Microprocessors and Microcomputers

One-chip Controllers to High-end Systems

edited by
Raymond P. Capece, Managing Editor (Technical),
and John G. Posa, Solid State Editor, *Electronics*

McGraw-Hill Electronics Magazine Books

Microprocessors and Microcomputers

One-chip Controllers to High-end Systems

edited by
Raymond P. Capece, Managing Editor (Technical),
and John G. Posa, Solid State Editor, *Electronics*

McGraw-Hill Electronics Magazine Books

ELECTRONICS BOOK SERIES

Also published by *Electronics*

- **Microprocessors**
- **Basics of data communications**
- **Large scale integration**
- **Applying microprocessors**
- **Circuits for electronics engineers**
- **Design techniques for electronics engineers**
- **Memory design: microcomputers to mainframes**
- **New product trends in electronics, number 1**
- **Personal computing: hardware and software basics**
- **Microelectronics interconnection and packaging**
- **Practical applications of data communications**

Library of Congress Cataloging in Publication Data

Main entry under title:
Microprocessors and microcomputers
(Electronics magazine books)
Articles originally published in electronics magazine.
Includes index.
1. Microprocessors. 2. Microcomputers.
I. Capece, Raymond P., 1952- II. Posa, John G., 1953-
III. Electronics.
QA76.5.M5222 001.64104 80-11816
McGraw-Hill Book Company ISBN 0-07-019141-7
McGraw-Hill Publications Company ISBN 0-07-606670-3

McGraw-Hill Publications Co.
1221 Avenue of the Americas
New York, New York 10020

Microprocessors and Microcomputers

One-chip Controllers to High-end Systems

Preface

Many new developments have occurred since *Electronics* published its first collection of articles, "Microprocessors," just five years ago. That volume—and its sequel, "Applying Microprocessors"—brought into perspective the advances made in large-scale integrated circuits as applied to microprocessors like the 8080, Z80, and 6800. But these new tools also brought a raft of challenges: which one to select, how to program them, how to partition the program memory, how to communicate between processors, and how to get the data in and out—to name the most obvious. The microprocessor thus created both the microcomputer industry and various problems of system design.

As both the microprocessor and its markets came to define themselves, new products emerged that sought to provide better solutions to processing problems. The low-end single-chip microcomputer was aimed at adding inexpensive intelligence to appliances and games. Second- and third-generation 8-bit microprocessors came on the scene to reduce the number of parts in systems and to enhance processing power. Not surprisingly, several 16-bit microprocessors—powerful, sophisticated, and as regular and clean in their architectures as any minicomputer—were created to satisfy the most demanding data- and word-processing applications.

At the same time, the task of putting the general-purpose processor to work to solve specific problems fostered a host of LSI devices—peripheral chips. Some are designed to increase processing power, like multipliers and mathematics chips; others, like cathode-ray-tube and disk-memory controllers, unburden the central processing unit from I/O control chores. These peripheral chips, some more complex than the processor itself, have helped spawn the development of new board-level microcomputers that are more powerful than the box-size systems built from the first generation of microprocessors.

Amid all that excitement, however, the industry has been faced with the fact that the production of software—the programs that tailor the microprocessor to the application—could be the limiting factor in the proliferation of microcomputers. Only now has the answer begun to emerge, in the form of high-level languages: programmed with English-like statements, their commands can be interpreted for any of the different processor types.

This book is intended to bring together all these developments and to provide the designer with an overall view of microcomputers today. Its nine parts are organized to reflect the diverse aspects of microprocessor-based system design.

Parts 1 through 4 cover major device types that have entered the market in the past few years. Part 1 introduces the single-chip microcontrollers—like Texas Instruments' 4-bit TMS 1000 and Mostek's 3870 family—that add low-cost, minimum-hardware intelligence to consumer and industrial products. Part 2 discusses the latest 8-bit mid-range microprocessors, like Intel's second-generation 8085 and National Semiconductor's NSC800 family, built with low-power complimentary-MOS technology. Part 3 presents the high-performance 16-bit processors, including Intel's 8086, Zilog's Z8000, and Motorola's 68000. Still another branch of processors, the fast bipolar types, including the various bit-slice devices, are in Part 4.

