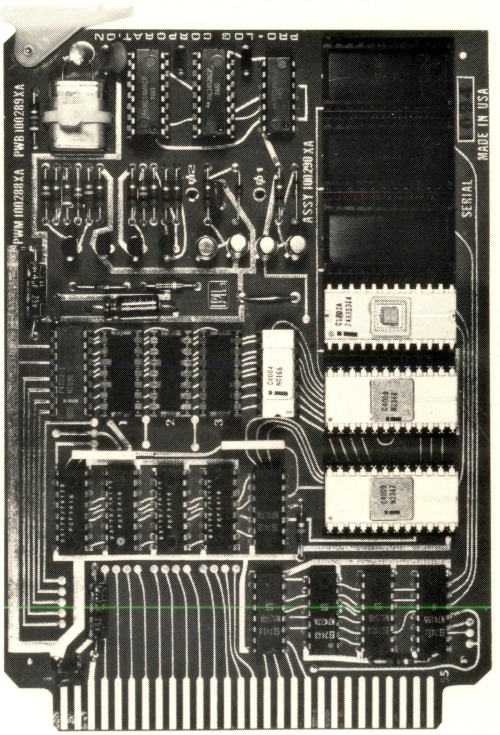
MICROPROCESSOR USER'S GUIDE

HOW TO SELECT AND USE MICROPROCESSORS

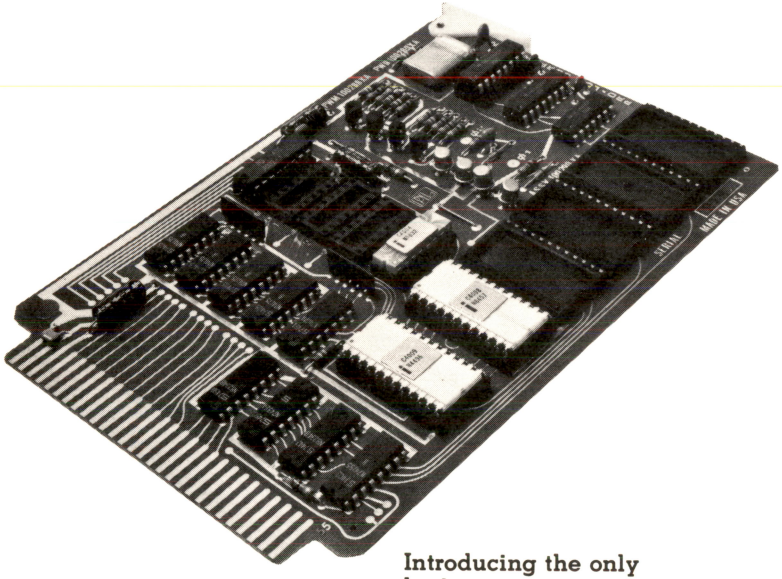


PRO-LOG CORPORATION

2411 Garden Road Monterey, California 93940 Telephone (408) 372 - 4593 TWX: 910·360·7082 ▪ 101145 9/75

Why start from scratch with microprocessors.

We've already developed the subsystems, instruments and education you need.



What would it cost you to design and build our new PLS-401A logic processor system yourself?

Introducing the only logic processor system priced under $100*.

It's our one-card 4004-based PLS-401A.

It includes a microprocessor; crystal controlled clock; 80-character RAM; built-in power-on reset; 16 lines of TTL input; 16 lines of TTL output; and sockets for 1024 words of program memory.

Quantity	Price
10-24	$175
25-99	150
100-499	125
500 and up*	99

4004, 4040, 8008, 8080, and 6800 logic processor and microcomputer cards.

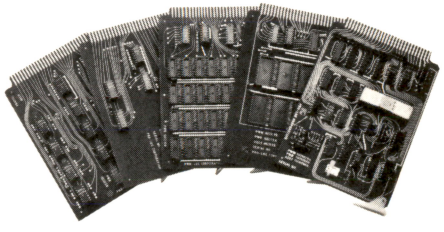
Off-the-shelf delivery on one-, two-, and three-card 4004 and 4040 logic processors.

Off-the-shelf delivery on three- and five-card 8080, 8008 and 6800 microcomputers.

All our systems are implemented to use 1702A MOS PROMs or equivalent.

We also have a wide line of input and output interface cards compatible with all our microprocessor systems.

For customers who order 250 systems, we throw in free a complete set of manufacturing and assembly plans allowing you to build your own hardware, relying on us as an established and dependable second source.



Five card 8080 microcomputer system. Includes microprocessor, 256-word instruction PROM with 2048-word capacity, and 1024-word program or data RAM with 4096-word capacity.

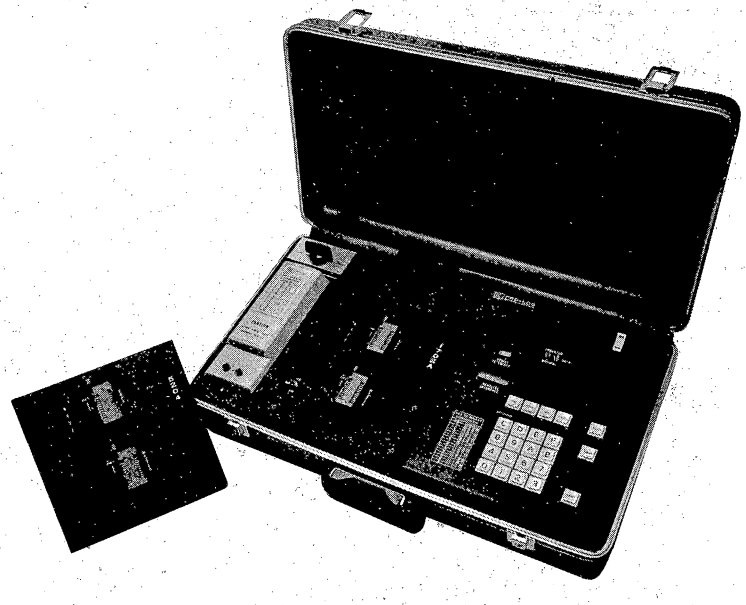
Series 90 PROM Programer

Lets the design engineer program or duplicate any MOS or bipolar PROM directly from a master PROM, keyboard, teletype, paper tape reader, or computer.

Automatically and quickly programs production line quantities of PROMs.

Lightweight and fully portable for field service work.

Through the use of plug-in personality modules, the Series 90 can program any AMD, Harris, Fairchild, Intel, Intersil, Monolithic Memories, Motorola, National, Signetics, or Texas Instruments bipolar or MOS PROM—in short, any PROM made.

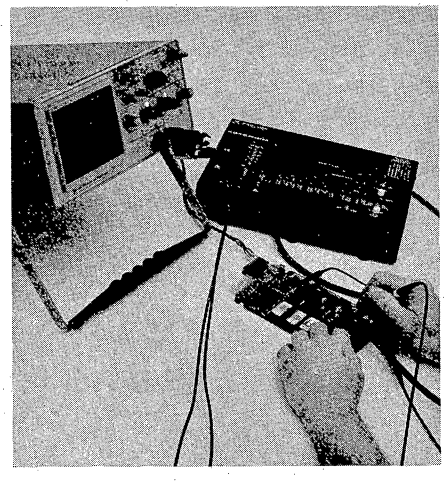


Microprocessor System Analyzers.

Instruments available for designing, troubleshooting, and testing both program and hardware in systems using 4004, 8008, 4040, 8080, or 6800 microprocessors.

They eliminate the need for control panels, diagnostic routines or other data processing tools for testing microprocessor-based systems.

Used in conjunction with a standard oscilloscope, they can test both program and hardware either together or individually. The analyzers display all data related to a selected instruction cycle and generate a scope sync pulse. They interface to the system under test through use of a DIP connector that clips onto the microprocessor. They can be attached to or detached from your system in a matter of seconds.



Education for the decision maker and the design engineer.

Our half day applications course for decision makers takes a hard look at the design and function of microprocessors in real world applications.

We've also got a three day hands-on course we've given to more than 1,000 design engineers in the past two years. The only prerequisite is that you know what a flip-flop and a gate are. If you do, we guarantee you'll come out of our course knowing how to design, program and use microprocessor modules because you'll have done it.

Contact Pro-Log for a complete list of course schedules and locations.

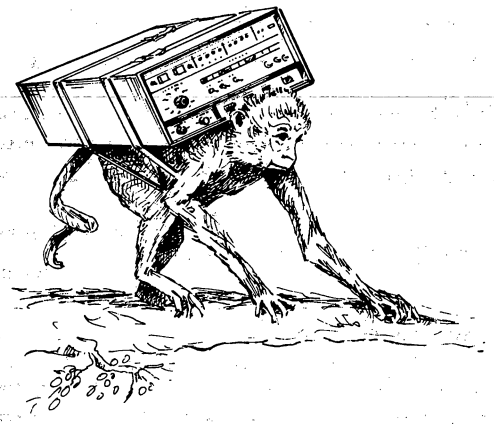
Ask for a free copy of our "PROM User's Guide."

Are you chasing TECHNOLOGY instead of PROFITS?

HOW to PROFIT from MICROPROCESSORS

A hard core look at the design and function of microprocessors in real world applications.

EQUATION FOR PROFITS: MICROPROCESSOR + PROM = UNIVERSAL LOGIC ELEMENT



Do you know the feeling?

A HALF DAY COURSE FOR DECISION MAKERS

This session has been specifically designed to meet the needs of **CORPORATE MANAGEMENT, ENGINEERING MANAGEMENT** and **DESIGN ENGINEERING** personnel.

If you think a microprocessor is "just a small computer" . . . you will:

- Turn design over to programmers.
- Waste money on "software".
- Waste time talking to your "software".
- At least **double** your hardware costs.
- Not have an effective way to service your systems in the field.
- Scare off the engineers who should be using the microprocessor in the first place.

TOPICS COVERED

HOW TO SELECT THE BEST MICROPROCESSOR FOR YOUR PROJECT. Includes a set of economical and technical screening tools to judge existing and future microprocessors on the basis of your priorities.

HOW TO DESIGN WITH MICROPROCESSORS. A step by step design approach from product specification thru field-trials to final production which is completely analagous to hardwired logic design. This technique can be learned and effectively used by design engineers within one week.

HOW TO SELECT PROMS AND ROMS. PROMs and ROMs are the key to the design revolution. Some ROMs now cost less than 0.1¢ per bit.

EACH PARTICIPANT RECEIVES:

- "Logic Processors for Dedicated Control" by Matt Biewer.
- "How to Design with Microprocessors" by Ed Lee.
- Prints of key slides from the presentation.
- Microprocessor literature

