed, *Theory and Design of Digital Machines,* McGraw-Hill, 1969

Bibbero, R., *Microprocessors in Instruments and Control,* John Wiley, 1977

Blakeslee, Thomas R., *Digital Design with Standard MSI and LSI,* John Wiley, 1973

Gear, C. William, *Computer Organization and Programming*, McGraw-Hill, 1969

Greenfield, Joseph D., *Practical Digital Design Using IC's,* John Wiley, 1977

Hansen, P.B., *The Architecture of Concurrent Programs,* Prentice-Hall, Inc., 1977

Hansen, P.B., *Operating System Principles,* Prentice-Hall, Inc., 1973

Knuth, D. E. *The Art of Computer Programming, VOL I, Fundamental Algorithms*, 2nd Edition, Addison-Wesley, 1973.

Knuth, D.E. *The Art of Computer Programming, VOL II, Semi-Numerical Algorithms,* Addison-Wesley, 1969

Knuth, D.E. *The Art of Computer Programming, VOL III, Sorting and Searching,* Addison-Wesley, 1973.

Luecke, G., J. Mize, W. Carr, *Semiconductor Memory Design and Application,* McGraw-Hill, 1973

Malrino, A., *Digital Computer Electronics,* McGraw-Hill, 1977

McWhorter, G., *Understanding Digital Electronics,* Texas Instruments Learning Center, 1976

Morris, R. L., J. D. Miller, *Designing with TTL Intergrated Circuits*, McGraw-Hill, 1971

Norris, B., *Power Transmission and TTL Integrated-Circuit Applications,* McGraw-Hill, 1977

Silver, G., *Computer Algorithms and Flowcharting,* McGraw-Hill, 1975

Sloan, M.E., *Computer Hardware and Organization,* Science Research Associates, Inc., 1976

Solomon, L., *Getting Involved with Your Own Computer; A Guide for Beginners,* Ridley Enslow Publishing, 1977

Soucek, B., *Microprocessors and Microcomputers,* John Wiley, 1976

Torrero, E., *Microprocessors, New Directions for Designers,* Electronic Design, Hayden, 1975

Wester, John G. and William D. Simpson, *Software Design for Microprocessors,* Texas Instruments Learning Center, 1976

Williams, Gerald E., *Digital Technology,* Science Research Associates, Inc., 1977

Zaks, R., *Microprocessors: From Chips to Systems,* Sybex, 1977

Staff of the Texas Instruments Learning Center, *Understanding Solid-State Electronics, 3rd Edition,* Texas Instruments Learning Center, 1978

ARTICLES

Barna, Arpad, and Dan I. Porat, *"Integrated Circuits in Digital Electronics"*, John Wiley, 1973

Reid-Green, K.S., *"A Short History of Computing"*, Byte, Vol. 3, No. 7, July 1978

Special Issue on Microelectronics, *"Scientific American"*, Vol. 237, No. 3, September 1977

Electronic Business, *"New Rules in an Old Game"*, Vol. 18, No. 6, June 1978

LIST OF PERIODICALS TO BE MONITORED

ACM Computing Surveys, The Survey and Tutorial Journal of the Association for Computing Machinery, ACM, Inc., Mt. Royal and Guilford Avenues, Baltimore, MD 21202

EDN, Cahners Publishing Co., 270 St. Paul St., Denver, Colorado 80206

Electronics, McGraw-Hill Inc., 1221 Avenue of the Americas, New York, N.Y. 10020

Electronics Design, Hayden Publishing Co., 50 Essex St., Rochelle Park, N.J. 07662

IEEE Spectrum, The Institute of Electrical and Electronics Engineers, Inc., 345 East 47 Street, New York, N.Y. 10017

Interface Age, McPheters, Wolfe & Jones, 16704 Marquardt Avenue, Cerritos, CA 90701