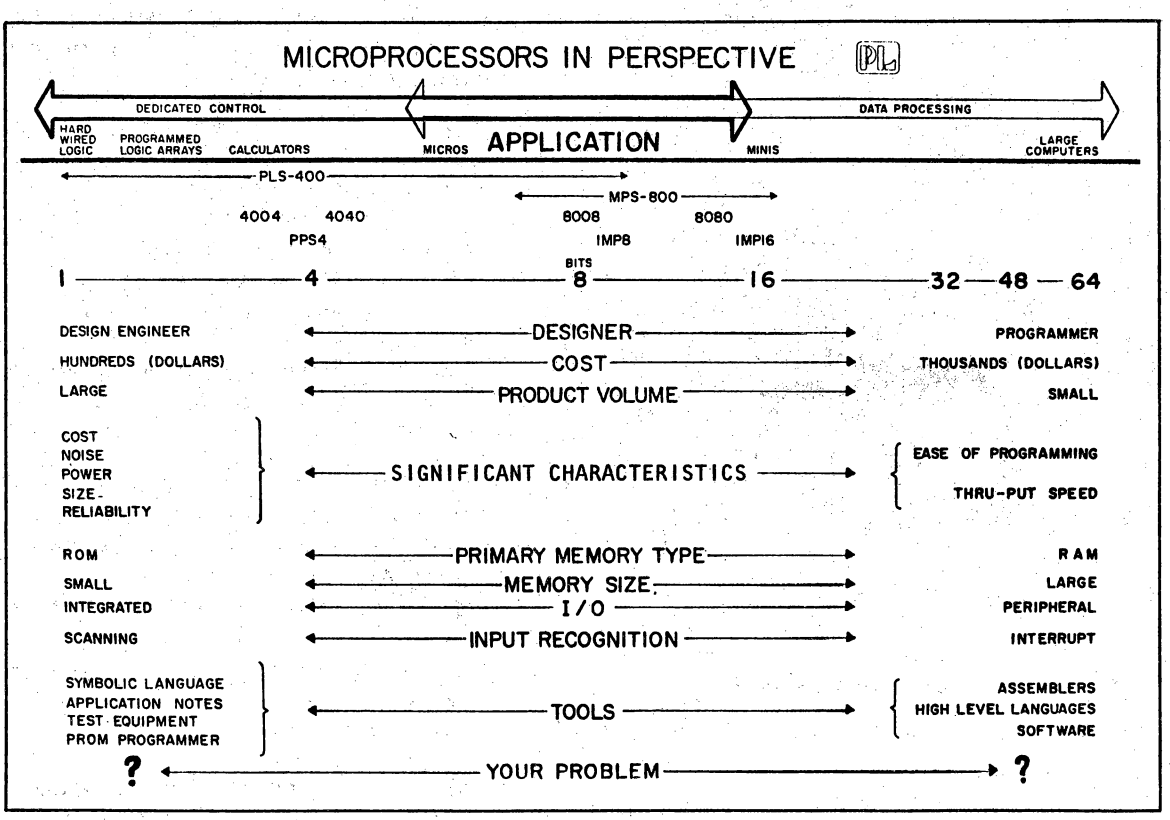
MICROPROCESSORS FOR DEDICATED CONTROL

By Matt Biewer

Presented Initially at WESCON 74

Revised for Presentation at WESTEC 75

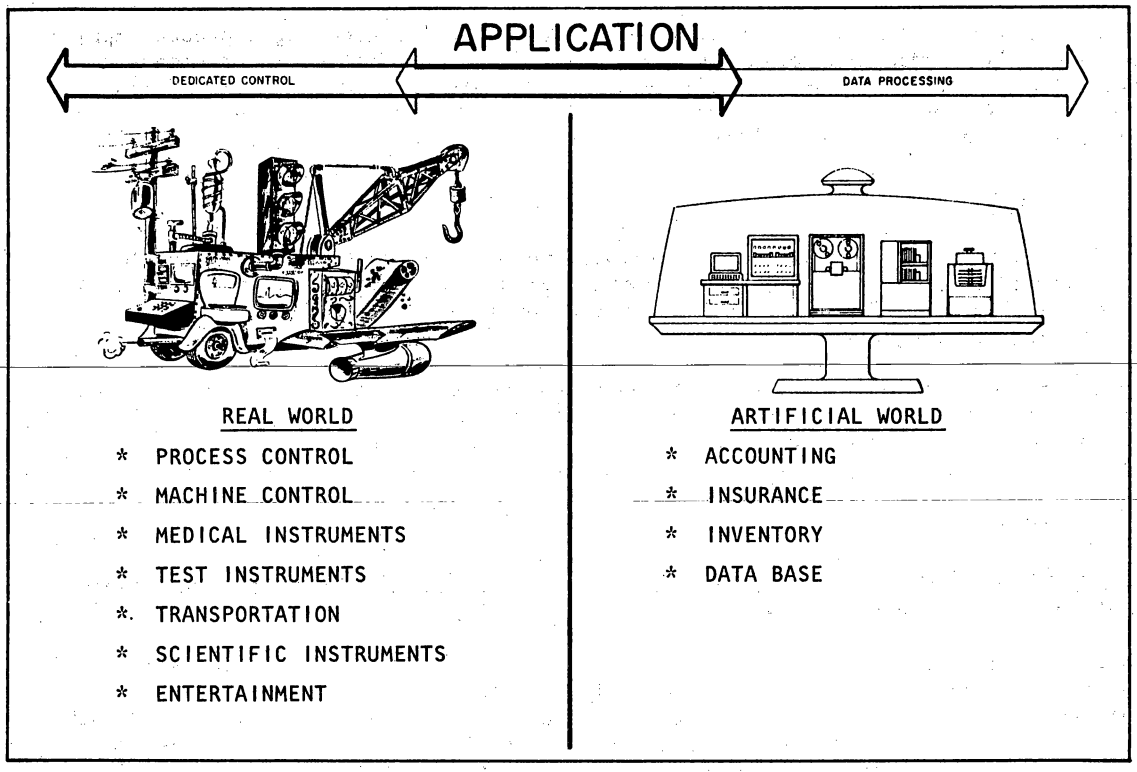
Microprocessors can be discussed intelligently only after identifying their market role. There is a spectrum of microprocessor applications which ranges from dedicated control to data processing. The spectrum as shown puts the microprocessor role into perspective by considering techniques, characteristics, tools and architecture needed for each type of application.



The computing nature of microprocessors implies a use in the data processing industry. The actual proven use of microprocessors, used in combination with Read Only Memory (ROM), is in the hard-wired logic replacement market as a logic element. This paper clarifies the Microprocessor/ROM memory role as a logic replacement element. It shows that a specialized microprocessor designed as a logic processing element is needed and the architectural requirements for a logic processor are given.

APPLICATION

At the one end of the spectrum, hard-wired logic by its nature has been used to solve the dedicated control problems. At the other end, large computers by their nature are used for data processing.



Data processing, with large volumes of data to be manipulated and a job that may change from hour to hour, is accomplished using a system with significant read write memory and canned programs to switch from job to job. Because of the undefined job requirements the system is normally configured to perform the largest anticipated job. This results in an installation which, to pay for itself, must run as many jobs as possible. Productivity is emphasized using the operational techniques of program loading and multiprocessing, and the hardware capabilities of interrupt and DMA.

Present microprocessors, with shortcomings in satisfying the data processing productivity requirement, make very poor computer replacements. The emphasis on a microprocessor for the data processing market must satisfy the two major requirements of use of existing software and thru-put.

In contrast, a dedicated control application is one where a piece of hardware performs a limited task repeatedly on demand. When power is applied it does a job. There is little or no need to process volumes of data. Every job is unique and a canned program isn't available. The job is precisely defined and either has a requirement of very low cost for high volume market or it must be easy to design and change because of its one-of-a-kind nature.

Microprocessors with ROM memory satisfy a wide range of random logic product applications. They offer cost advantages and ease of use for fast market response to high volume products, plus flexibility and a standard approach to one-of-a-kind products. In addition, microprocessors satisfy random logic design needs, such as low power, small size, and reliability.

PRODUCT COST

Product costs in data processing are related more to memory and peripheral I/O than to the cost of the processing element. Significant recurring costs are also involved with software. Microprocessors provide insignificant impact to cost in the data processing environment.

In the high volume product market of dedicated control, cost is the single controlling factor. Cost related hardware items such as package size, printed circuit board space, interconnects, cabling and power consumption become important. Microprocessors tend to satisfy the cost constraints for dedicated control designs.

WORD SIZE

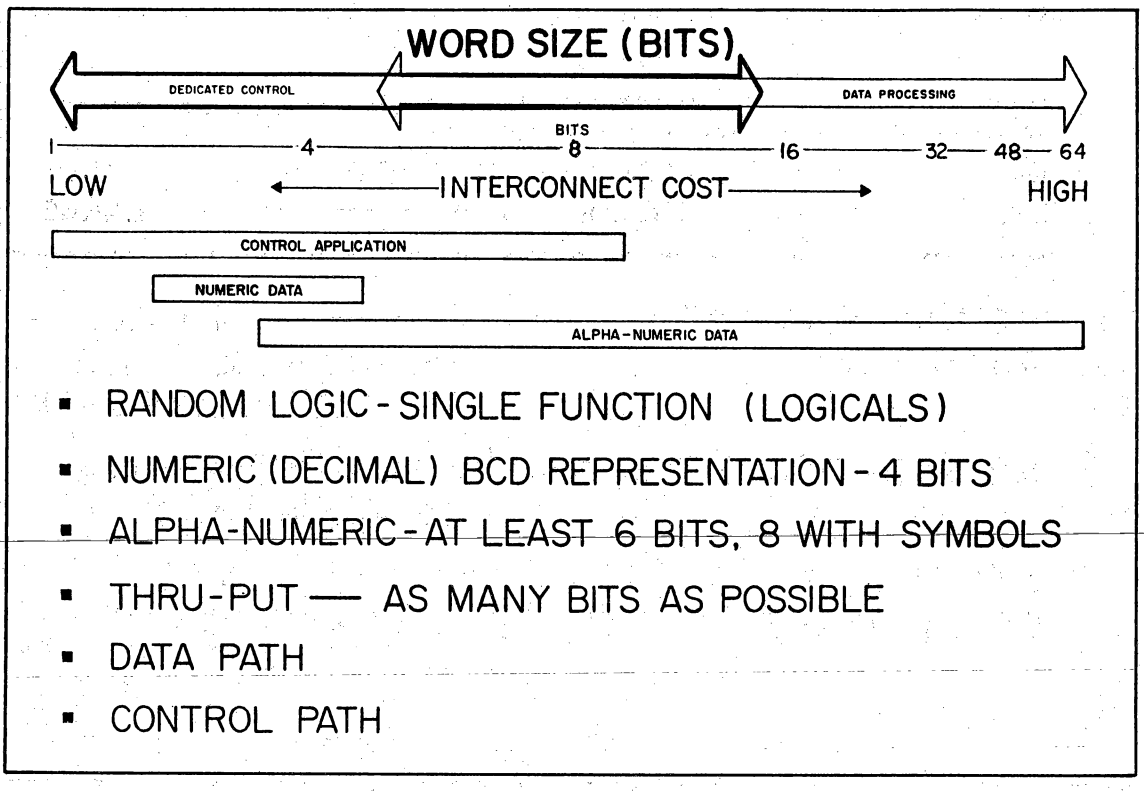
Data processing puts no restriction on word size as long as thru-put is not compromised. Generally the larger the word size the higher the thru-put.

The cost constraint on effective logic processing systems requires that word size and thru-put be compromised. The compromise is made to keep packaging and related interconnection costs low. Word size, thru-put, and cost are interrelated through the data, instruction, and addressing paths of the microprocessor. The best compromise is a word size and architecture which balances speed of operation with package pin count.

The data path is defined as the transfer bus for input/output and data handling operations. Data path width is suggested by application where random logic suggests individual bit manipulation. Numeric operations such as calculators suggest a 4-bit width for BCD representation. Alpha-numeric data handling suggests an 8-bit representation and scientific processing suggests larger byte sizes.

Regardless of what byte size the applications suggests, data path width is actually only limited by the speed desired in performing the operation. A one bit microprocessor can handle a sixteen bit operation and, in the same sense, a sixteen bit microprocessor can do individual bit manipulation. The real tradeoff is between speed and cost.

The instruction path is defined as the transfer bus for retrieving instructions from the program memory. Instruction word width is determined by the size of the instruction set which affects processing power.

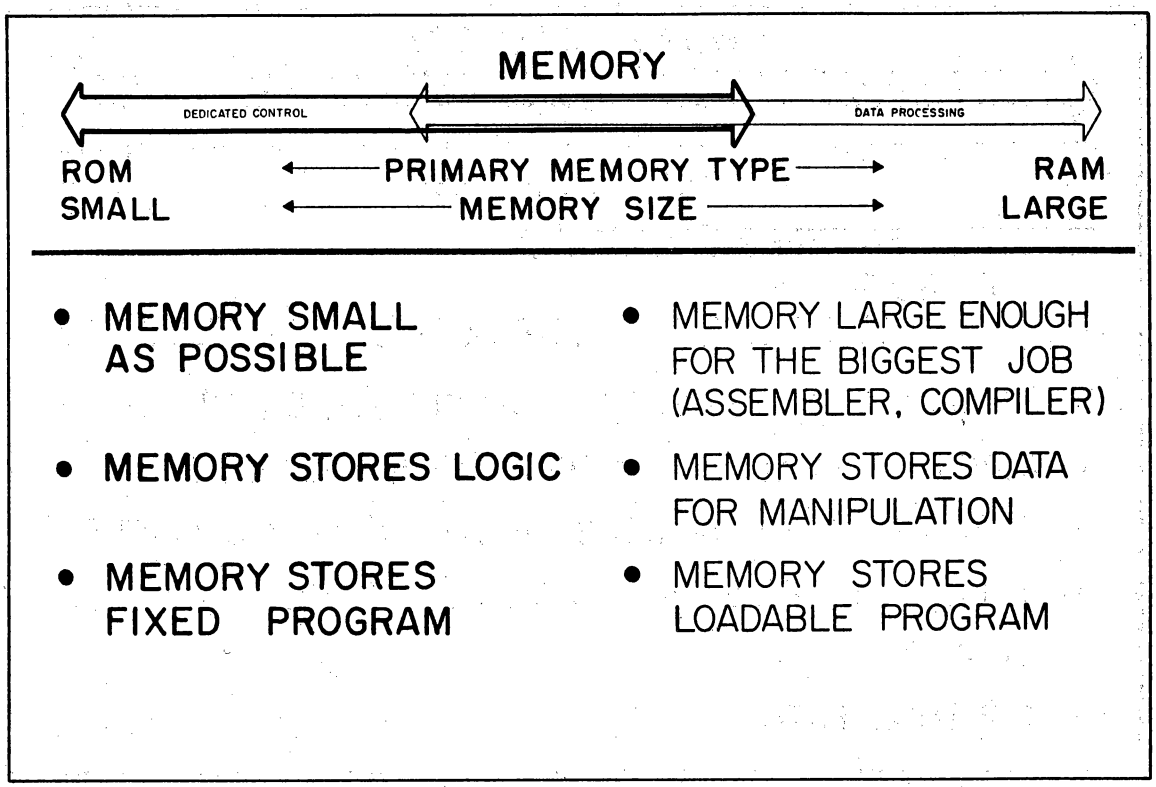


Data processing demands an instruction set which satisfys the thru-put requirement. In logic processing the cost constraint requires that the word size/thru-put compromise prevail. The data and instruction paths are combined and a reasonable trade off is made on the instruction set and data word size to satisfy pin limitations and interconnection costs.

The address path is the selection path for memory and I/O data. For data processing memory and I/O often use separate addressing or selection schemes. This suits the need for maximum memory and extensive peripherals. For logic processors a combined addressing path for memory and I/O is the most efficient. Interconnection is simplified and the package pin limitation is not overextended.

MEMORY TYPE

Data processing requires large segments of read write memory to solve the data manipulation type of problem. This memory is generally used for loadable program storage and bulk data storage. The fixed nature of dedicated control designs needs only ROM program storage, which deserves as much credit for the hard-wired logic replacement revolution as the microprocessor. Dedicated control systems previously hard-wired to perform the logic functions can be designed with the same permanence and yet enjoy the standardization and flexibility of storing logic sequences in ROM.



Dedicated control applications that require bulk data storage, can use economical storage treated as I/O rather than expensive RAM memory. The usual lack of a speed requirement allows the use of inexpensive shift registers. Generally, total design costs benefit from the packaging efficiency of these devices.

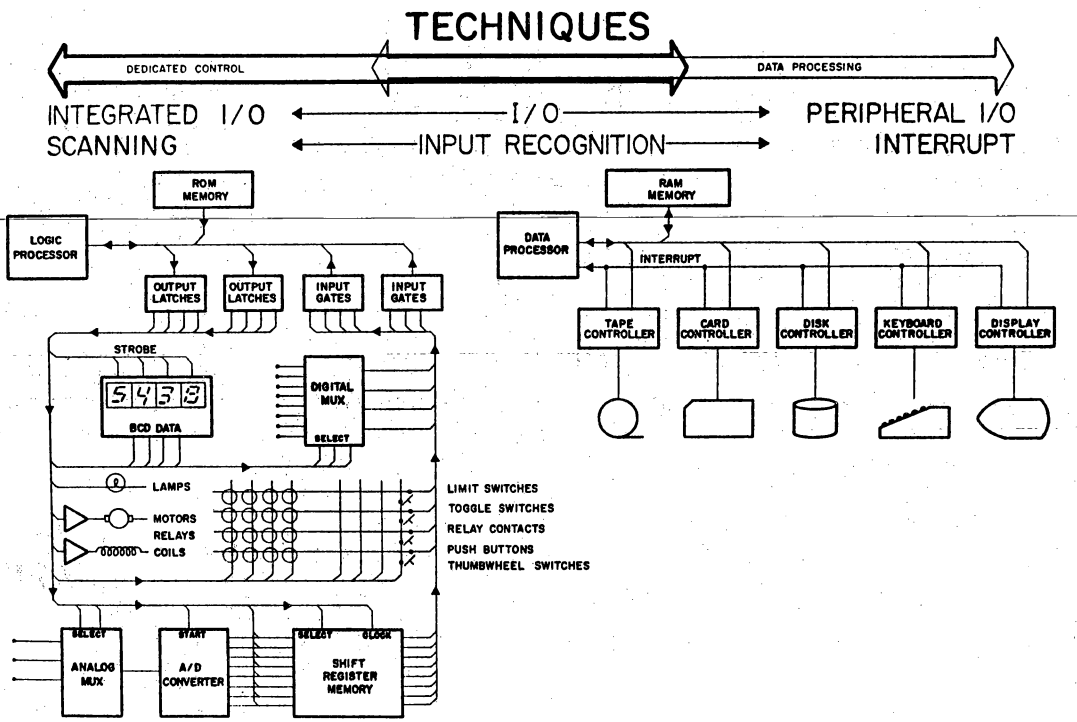
MEMORY SIZE

The undefined nature of the data processing problem requires systems with sufficient memory capacity even for the unanticipated job. It is not uncommon to find 64 K bytes of memory on a data processing system. The fixed requirement of dedicated control defines precisely the logic storage needs. Many dedicated control jobs can be done with 1,000 or less words of programmed logic. The cost constraint of dedicated designs usually dictates minimizing memory size.

Memory size relates directly to instruction addressing efficiency. Addressing 64 K bytes of memory requires 16 bits of address. This can grow to 24 bits when used with an instruction to make a decision, using a subroutine or for handling memory data. In contrast, an efficient instruction set with only twelve bits of address needs only 16 bits total, providing adequate program storage capacity for dedicated control applications. Small memory and efficient memory addressing are requirements for microprocessors in dedicated control applications.

I/O

Input/Output on large computers is generalized to handle peripheral I/O equipment. Communication with the I/O device is accomplished through a controller with canned software. In contrast, dedicated control functions are implemented with a tight-knit integrated I/O structure where the functions are directly manipulated by the microprocessor. Keyboards and switches are scanned, displays are loaded, A/Ds and D/As are directly controlled, etc. This programmed control of every system element is the key to the cost effectiveness and flexibility of using microprocessors for hard-wired logic replacement.



TECHNIQUES

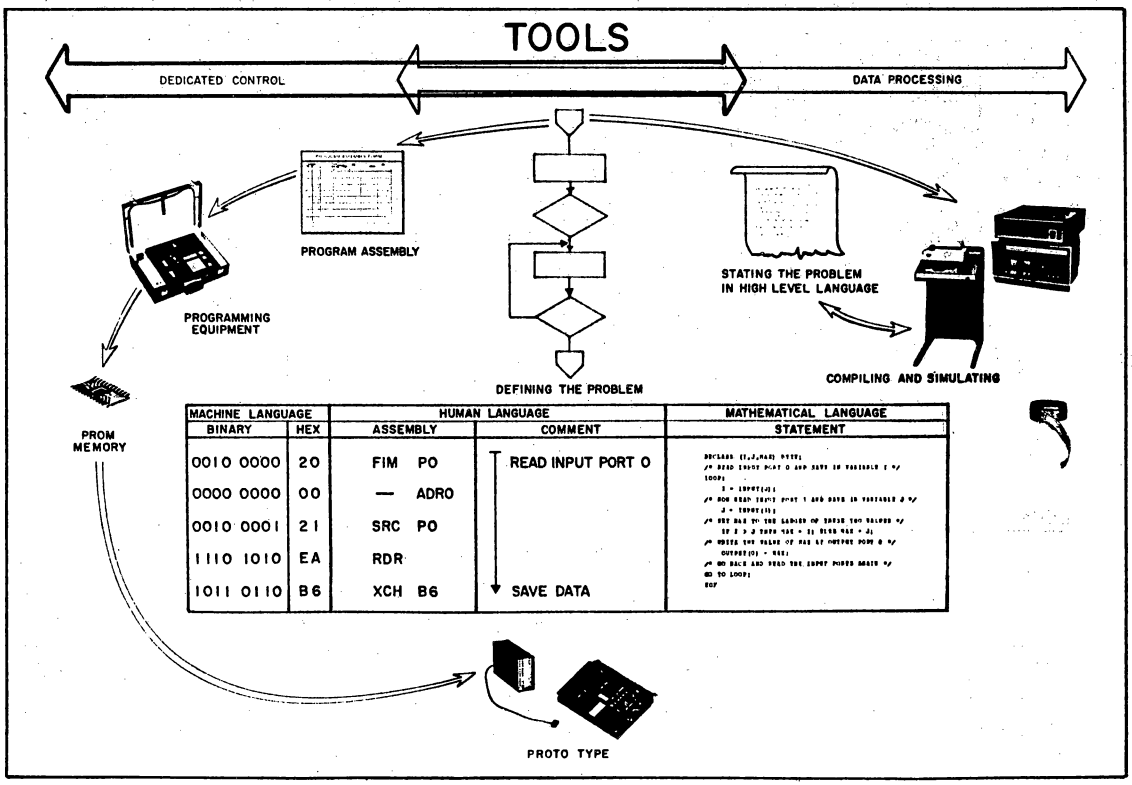
In data processing, multiple-over-lapping tasks requiring interrupt are assigned to the data processor to maintain its productivity. In random logic designs where microprocessors are dedicated to limited tasks the productivity requirement relaxes. Instead of the logic processor waiting for external interruption, it is put to work analyzing the external situation, continuously scanning and testing for inputs. The logic processor has complete command of the operation knowing when to accept or ignore inputs. This simple but effective technique eliminates the need for expensive and complex interrupt structures.

In data processing, volumes of data are often exchanged between the data processor and its storage devices. For data streams faster than the data processor can process, a technique called direct memory access (DMA) is used where the peripheral accesses memory directly. In random logic designs with little or no RAM memory the logic processor is not faced with a requirement for DMA.

The years of experience and the techniques of the data processing industry should not be blindly accepted as being best for microprocessor random logic designs. The requirements for this field need to be evaluated independently in light of the nature of the task.

TOOLS

The data processing industry has developed a wide range of tools some of which can be applied to microprocessors. Logic schematics are replaced using flow charts, program listings, and register and memory mapping techniques. The higher level computer-aided design techniques of assemblers, compilers and simulators are also available. These more sophisticated tools have not yet been developed to service the requirements of the logic processing market. The integrated nature of the logic processor with the hardware and its total control over the minute system details is contrary to the use of generalized, packaged problem solutions offered by the data processing industry.



MACHINE LANGUAGE		HUMAN LANGUAGE		MATHEMATICAL LANGUAGE
BINARY	HEX	ASSEMBLY	COMMENT	STATEMENT
0010 0000	20	FIM P0	READ INPUT PORT 0	
0000 0000	00	— ADR0		
0010 0001	21	SRC P0		
1110 1010	EA	RDR		
1011 0110	B6	XCH B6	SAVE DATA	

Cost effective use of logic processors requires that the designer be intimately familiar with his hardware. The use of higher level languages tends to insulate the designer from contact with the reality of the hardware problem. Instead, it forces him to rigorously define his problem in a precise mathematical format, molding the problem to fit a predefined solution.

Effective use of microprocessors for logic processing requires program design discipline not usually exercised in the data processing environment. Microprocessors may change random logic design techniques but the need for well documented designs still exists. The program designer must use the same design documentation disciplines exercised in hard-wired logic designs. In addition, the designer must learn to partition his programs with a view of flexibility in making anticipated design changes.

The computer-aided design tools such as assemblers and compilers are useful but tend to be lax in supplying the necessary design documentation. The ease of generating code with these computer-aided tools tends to encourage poor design documentation discipline, resulting in runaway designs.

Designers should avoid assemblers resident within his microprocessing system. These resident assemblers are not powerful enough to provide sufficient output to properly document the design. The designer should instead select an assembler on a larger host computer with sufficient output documentation capability.

DESIGNER

Programmers from the data processing industry best understand what microprocessors can not do. They recognize the microprocessor shorcomings when used for data processing. The data processing industry has also not yet accepted microprocessors. There is a demand and need for improvement in microprocessors before they can impact the data processing industry.

Design Engineers, on the other hand, are in the best position to benefit from microprocessors. The reality of microprocessors is an answer to a dilema. Technology has moved from discrete circuits to small scale integration (SSI), to medium scale integration (MSI) and to large scale integration (LSI). The transition through MSI was easy but technology presented a roadblock with LSI. LSI was too inflexible for developing designs in the laboratory. Microprocessors have removed the roadblock. LSI microprocessors with programmable ROM give the ball back to the design engineers.

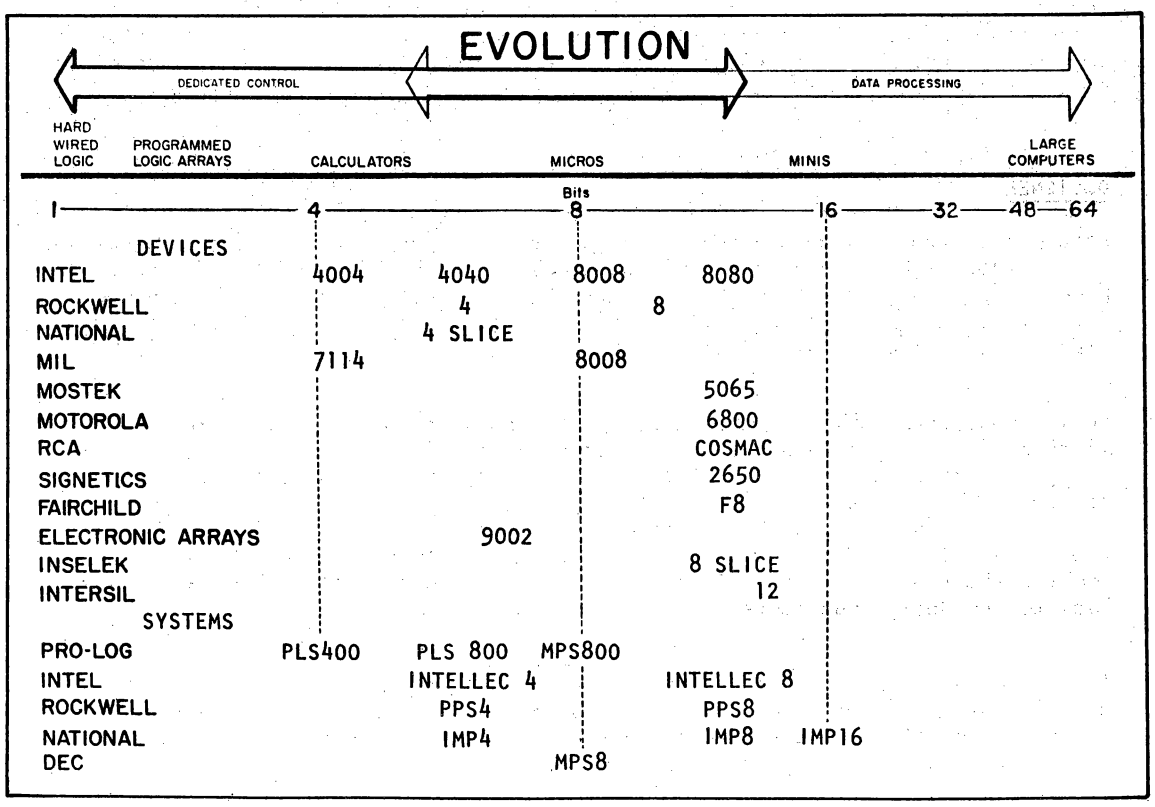
The design engineer understands the hardware problem and is the only one who can make decisions on possible cost effective hardware and program tradeoffs. It is an easy transition for design engineers to extend their logic design expertise into program design.

WHERE ITS AT

The best logic processing element on the market thus far has proven to be the Intel 4004. This device was not a purposeful entry into the logic processing market. The nature of the calculator it was originally designed for was a perfect example of a very cost effective dedicated control application. The 4004 thus evolved somewhat accidentally as fitting the requirement for a generalized logic processing element. The 4004 continues to survive as a logic processing standard, even as more powerful processors come on the scene. This will probably remain so until a specific microprocessor is designed exclusively for the logic processing market.

WHERE ITS GOING

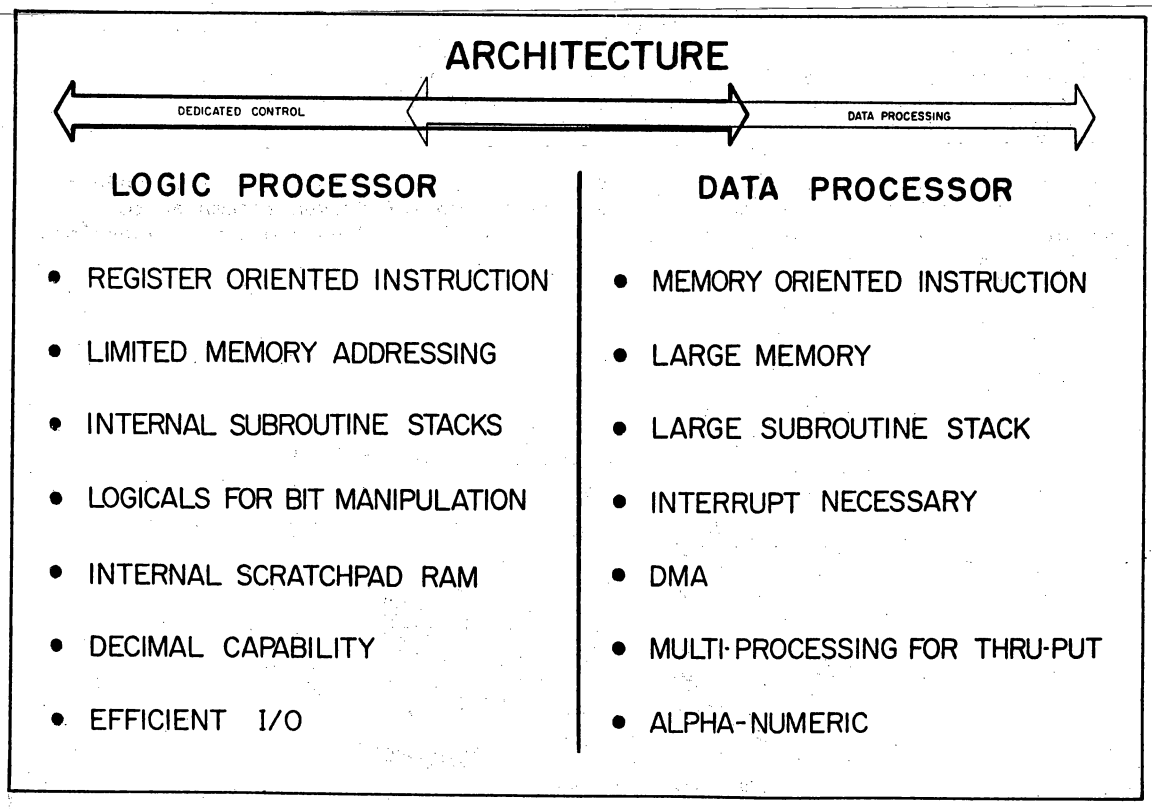
The list of microprocessor devices being designed today is impressive. Analysis of the list shows there is little recognition of the need for an improved logic processor.



New microprocessor developments are being influenced by the computer industry, which is evidenced by the evolution that is occuring. Newer microprocessors are coming forth with emphasis on serving the data processing market. There are various reasons for the evolution going on in microprocessors. Microprocessors are thought of as computers and without any challenge to this thought, it is only natural they should evolve to be better computers. The computer industry for its part is demanding "improved" microprocessors (Interrupt, DMA, more memory, speed, etc.) to satisfy the thru-put requirement.

This evolution is also influenced by the existence of the large RAM memory device market in the realm of the data processing industry.

In contrast, the voice of the dedicated control market is non-existent in demanding a better microprocessor as a logic processing element. This industry lacks the awareness and knowledge of what style of logic processing element is needed and thus provides no impetus to the microprocessor evolution.



WHAT TO LOOK FOR

In selecting a microprocessor for logic processing the designer should look for a ROM oriented architecture, logic capability, minimum external and interfacing requirements, and a human factor of understandability.

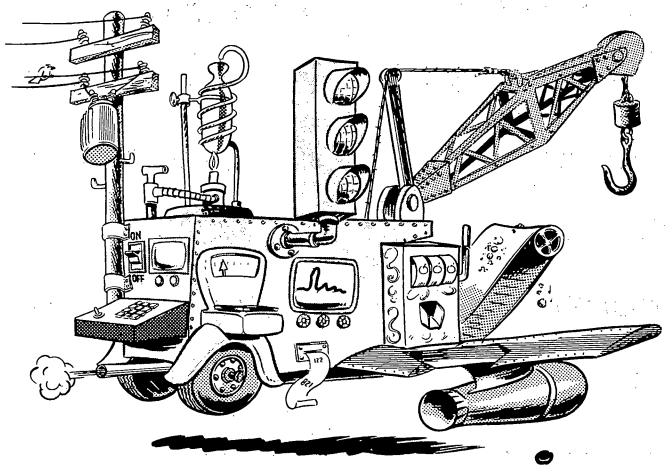
A ROM oriented architecture is one where the microprocessor instruction set can be executed completely from ROM without the need for external RAM. This requires an internal stack for subroutines and internal registers for indirect addressing operations.

The instruction set needs logical instructions for bit manipulation in addition to good byte manipulation. For the extensive interface to the real world decimal arithmetic capability is also desirable.

Memory size need not be large and in fact a small memory size implies efficiency in addressing, indicated by short memory and decision branching instructions.

Minimal external and interfacing requirements will be evidenced by small package size and few interconnects. Package size is kept small by a reasonable word size (8 bit maximum) and addressing capability (4K maximum). Interfacing requirements are minimal due to simple control using techniques such as combined I/O and memory addressing (memory mapped I/O) and combined instruction and data paths. For cost effectiveness the addressing and data paths need direct interface to off-the-shelf ROM and RAM memory devices.

From the human factors standpoint, a good logic processor should be easily understood. The instruction set should not require a computer driven assembler to generate a program. The architecture should be logical and have minimal exceptions and pitfalls to the user.



Go Real-World With
PRO-LOG Logic Processors

HOW TO DESIGN WITH MICROPROCESSORS

Presented at WESCON 75

by Edwin Lee

SUMMARY

This paper presents a procedure for using microprocessors in dedicated control applications. The procedure fulfills needs from design to field service. It was developed by Matt Biewer, a design engineer, over three years ago and is readily learned by engineers and technicians. It deliberately avoids data processing aids, such as assemblers, high level languages, simulated systems, and control panels. These computer aided design tools generally get in the way of cost-effective design and are more a result of the cultural influence of data processing, rather than practical need. In nearly four years of designing and maintaining systems using microprocessors, neither the author nor those with whom he works have ever required these data processing tools.

The paper first describes the Scientific Approach to problem solving and the engineering culture's adaptation of that approach. The microprocessor system is shown as a logic element and the programming and debugging of microprocessor systems is shown as a step for step parallel to the engineering design approach. The approach involves visual techniques and conventions for documentation, as well as the use of clip-on test equipment. This documentation and test equipment also provide the ability to debug hardware and programs on equipment as it operates in the field.

THE SCIENTIFIC APPROACH TO PROBLEM SOLVING

The Scientific Approach is the basis of organized problem solving. It has four distinct phases (see Fig. 1): Define the problem, partition the problem into humanly comprehensible chunks, synthesize the solution, and verify the solution in the real world. Defining the problem is primarily a process of selecting the finite number of variables which are significant to the problem and of categorizing to reasonable limits the goal sought.

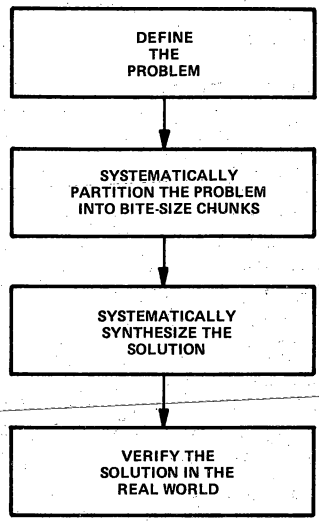


Fig. 1. The Scientific Approach to problem solving.

Systematic partitioning creates bite-sized chunks of the problem, which can be solved individually with human reason. It also provides a road map by which these bite-sized chunks can be fitted together, from the bottom up, to produce the final result.

Synthesis includes several steps; synthesizing solutions to the bite-sized chunks, integrating groups of chunks into clusters and finally integrating clusters of chunks into the final solution. Each step is kept humanly comprehensible.

THE ENGINEERING ADAPTATION OF THE SCIENTIFIC APPROACH

Fig. 2 shows the engineer's procedure for designing hardwired logic systems. The product specification defines the problem. Block diagraming breaks the design problem into bite-sized chunks. The schematic--breadboard--test cycle designs and debugs in the synthesis phase. The design and debug process is first done at the

module level, then the subsystem level, and then the system level. The field trials are used to verify the solution in the real world.

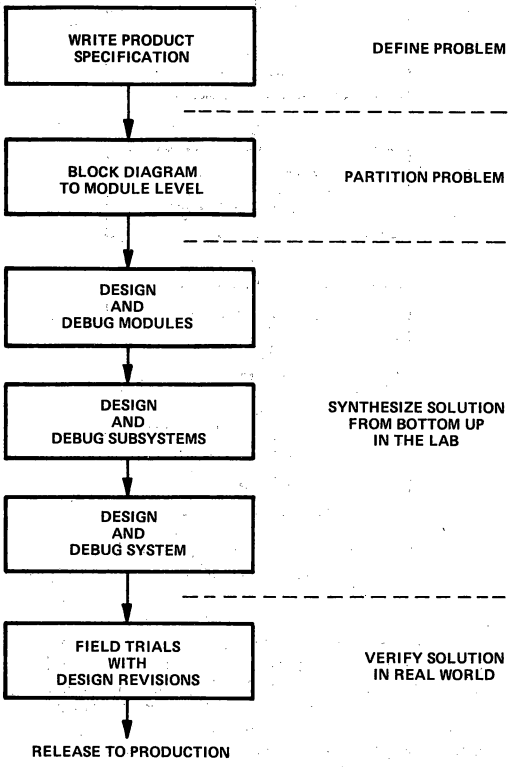


Fig. 2. The engineer's approach to system design and its relationship to the Scientific Approach.

A detailed look at the synthesis technique is shown in Fig. 3. It shows the sequence used by the design engineer to design and debug at each level of synthesis. The conceptual design is drawn in schematic form. The schematic is the engineer's language for visualizing solutions. Each of the schematic symbols represents the functioning of a particular type of hardware. The key to a good schematic is not just the symbols used, but their visual grouping and proper labeling to clearly show their interaction. Schematics are not drawn to efficiently fill a sheet of paper, but to serve as visual tools. White space, labels, right-left conventions for inputs and outputs, all help the designer and anyone else who uses the documents.

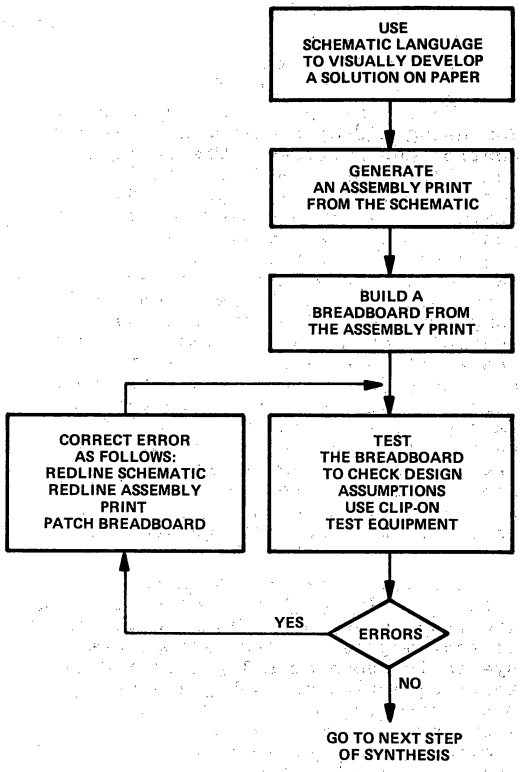


Fig. 3. The engineer's design and debug procedure.

The schematic of Fig. 4 uses standard symbols and these conventions and is, therefore, clear to most engineers and technicians.

An assembly print and/or wire list is generated from the schematic. The assembly print maps the actual layout and wiring of hardware. The breadboard is tested to verify the engineer's design assumptions. In the testing phase, engineers follow a customary approach: The engineer hangs his schematic and assembly print on the wall next to his workbench. On the workbench are the breadboard, the clip-on test equipment, such as scopes, voltmeters, waveform generators and power supplies and the interface exercising circuitry. The test equipment is not built into the

breadboard, because that would severely limit the usefulness of the equipment and would require a later redesign to take it out. The engineer hooks up the hardware and begins testing.

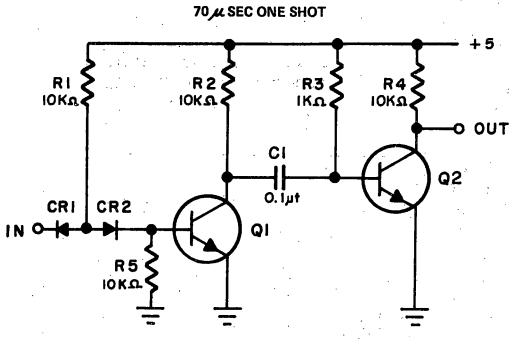


Fig. 4. A typical circuit schematic showing the visual conventions used by design engineers.

Normally, the discovery of an error is not long in coming. A correction, for the wise engineer, follows this sequence: The schematic is redlined in the white space near the part of the circuitry involved with the change. The assembly print is then redlined and the breadboard is patched to reflect the redline. The engineer changes his documentation first, to keep his documentation in lockstep with his hardware. He simply redlines and patches, because it is a waste of time and money to redraw and rebuild after each of the inevitable corrections. Patching is easy when a schematic is drawn with white space and the hardware is layed out with room for patches.

The debugging cycle just described is repeated many times during a normal design. The whole process is carried out at the workbench...requires no computer assistance and can be started and stopped at any point without loss of time. The engineer may turn off power, leave the workbench (for phone calls, meetings, or to go home at 3 a.m.) and then return, turn on power and pick up right where he left off.

The design-debug cycle is systematically employed first at module level, then at subsystem level and finally at system level. At each level the engineer checks only a humanly comprehensible number of items. At the module level he checks the interaction of components to perform the module function, at the subsystem level the interaction of modules, and at the system level the interaction of subsystems. This synthesis approach also allows different people or different groups to work in parallel... so long as all groups follow the same documentation conventions.

Field trials are an essential part of the design cycle. They test not only the design in the real world, but the product specification. Generally, when the engineer finally does design something to specification, marketing changes it. Equipment can and must be modifiable in the field, if the field trials are to proceed effectively. The clip-on test equipment and a soldering iron allow test and hardware changes in the field.

THE MICROPROCESSOR SYSTEM AS A LOGIC ELEMENT

The key to using the microprocessor system effectively is to treat it as a logic element, rather than as a computer. This approach is especially necessary for dedicated control systems, and these systems constitute at least 98% of the market for microprocessors. The distinctions between the real world of dedicated control and the artificial world of data processing are spelled out in Reference 1. A dedicated control system has only one essential characteristic...when its power is turned on the system does its job. A well designed dedicated control system has a second characteristic... all system hardware is essential to doing the job. Things such as computer control panels, loaders, diagnostic routines, and peripheral controllers do nothing to solve problems, but merely service the problem solver.

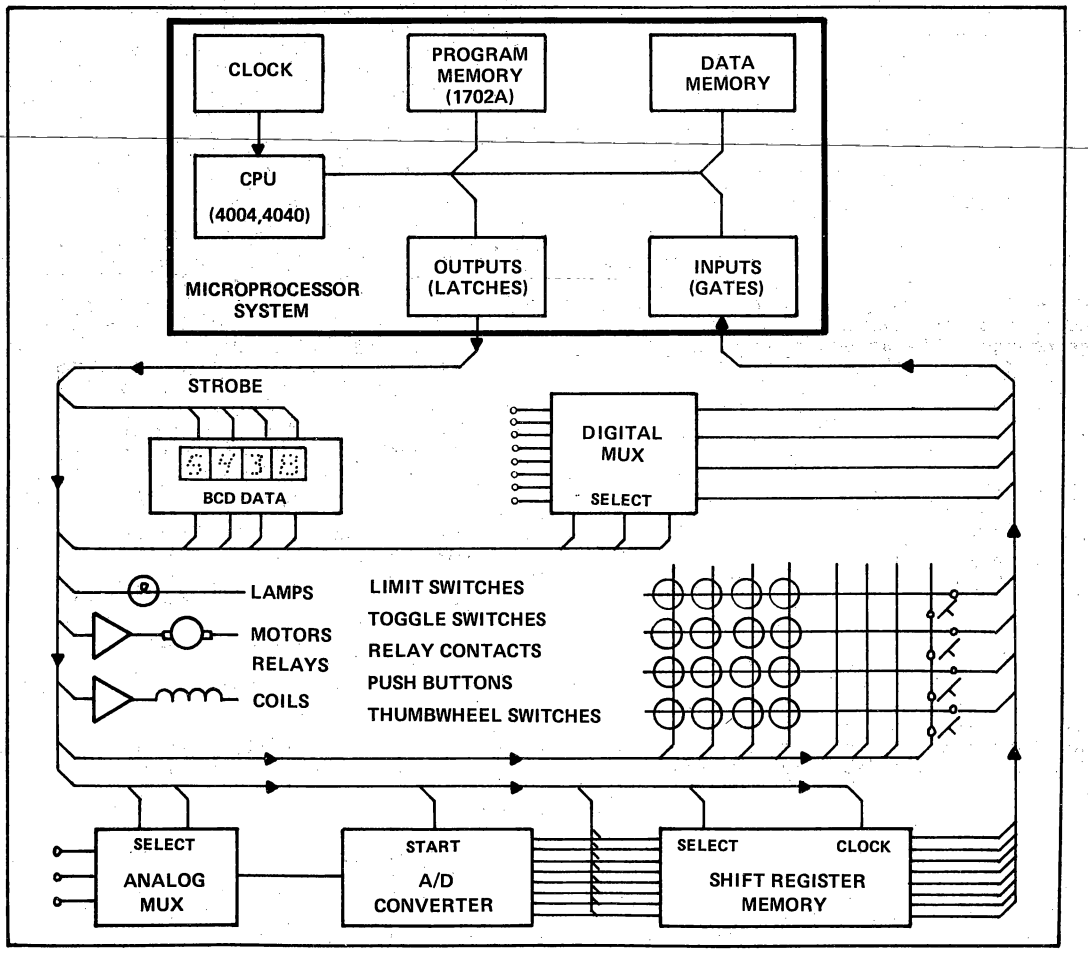
In Fig. 5, a microprocessor system is shown as a logic element wired to loads. The microprocessor system is a black box. It has a number of input gates and a number of latching output drivers (the outputs of flip-flops). These inputs and outputs are simply wired to level compatible loads. If a load happens to be a group of switch contacts (keys, relay contacts, thumbwheel switches), these contacts are put in a matrix with one axis driven by the outputs and the other axis sensed

by the inputs. Functions such as noise rejection, switch-bounce elimination, and assigning meaning to the various contacts, are all done by the program.

The microprocessor system can do anything any other black box with N inputs and M outputs can do. It can do timing, make decisions, store data, do arithmetic and other analysis, convert codes, linearize curves, etc. The basic difference with this black box is that instead of having to wire up components inside, the designer simply puts specific coding into the Program Memory. PROM or ROM is used as Program Memory, because of the previously stated requirement that the system does its job whenever power is turned on. The PROMs or ROMs are coded before being plugged into the system.

The data memory in the microprocessor system is generally no more than a few hundred bits of register storage. If the designer wants to store more data, it is far more cost-effective to put data memory outside the microprocessor system as an I/O module.

The internal workings of the system are time sequential, so a clock is necessary to step the system along.



The system operates through the interaction of the CPU and the Program Memory. Conceptually, this interaction is indentical to the process undergone by a person reading a set of instructions and performing them one after another. When power is turned on, the CPU reads the first code word in Program Memory, interprets its meaning, and does what it is told. When it finishes doing the first operation, it reads the next line and does that. The CPU is preconditioned (just as we are in school) to follow this Read/Do cycle.

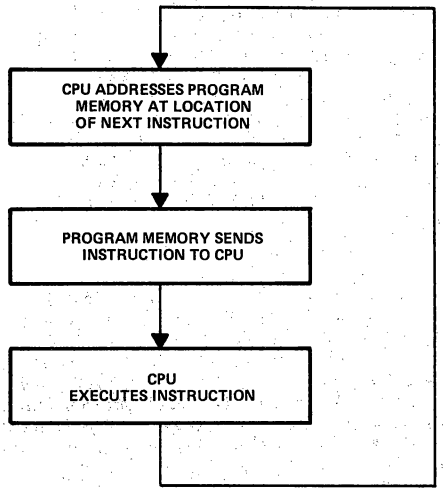


Fig. 6. The basic steps in an instruction cycle.

Fig. 6 shows the timing sequence undergone by the microprocessor system to do one particular task. The full sequence is called the instruction cycle. Using the 4004 as an example, the sequence takes eight clock times to complete... three for the CPU to send a twelve bit address to Program Memory in four-bit bites, two for the Program Memory to send back the eight-bit instruction in four-bit bites, and three to decode and execute the instruction. The instruction cycle takes around eleven microseconds and might do something such as send four bits of data to output latches, sense the data at four gates, or add two four-bit words. Some instructions require more than one instruction cycle to define and execute. For example, there might be a lookup table in Program Memory (to linearize a curve, multiply two decimal digits, etc.). To take one eight-bit word and translate it to another eight-bit word using the lookup table takes two instruction cycles with the 4004.

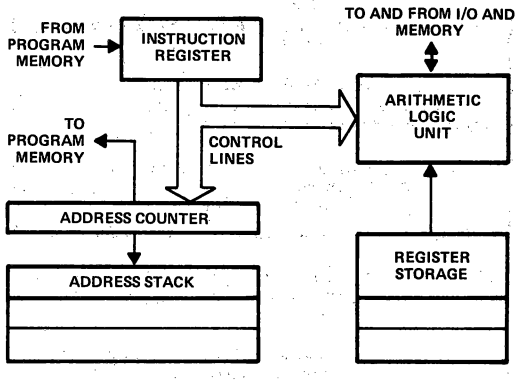


Fig. 7. The working registers and data flow in the 4004 CPU.

The functional circuitry inside the 4004 CPU chip is shown in Fig. 7. The address counter controls which line of program memory is to be read next. It starts at 000 and counts up. The Instruction Register and related decoding gates interpret the eight-bit word and cause things to happen. The Arithmetic Logic Unit and Register storage are Read/Write Memory, in which data can be manipulated by an Instruction. The address stack enables the CPU to return to the main program after it's used a program module called a subroutine. It's similar to the memory required by a person reading instructions, where he's been told on page 1 to read the details on page 10, and after reading the details must remember to return to page 1. The judicious use of subroutines is essential to the bottom-up synthesis approach described below.

THE ENGINEERING APPROACH

The engineer's design approach fits the microprocessor based system of Fig. 5. The key to understanding the approach and the techniques which make it work is duality...for each step in the hardware design process there is a corresponding element in the programming process (see Fig. 8).

HARDWARE	PROGRAM
BLOCK DIAGRAM	FLOW CHART
SCHEMATIC LANGUAGE	MNEMONIC CODING
WIRE LIST OR ASSEMBLY PRINT CODING	HEXADECIMAL CODING
SCHEMATIC AND ASSEMBLY PRINT	PROGRAM ASSEMBLY FORM
COMPONENT	INSTRUCTION
MODULE OR SUB SYSTEM	SUB ROUTINE
BREADBOARD OR PROTOTYPE	PROM
WIREWRAP GUN OR SOLDERING IRON	PROM PROGRAMMER
OSCILLOSCOPE	CLIP-ON TESTER

Fig. 8. The dual or equivalent elements in hardware and programs.

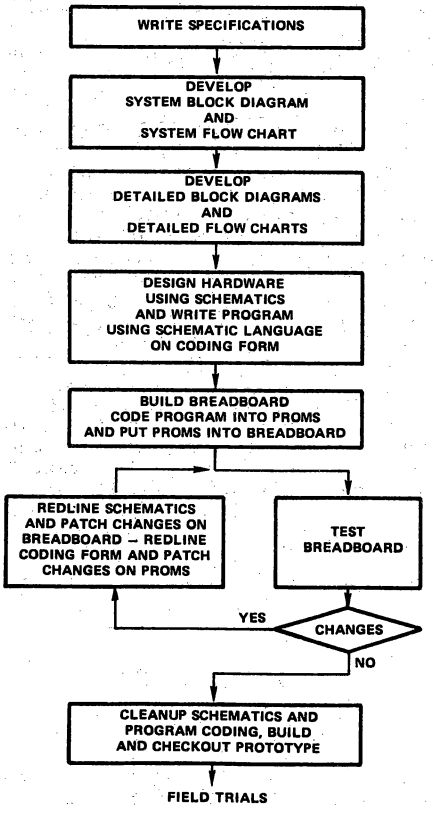


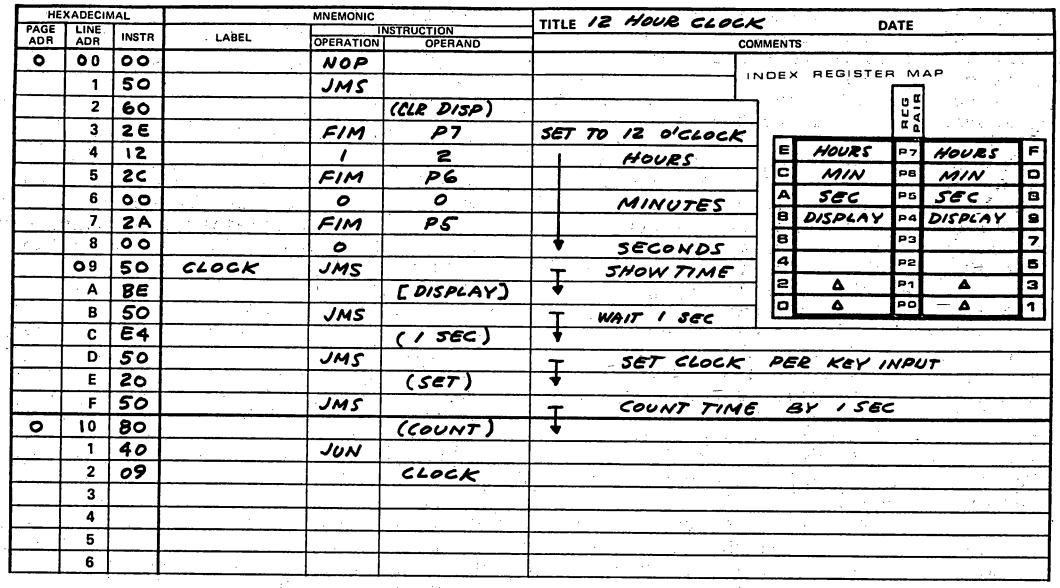
Fig. 9. The step-by-step design procedure for microprocessor based systems.

In Fig. 9, the design process is shown with the additional steps for microprocessors. Writing the product specification is not significantly altered because of microprocessors, although, once the capability of microprocessors is recognized, the specifications will change to eliminate buttons and pots, simplify the operator functions and allow for many more options.

Just as the block diagram is used to partition the hardware into bite-sized chunks, the flow chart is used to partition the program into bite-sized sequences. Flow charting is simply a tool for describing a sequence of events. It can be adequately represented with two symbols...a box for a process and a diamond for a decision. The number of levels of flow charts is strictly a function of the complexity of the problem. Fig. 9 itself is a flow chart.

Once flow charting has generated bite-sized modules, these modules can be synthesized visually with mnemonic coding. Mnemonic coding is the dual of the engineer's schematic language for hardware. I prefer to call mnemonics a schematic program language. "Mnemonics" are simply shorthand words, which sound like their functions.

The successful use of mnemonic symbols for synthesis is through their visual organization on paper with white space, conventions and proper labeling. Figs. 10 and 11 show the Program Assembly Form developed by Matt Biewer and described in Reference 2. This form serves as both the schematic and assembly print for the Program. The Fig. 10 shows the main program and Fig. 11 shows a time delay subroutine for a real time clock program. The (1 SEC) subroutine is equivalent to the one shot circuit of Fig. 4. The portion of the assembly form which serves the function of the schematic is under the MNEMONIC columns. The labels in the LABEL column have the same function as circuit titles (such as the title to the one shot circuit in Fig. 4). A specific label appears only once in a system, at the beginning of the group of instructions which form the module. The Operation and Operand Columns contain the schematic. The mnemonics shown here are the Intel mnemonics for the 4004, 4040 CPU chips.



HEXADECIMAL PAGE ADR	HEXADECIMAL LINE ADR	HEXADECIMAL INSTR	MNEMONIC LABEL	MNEMONIC INSTRUCTION OPERATION	MNEMONIC INSTRUCTION OPERAND	COMMENTS
0	00	00		NOP		
	1	50		JMS		
	2	60			(CLR DISP)	
	3	2E		FIM	P7	SET TO 12 O'CLOCK
	4	12		1	2	HOURS
	5	2C		FIM	P6	
	6	00		0	0	MINUTES
	7	2A		FIM	P5	
	8	00		0		SECONDS
	09	50	CLOCK	JMS		SHOW TIME
	A	BE			[DISPLAY]	
	B	50		JMS		WAIT 1 SEC
	C	E4			(1 SEC)	
	D	50		JMS		SET CLOCK PER KEY INPUT
	E	20			(SET)	
	F	50		JMS		COUNT TIME BY 1 SEC
0	10	80			(COUNT)	
	1	40		JUN		
	2	09			CLOCK	
	3					
	4					
	5					
	6					

TITLE 12 HOUR CLOCK DATE

INDEX REGISTER MAP

		REG PAIR		
E	HOURS	P7	HOURS	F
C	MIN	P8	MIN	D
A	SEC	P5	SEC	B
8	DISPLAY	P4	DISPLAY	9
6		P3		7
4		P2		5
2	Δ	P1	Δ	3
0	Δ	P0	Δ	1

Fig. 10. The complete documentation of the main program for a 12 Hour Clock System.

HEXADECIMAL PAGE ADR	HEXADECIMAL LINE ADR	HEXADECIMAL INSTR	MNEMONIC LABEL	MNEMONIC INSTRUCTION OPERATION	MNEMONIC INSTRUCTION OPERAND	COMMENTS
0	E0					
	1					
	2					
	3					
	E4	22	(1 SEC)	FIM	P1	1 SECOND DELAY - INITIALIZE
	5	F5		F	5	MSD
	6	20		FIM	P0	
	7	00		0	0	LSD
	E8	70	Δ	ISZ	R0	DELAY TIMING OPERATION
	9	E8			Δ	
	A	71		ISZ	R1	
	B	E8			Δ	
	C	72		ISZ	R2	
	D	E8			Δ	
	E	73		ISZ	R3	
	F	E8			Δ	
0	F0	C0		BBL	0	
	1					
	2					
	3					
	4					
	5					
	6					

TITLE 12 HOUR CLOCK DATE

Fig. 11. The documentation of a 1 Second Delay subroutine.

In normal operation, the microprocessor sequences through the instructions from the top down. Some instructions act like components, others act like wires. Instructions that act like components cause the microprocessor to do something...load a register, add two numbers, write data out (FIM, ADD, WRR). Instructions that act like wires cause the processor to go from place to place (module to module) in the program (JUN, JMS, BBL). Some instructions

compound these two functions (JCN, ISZ). Once the designer understands the functions related to the symbols, he groups the symbols on paper to achieve the functions of a module. A module is a subroutine. A subroutine is like a 5 volt power supply. If an engineer designs a system using 30 TTL chips, these chips all use +5 volts, but he builds only one 5 volt supply and provides a power wire and a ground return to each chip from that supply. The JMS instruction performs the function of the power wire, BBL is the ground return (in the 4004 mnemonics, other mnemonics use RET instead of BBL). The functions of JMS and BBL are illustrated in Figs. 10 and 11. JMS (1 SEC) indicates the program should go to the (1 SEC) subroutine. At the end of that subroutine, BBL sends the processor back to the main program at JMS (SET). The subroutine module should be visually obvious, therefore, the label is at the top and there is white space (unused lines) above and below the module. Equally important is the documentation shown in the Comments Column of the schematic. Failure to document Comments is known as job security in data processing...it's known as sloppy documentation in engineering. The Comments explain to the design engineer, to manufacturing test, and to field service what the program is doing to affect the hardware...this link is vital to debugging.

After the program schematic is generated, the INSTR Column is coded in hexadecimal. The ADR Columns represent the program memory locations. Coding the Instruction Column is equivalent to generating a hardware wire list, since this coding is put in the PROM to configure the hardware. We do coding with the aid of a Look-Up Table, such as the partial table shown in Fig. 12. After a while, most coding is done from memory at an average rate over ten lines per minute. Address and instruction coding should be in hexadecimal (see Fig. 13), not in octal or binary. The reason is simple...hardware debugging requires a humanly comprehensible code in which what happens in a given interval of time is readily related to what is on a piece of paper. Binary is a nightmare. Octal does not provide a specific character for specific time slots for four-bit or eight-bit microprocessors. Hexadecimal is the best compromise...one character represents four bits. Most instructions are eight bits or two hexadecimal characters.

HEX CODING		MNEMONIC OPR	MNEMONIC OPA	DESCRIPTION OF OPERATION
0	0	NOP		No operation.
1	0	SKP		Skip the next instruction word.
1	C_X	JCN	C_X	Jump on condition C_X to the program memory address A_1.
A_2	A_1		LABEL	A_2, otherwise continue in sequence. (see back cover).
2	P_X0	FIM	P_X	Fetch immediate from program memory data D_1. D_2 to
D_2	D_1	D_2	D_1	index register pair P_X
2	P_X1	SRC	P_X	Send register control. Send the contents of index register pair P_X to I/O ports and RAM register as chip select and RAM character address.
3	P_X0	FIN	P_X	Fetch indirect. Send contents of register pair 0 out as a program memory address. Data fetched is placed into register pair P_X
3	P_X1	JIN	P_X	Jump indirect. Jump to the program memory address designated by contents of register pair P_X
4	A_3	JUN		Jump unconditional to program memory address A_1. A_2. A_3.
A_2	A_1		LABEL	
5	A_3	JMS		Jump to subroutine located at program memory address A_1.
A_2	A_1		LABEL	A_2. A_3. Save previous address (push down in stack).
6	R_X	INC	R_X	Increment contents of register R_X.
7	R_X	ISZ	R_X	Increment and step on zero. Increment contents of register
A_2	A_1		LABEL	R_X. if result is not 0 go to program memory address A_1. A_2. otherwise step to the next instruction in sequence.
8	R_X	ADD	R_X	Add contents of register R_X to accumulator.
9	R_X	SUB	R_X	Subtract contents of register R_X to accumulator with borrow.
A	R_X	LD	R_X	Load contents of register R_X to accumulator.
B	R_X	XCH	R_X	Exchange contents of index register R_X and accumulator.
C	D_X	BBL	D_X	Branch back one level in stack to the program memory address stored by a prior JMS instruction. Load data D_X to accumulator.
D	D_X	LDM	D_X	Load data D_X to accumulator.
E	X	I/O and RAM register instructions		
F	X	Accumulator instructions		

Fig. 12. The translation table for generating PROM coding from mnemonics.

HEX TO BINARY CONVERSION				
HEX	8	4	2	1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
A	1	0	1	0
B	1	0	1	1
C	1	1	0	0
D	1	1	0	1
E	1	1	1	0
F	1	1	1	1

Fig. 13. A table showing the relationship between hexadecimal and binary coding.

Because of the manner in which the schematic and wire list are represented together on the same document, the function of the white space around the module serves both schematic and hardware in leaving room for patches.

Once the program schematic and wire list have been properly documented, it's time for the debugging phase. The equivalent of a breadboard is a programmed PROM. The only really useful PROM for this purpose is the 1702A MOS PROM. It is readily available and erasable. The 1702A PROM has 256 (hexadecimal FF) eight-bit words. The tool we use to code the PROM is the PROM Programmer, shown in Fig. 14. This instrument is equivalent to a soldering iron. The programmer has four basic operations...List, Program, Duplicate (with corrections), and Verify. When a PROM is coded for the first time, the Program mode is used. In this mode, data is entered through the keyboard into the PROM (addressing is automatically kept track of by the instrument). Only two keyboard entries per line of code are required. An entire 1702A can be coded from the assembly form in under 15 minutes by hand. If the systematic module-up method of synthesis is being used, the operator will never be coding more than 60 or 70 instructions at a time. For example, he would code and debug the (1 SEC) module, and this takes less than a minute to key in. Once the basic code is in the PROM, the Duplicate mode is used to make corrections or new PROMs. In the Duplicate mode, the original

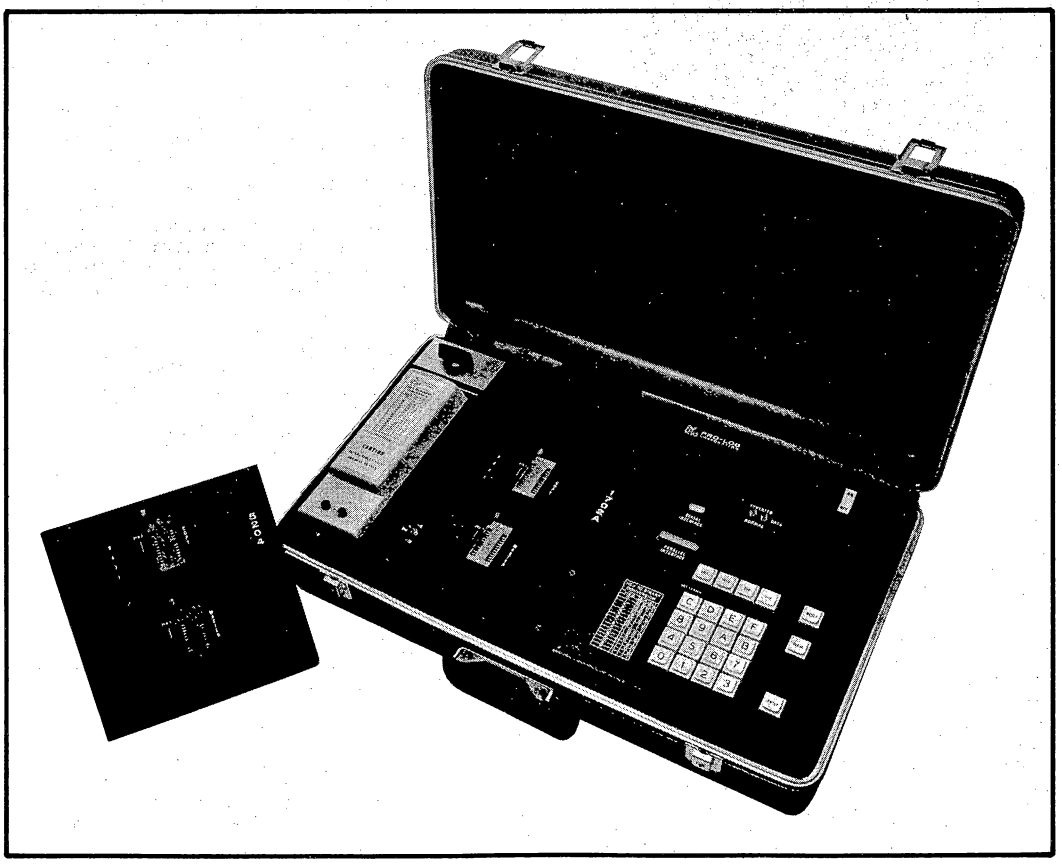


Fig. 14. The Pro-Log Series 90 PROM Programmer used in the design cycle.

PROM ("was" condition) is put in the Master Socket, a blank PROM is put in the Copy Socket, and the red lines are keyed in through the keyboard. A new copy ("is" condition) takes 30 seconds to generate.

Once the PROM is coded, it is placed in the microprocessor system at the workbench and breadboard testing begins. The Program Assembly Form is taped on the wall, one page of forms to a PROM. At the workbench are the oscilloscope, power supplies, voltmeter and a new piece of test equipment, the System Analyzer pictured in Fig. 15.

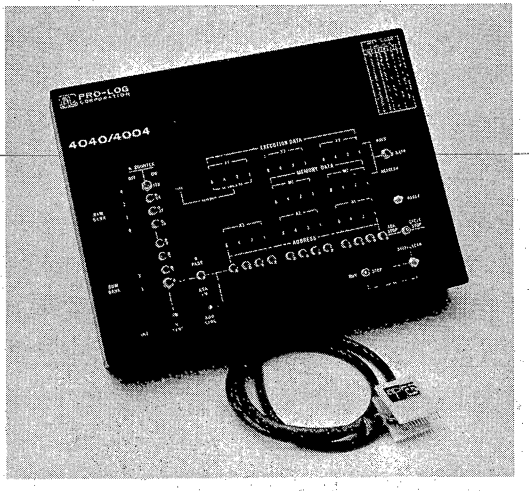


Fig. 15. The Pro-Log M422A System Analyzer used in design and field trials.

This unit clips onto the CPU chip in the microprocessor breadboard. It is self-powered, with built-in logic that enables it to observe without affecting the operation of the system (some control functions are available). Because it clips directly onto the microprocessor CPU, it observes everything that flows into or out of the chip; program addresses, instructions coming back from PROM, and data as it is being manipulated by the CPU. The primary characteristic of the System Analyzer is its ability to synchronize on any step of the program. To observe the time delay subroutine in action, set up the address switches to the appropriate program address and observe the display lights which shows everything going on whenever that instruction is executed by the CPU. Equally important, the Analyzer generates a Scope Sync at the same time. If something goes wrong, an oscilloscope can be used to see if it is a hardware or timing problem, rather than a program problem.

The System Analyzer and an oscilloscope are all that we've found necessary to debug hardware. If a program error is found during test, red line the documentation (mnemonics and comments, then the Hex coding) then take the original PROM, put it in the Programmer and create a duplicate PROM with all the corrections as described above. (Note this luxury, one that the hardware designer doesn't have: The old breadboard (PROM) is untouched until erased with an ultra-violet light and is available in case the patches don't work.) Put the new PROM back into the breadboard and continue debugging until all the corrections are made. Remove the breadboard system...ready to use.

Repeat the process at subsystem and system level. For the 12 Hour Clock System, the instructions in Fig. 10 were all that were written and debugged at system level. What need is there for a computer to help? At system level tape the whole program on the wall. The modules are visually obvious and sequenced in PROMs according to convention...main program at the beginning of the first PROM and subroutines placed after the main program. Because of white space, the modules are highly visible and do not move around on the paper when corrections are made. Corrections are placed in the white space near the affected module. Because of stability of location, you become familiar with the location of modules in the program. You develop a system understanding which is far greater than that allowed by reams of computer printout, which shifts around each time the program is reassembled. Once the system program is debugged, all modules may be moved once to eliminate unused PROM. Often, even this isn't necessary.

Once the design is complete and systems are shipped, they can be tested in the field simply by clipping on the System Analyzer and an oscilloscope. The field service man needs only the Program Assembly Form (properly documented) and hardware schematics.

PRACTICAL EXPERIENCE

The techniques described here have been used on many significant system designs. One of the most complex of these was a heart monitoring system, designed and built three years ago. This system monitored and recorded several heart waveforms, did waveform analysis, real time signal averaging, had automatic gain control and was entirely operated by the microprocessor. The program was about 3,200 instructions long, was written and debugged (including most of the hardware debugging) by two engineers working in parallel on different parts of the design, on a part time basis, in under 600 manhours. It's my observation that the bulk of real-world control problems require less than 2,000 instructions to implement. For this size program computer aided design does little to improve the design approach and does a lot to separate the design engineer from intimate knowledge of his hardware.

CONCLUSIONS

The design--debug procedure described here seems superior to the RAM memory, simulated system approach for many reasons. The documentation is forced to be in lockstep with the hardware and is made visually useful with white space register maps and comments. Therefore, the design is under control. Power can be turned off at anytime in the debugging cycle and turned back on to pick up instantly where the designer left off. No time is wasted loading and dumping programs. The old breadboard can be maintained until the new breadboard is proven. Many times a fix doesn't work and the designer wants to get back to the way it was. With this method, he simply plugs in his last PROM. Old PROMs are erased only when the designer is satisfied with the corrections. Two or more engineers can work in parallel, at two or more workbenches, on two or more modules and simply tie the modules together at subsystem or system level. Engineers can carry test equipment into the field to modify operating systems.

What engineers need to do, rather than to blindly embrace computer aided design, is to standardize mnemonics and the visual conventions for using them. Program design and debugging can then be as efficiently handled as their hardware equivalents. The IEEE should sponsor this effort, rather than pursue its present course of telling engineers they must use computers and high level languages to design microprocessor based systems.

BIBLIOGRAPHY

1. "The Designer's Guide to Programmed Logic", by Matt Biewer. Published by Pro-Log Corporation.

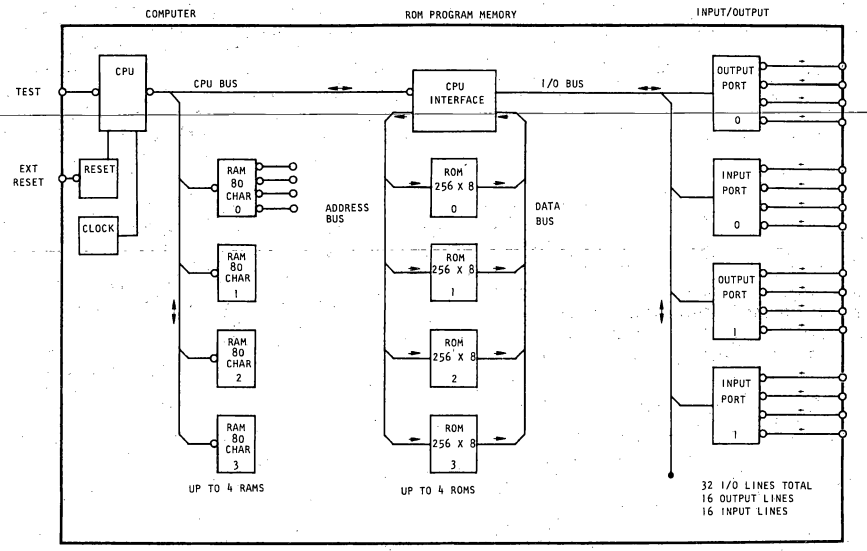
2. "Microprocessors for Dedicated Control", by Matt Biewer. Presented at WESCON '74. Available from Pro-Log Corporation.

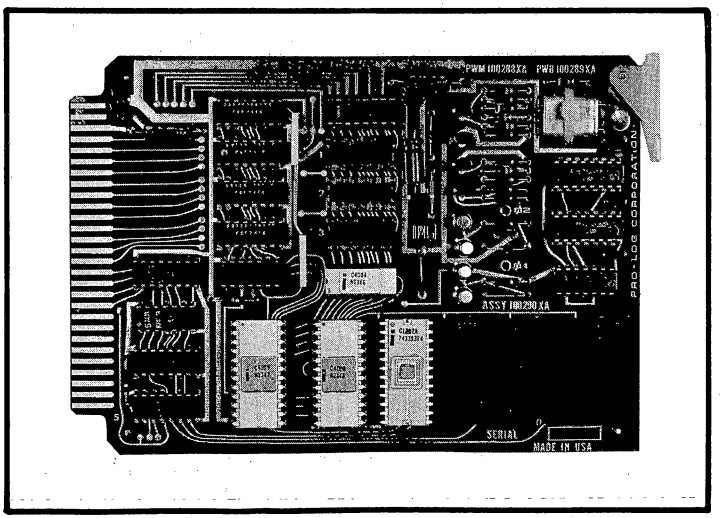
3. "What Can You Do With A Microprocessor?" by Robert H. Cushman, EDN, March 20, 1974, pg. 42.

A programmable logic system which implements the 4004 Microprocessor into a working system with CPU, ROM program memory, RAM register storage and I/O on a single card. The PLS-401 organization provides for reasonable program and I/O capability to give the lowest cost approach to investigating PLS technology.

FEATURES

- Single card programmed logic system for protypes or production
- 1024 words of ROM program memory capacity (4 ROMs)
- 320 characters of RAM register storage capacity (4 RAMs)
- Four output ports (16 lines)
- Four input ports (16 lines)
- One RAM output port (4 lines)





PLS-401 ONE-CARD SYSTEM

PLS-401 ONE-CARD SYSTEM SPECIFICATIONS

Card Dimensions

4.5 inches high
6.5 inches long
0.48 inch maximum profile thickness
0.062 inch printed circuit board thickness

Includes

Card ejector
One 4004 CPU
One 4002 RAM and four RAM sockets
One 1702A ROM and four ROM sockets
Master power-on and external reset circuit
Crystal clock circuit
Four TTL output ports (16 lines)
Four TTL input ports (16 lines)
One MOS output port (4 lines)
CPU test input (MOS)

Maximum Systems Capabilities

Four 4002 RAMs (320 four bit characters)
Four 1302, 1602, or 1702 ROMs (1024 words of program memory)
20 output lines

16 TTL port lines
4 MOS RAM port lines

16 TTL input lines

Instruction Execution Capability

Capable of executing all 46 of the 4004 CPU Instruction except for DCL and WPM
11.2 microseconds instruction execution time

Logic Levels Of External Connections

Low level active:

TTL Port:	TTL compatibility and loading
MOS Input:	TTL compatibility
MOS Output:	Drive capability, one LPTTL or one TTL load with 12K pull-down to -VDD

Power Requirements

+VCC = +5 volts 5% @ 550 mA maximum fully loaded (30 mA per RAM, 35 mA per ROM)
GND = 0 volts
-VDD = -10 volts 5%@ 350 mA maximum fully loaded (30 mA per RAM, 35 mA per ROM)

Connector Requirements

56 pin, 28 position dual-readout on 0.125 centers

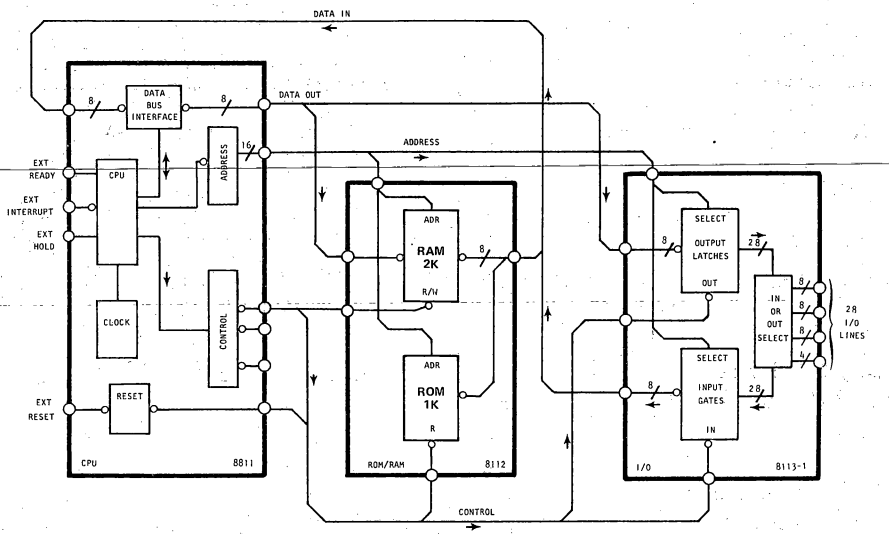
EDGE CONNECTOR PIN LIST

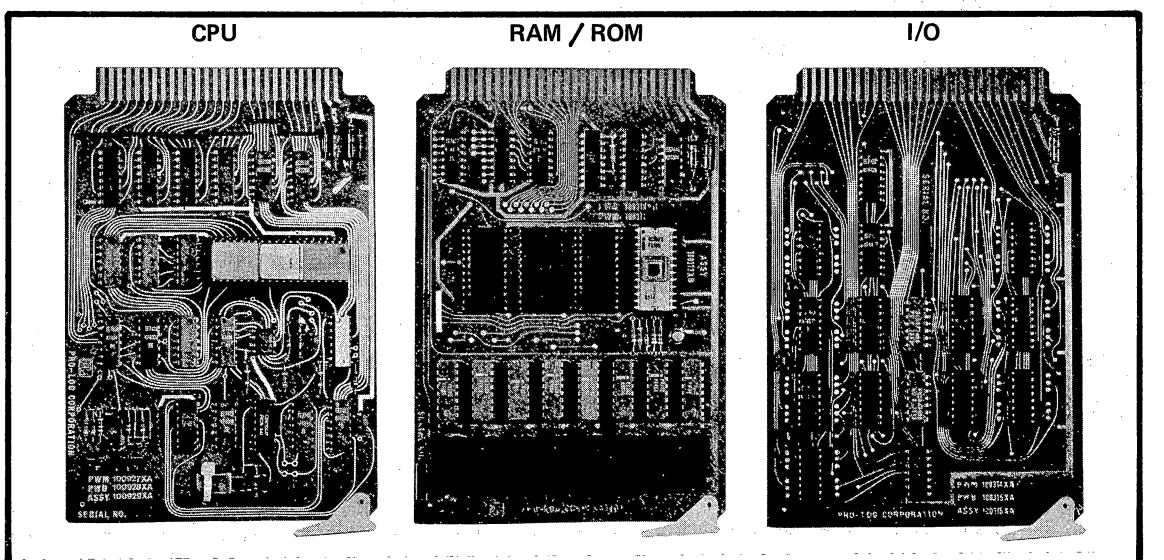
PIN NUMBER SIGNAL FLOW		PIN NUMBER			PIN NUMBER SIGNAL FLOW
SIGNAL					SIGNAL
+5 VOLTS	IN	2	1	IN	+5 VOLTS
GROUND	IN	4	3	IN	GROUND
-10 VOLTS	IN	6	5	IN	-10 VOLTS
RO-8*	OUT	8	7	OUT	OUT 2-8*
RO-4*	OUT	10	9	OUT	OUT 2-2*
RO-2*	OUT	12	11	OUT	OUT 2-1*
RO-1*	OUT	14	13	OUT	OUT 2-4*
CLOCK Ø1*	OUT	16	15	OUT	OUT 1-8*
TEST*	IN	18	17	OUT	OUT 1-2*
OUT 3-8*	OUT	20	19	OUT	OUT 1-1*
OUT 3-2*	OUT	22	21	OUT	OUT 1-4*
OUT 3-1*	OUT	24	23	OUT	OUT 0-8*
OUT 3-4*	OUT	26	25	OUT	OUT 0-2*
RST*	OUT	28	27	OUT	OUT 0-1*
EXT RESET*	IN	30	29	OUT	OUT 0-4*
IN 0-1*	IN	32	31	IN	IN 2-8*
IN 2-1*	IN	34	33	IN	IN 0-8*
IN 0-2*	IN	36	35	IN	IN 2-4*
IN 2-2*	IN	38	37	IN	IN 0-4*
IN 1-8*	IN	40	39		
IN 3-8*	IN	42	41		
IN 1-1*	IN	44	43		
IN 3-1*	IN	46	45		
IN 3-2*	IN	48	47		
IN 1-2*	IN	50	49		
IN 3-4*	IN	52	51		
IN 1-4*	IN	54	53		
CLOCK Ø2*	OUT	56	55		

A three card 8 bit microcomputer which implements the 8080 CPU into a working system complete with ROM/RAM and I/O. The system includes features to accomodate DMA and Interrupt. The 883 is designed for low cost control or mimimum data handling applications.

FEATURES

- 8080 CPU with Interrupt and DMA capability
- All memory usable as data or program storage
- 256 words of ROM memory with 1024 word capacity
- 1024 words of RAM memory with 2048 word capacity
- 28 TTL I/O lines field selectable as input or output
- Limited expandability on ROM, RAM and I/O





MPS-883 THREE-CARD SYSTEM

MPS-883 THREE-CARD SYSTEM SPECIFICATIONS

System (Three 4.5" by 6.5" printed circuit cards)

--One 8811 CPU card
--One 8112 ROM/RAM card
--One 8113-1 I/O card

Connector Requirements

56 pin, 28 position dual read-out on 0.125 centers

CPU Card Includes

8080 CPU
Crystal clock
Address latches, data buffers, and control decode circuits
Power-on and external reset
DMA buffers

ROM/RAM Card Includes

One 1702A PROM (256 bytes) and four PROM sockets
Eight 2102 RAM (1024 bytes) and sixteen RAM sockets
Socket for card expansion circuit. Limit 2 ROM/RAM cards

I/O Card Includes

28 TTL I/O circuits selectable in groups of 4 as input gates or output latches.

CPU

Executes all of the 8080 instructions.
1.6 microsecond time state cycle.

Memory for Data Program Storage (Expandable to Two Cards)

ROM, 1024 word capacity per card.
RAM, 2048 word capacity per card.

Input/Output

Input gates implement the INP instructions.
Output latches implement the OUT instructions.
Requires external address decoding for card expansion.

Interrupt

Single line, synchronized interrupt on CPU card can be optionally wired for multi-level interrupt.
Multi-level Interrupt: Control lines available for external interrupt such as 8118 priority interrupt card.

DMA (Direct Memory Access)

Data, address, and control lines are 3-state disconnected in response to the HOLD input allowing DMA by peripherals.

Electrical Requirements

Refer to individual data sheets and schematics on 8811, 8112, and 8113 for interface and wiring.

Power Requirements

+VSS = +12 volts ±5% @ 80 mA maximum
+VCC = +5 volts ± 5% @ 2 Amp maximum (35mA per ROM, 50 mA per RAM)
GND = 0 volts
-VDD = -10 volts ±5% @ 750 mA maximum (35 mA per ROM)

Hardware Accessories

Compatible with Series 8400 interface cards.
Fits CR5, CR10 or CR19 card racks
PROM's programmable on Series 90 programmers.

Software

MPS 883 hardware is fully compatible with any 8080 software assuming I/O and interrupt can be assigned compatibly. Teletype operating system and system monitor available. Assemblers, compilers and simulators available through computer time-sharing services.

SERIES 90
PROM PROGRAMMER

The Series 90 PROM Programmer is a low cost, portable and highly versatile solution to programming requirements for MOS and bipolar PROMs. Conversational interaction with the operator makes it simple to use in engineering, manufacturing, quality assurance or in the field.

One of a series of Personality Modules may be plugged into the Series 90 to program any PROM now being manufactured and any that could be introduced in the foreseeable future. The unit has the capacity to program PROMs or PROM arrays having up to 4096 words with up to 8 bits in each word. The Personality Modules are readily pluggable modules which contain the specialized interfacing, power supplies and controls required to program specific PROMs or families of PROMs. In many cases a single module enables the user to program several different types of PROMs. The Series 90 Programmer uses the same Personality Modules as its companion Series 91 Duplicator and all modules are interchangeable between these instruments.

The Master Control Unit contains a microprocessor system which gives it the capability to handle the wide variety of PROMs and to interface with TTY, Paper Tape readers or punches, minicomputers and a host of other equipment. Most of these interfaces are available as standard options to the system.

The unit is packaged in a high impact plastic carrying case and weighs less than 18 pounds with a personality card plugged in.

FEATURES

- Simple to operate conversational system.
- Microprocessor controller gives computer power and flexibility.
- PROGRAM, LIST, DUPLICATE, and VERIFY, modes of Operation.
- Unique Program-Verify sequence adapts to needs of each bit.
- A DUPLICATE with advance substitution capability that allows up to sixteen changes to be made in the copy

- Automatic Zero Check of defined address field.
- Hexadecimal Keyboard (0-9, A-F)
- Six Character Hexadecimal Display of Addresses and Data.
- Auxiliary Binary Data Display.
- Quick Load, Zero Insertion Force, PROM Sockers.
- Forced Air Cooling of PROMs and System.
- Fully portable for field or in-plant use.

SERIES 90 Prom Programmer Shown with Modules and Erase Light

Functions

LIST: Data stored in a PROM is read out a word at a time. The address is displayed in Hexadecimal the data is displayed both in Hexadecimal and Binary.

PROGRAM: Keyboard Data is Programmed into the copy PROM. A Hexadecimal character defines each 4 bits at each address location in the PROM. Both address and Data are displayed for verification prior to actual programming. The unit automatically reads the PROM to verify correct programming.

DUPLICATE: Data in a MASTER PROM is automatically programmed into the COPY PROM. Prior to actual programming the operator can enter data corrections for up to 16 words.

VERIFY: Data in a MASTER PROM is automatically compared to data in the COPY PROM. The Programmer halts on a mismatch and displays the address and data in the MASTER PROM (in Hexadecimal) and the data in the COPY PROM (in Binary). The operator can continue comparing beyond the mismatch. Verification of two matching PROMs takes about two seconds.

ZERO CHECK: Once the operator defines the address field over which he will work, the unit automatically checks that field in the PROM and indicates whether or not it is all zeros.

M900 Master Control Unit Includes

Microprocessor Controller with crystal clock and program expansion capability.
Power On/Off Control - Lighted Circuit Breader
Sixteen Key Data Entry Keyboard (0-9, A, B, C, D, E, F)
Seven Control Keys: PROGRAM; DUPLICATE; LIST; VERIFY; RESET; CORRECT; & ENTER
Data Invert Control Switch.
Six Digit Hexadecimal Display
Zero Field Status Light
Cooling Fan
Attache Case
Receptacle and Connectors for Personality Modules

PM9000 Personality Module Includes

Zero Insertion Force PROM Sockets for Master and Copy PROMs
Binary Data Display for Copy PROM (4 or 8 bits)
Specialized Interface Circuits, Regulators and Program Instruction for specific PROM
Control Switches as required to enable special functions.

Physical Characteristics

Housed in a high impact plastic carrying case (21 inch x 12 inch x 4.5 inch)
Maximum weight: 18 pounds

Power Requirements

Factory wired for either 117 V 60Hz, or 220 V 50-60 Hz (100V 50Hz optional)
Maximum Power: 50 watts.

Personality Modules

PM9000 series of plug-in modules allow programming of a wide range of popular PROMs. Consult PROM selection guide and cross reference tables for module selection assistance.

Optional Equipment and Capability

9101 - PAPER TAPE READER - Allows DUPLICATE and VERIFY operations from paper tape. ASCII HEX, ASCII BNPF, and binary tape formats available. Includes reader.

9102 - TELETYPE INTERFACE - Current loop interconnect for ASR33 allows LIST, PROGRAM, DUPLICATE and VERIFY from TTY. TTY punch and reader allow LIST, DUPLICATE and VERIFY paper tape operations with PROM.

9103 - ERASE LIGHT SYSTEM - Ultra-Violet light source with timer for erasing MOS PROMs.

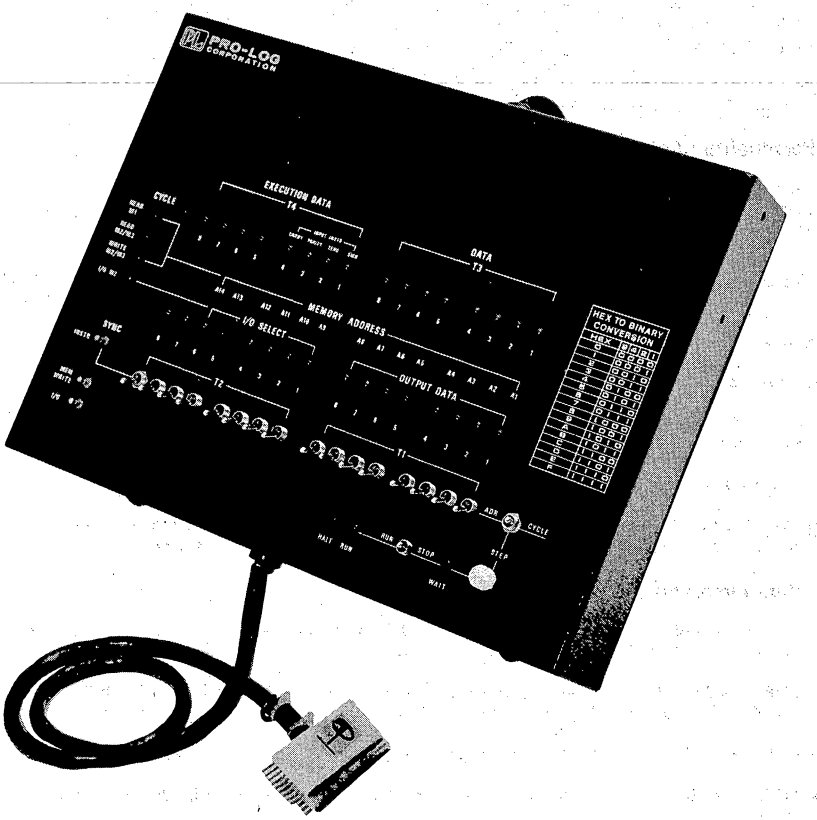
9104 - COMPUTER INTERFACE - Allows remote controller to PROGRAM or LIST the COPY PROM using 8 bit parallel data transfer bus.

9105 - RS232 INTERFACE - Allows remote controller to PROGRAM or LIST the COPY PROM using industry standard RS232 interconnect. 1200 baud serial ASCII data transfer.

The M-821 System Analyzer is a clip on tester for systems using the 8008 microprocessor. The M-821 allows the designer to debug programs in either a static single step mode or a dynamic run mode. The tester operates by monitoring the 8008 CPU bus and displaying cycle data, CPU status and time state data to allow the designer to analyze program and equipment operation.

FEATURES

- Static STEP and dynamic RUN modes
- Displays T1, T2, T3 and T4 data in STEP
- Displays T3 and T4 data selected by T1 and T2 switches in RUN
- Displays type of cycle; Read, Write, or I/O
- Generates SYNC signal selected by T1 and T2 switches
- MEM WRITE and I/O SYNC signals
- STEP by address or cycle
- Displays HALT, RUN or WAIT
- MOS interface and built in power supply



M821 SYSTEM ANALYZER SPECIFICATIONS

Inputs

All inputs are derived from the 8008 CPU through the clip-on device. The analyzer uses the data bus signals and the time state signals to capture and display the selected operation. The analyzer input circuits are CMOS and add negligible loading to the CPU lines.

Controls

RUN/STOP SWITCH: Selects dynamic run mode or static step mode.

STEP PUSHBUTTON: Allows single step operation when RUN/STOP is in stop position.

ADR/CYCLE SWITCH: Allows single step operation either step by address or step by cycle.

T1 AND T2 SELECT SWITCHES: All T3 and T4 data selection in the RUN mode. Allow memory address SYNC selection.

Displays

CPU STATUS HALT, RUN and WAIT indicators

CYCLE INDICATORS indicate READ M1, READ M2/M3, WRITE M2/M3 memory cycles or I/O cycle. Define meaning of T1 and T2 indicators.

T1 and T2 INDICATORS indicate either memory address or I/O information as defined by the cycle indicators. T2-7 and T2-8 also indicate binary cycle data.

T3 INDICATORS indicate memory data.

T4 INDICATORS indicate execution data depending on instruction.

Sync Points

INSTRUCTION SYNC generates a pulse which occurs whenever the T1 and T2 time states of the 8008 match the T1 and T2 switch settings.

MEM WRITE SYNC generates a pulse each time a memory write occurs.

I/O SYNC generates a pulse each time an I/O instruction occurs.

Power Requirements

60 Hz, 117 V---10 Watts Maximum

Package Dimensions

Height --- 3 inches
Width --- 8.5 inches
Length --- 14.5 inches

Weight

4.5 Pounds

Includes

Operator Manual with diagnostic information on each 8008 CPU instruction.

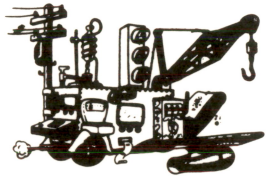
enroll in PRO-LOG'S hands-on MICROPROCESSOR DESIGN COURSE

A 3 DAY COURSE FOR DESIGN ENGINEERS ON **"HOW TO DESIGN WITH PROGRAMMED LOGIC."**

Are you or your company considering microprocessors? Are you a hardware designer familiar with registers and gates, but want to learn how to use microprocessors? If it is your responsibility to develop cost-competitive systems involving logic and control, then this course is for you.

- 3 full days
- Presented by design engineers in engineering terms.
- Learn to develop cost-competitive logic and control systems.
- Develop a cost effective design approach applicable to most microprocessors.
- Go from theory to actual hands-on microprocessor implementation in daily laboratory sessions.
- Only $300 per student.
- Includes textbooks, all lab materials and lunches.
- Informal evening Lab work.
- Enrollment is on a first-come first-served basis and limited to 25 students per session. Telephone enrollment is accepted if followed by check or P.O. within 5 days. Literature will be provided prior to the session to enable the student to prepare.

31



Go Real World with PRO-LOG Logic Processors.

FIRST CLASS
Permit No. 221
Monterey
California
93940

BUSINESS REPLY MAIL

NO POSTAGE STAMP NECESSARY IF MAILED IN UNITED STATES

Postage Will be Paid by:

PRO-LOG
CORPORATION

2411 Garden Road Monterey, California 93940

PRO-LOG
CORPORATION 2411A Garden Road Monterey, California 93940 - (408) 372-4593 TWX: 910-360-7082

NAME _____ TITLE _____

COMPANY _____ DEPARTMENT _____

ADDRESS _____ CITY _____

STATE _____ ZIP _____ TELEPHONE _____

MY REQUIREMENT IS: _____

PLEASE SEND ADDITIONAL INFORMATION

☐ PLS-400 4-BIT PROGRAMMED LOGIC SYSTEM AND AUXILIARY HARDWARE.

☐ MPS-800 8-BIT MICRO-COMPUTER SYSTEMS AND AUXILIARY HARDWARE.

☐ SYSTEM ANALYZER: ☐ 4-BIT ☐ 8-BIT
☐ DESIGN MANUAL: ☐ 4-BIT ☐ 8-BIT
COST $5 .00, CASH OR CHECK PLEASE

☐ 3-DAY DESIGN COURSE.
☐ HALF-DAY ECONOMICS SEMINAR

☐ SERIES 90 PROM PROGRAMMERS
TYPES OF PROMS: _____

☐ I HAVE AN IMMEDIATE NEED. _____

_____

_____

I WOULD LIKE

☐ A DEMONSTRATION OF _____

☐ A COPY OF "PROM USER'S GUIDE"

☐ A SHORT-FORM CATALOGUE SHOWING HARDWARE AND PRICES.

☐ ADDITIONAL COMMENTS: _____

101145 2/76