Part 5 assesses the status of the peripheral support chips that vastly amplify the capability of microprocessors. And although it could have been added to the peripherals section, signal processing is given separate treatment in Part 6 because of its importance. Including American Microsystems' Signal Processing Peripheral, the 2811, and Intel's ground-breaking 2920 single-chip signal processor, this type of circuit represents what will eventually become one of the most important aspects of microcomputing. It lays the foundation for speech-synthesis and -recognition systems, pattern recognition, and other real-time signal-processing applications yet unthought of.

Part 7 covers some of the board-level microcomputers that have emerged over the last few years, including a report on the support products like analog-I/O and math-processing boards several manufacturers have added to their product lines.

Part 8 is devoted to microcomputer software. Spanning this essential topic from an assortment of simple assembly-language routines to the attributes of high-level languages like Pascal and Basic, it provides much useful information on producing software.

Finally, Part 9 brings together a number of microprocessor applications that have appeared in the pages of *Electronics*. In addition to hardware and software aids to using microprocessors found in the Engineer's Notebook section of the magazine, it includes several detailed applications articles, like the ones on error-correction designs and memory-expansion techniques.

—Raymond P. Capece, John G. Posa

Contents

Part 1: Low-End Microcontrollers

Single-chip microprocessor rules the roast **2**
Bill Bell and Deene Ogden

Four design principles get the most out of microprocessor systems **8**
David H. Chung

Single-chip microcomputer expands its memory **17**
Harold W. Dozier and Robert S. Green

Microcontroller includes a-d converter for lowest-cost analog interfacing **23**
W. Check, E. Cheng, G. Hill, M. Hollen, and J. Miller

Microcomputer's on-chip functions ease users' programming chores **29**
William F. Ittner and Jeffrey A. Miller

Expanded package speeds design with new, one-chip microcomputer **34**
D. Starbuck, D. Peeters, K. Hogan, and R. Eufinger

Introducing the Z8: Part 1, one-chip microcomputer excels in I/O- and memory-intensive uses **40**
Bernard L. Peuto and Gary J. Prosenko

Introducing the Z8: Part 2, rich instructions, nine addressing modes make coding easy **46**
Charles Bass, Judy Estrin, and Bernard L. Peuto

16-bit microcomputer is seeking a big bite of low-cost controller tasks **50**
John D. Bryant and Rick Longley

Part 2: Mid-Range Microprocessors

Microcomputer families expand: Part 1, the new chips **58**
Laurence Altman

Third-generation microcomputer set packs it all into 3 chips **69**
D. W. Sohn and Andrew Volk

Compatibility cures growing pains of microcomputer family **74**
Michael Wiles, Fuad Musa, T. Frank Ritter, Joel Boney, and Tom Gunter

P²C-MOS microcomputer family attains n-MOS performance **83**
George Simmons, Rich Burnley, Chuck Seaborg, and Keith Winter

Microcomputer can stand alone or join forces with other chips **91**
David Wayne Smith

Part 3: High-Performance 16-Bit Microprocessors

How a 16-bit microprocessor makes it in an 8-bit world **98**
Mitch Goozé

8-bit microprocessor harbors 16-bit performance **102**
Irving H. Thomae

One-chip CPU packs power of general-purpose minicomputers **107**
Dan Wilnai and Peter W. J. Verhofstabt

8086 microcomputer bridges the gap between 8- and 16-bit designs **112**
B. Jeffrey Katz, Stephen P. Morse, William B. Pohlman, and Bruce W. Revenel

Two versions of 16-bit chip span microprocessor, minicomputer needs **118**
Masatoshi Shima

16-bit 68000 microprocessor camps on 32-bit frontier **126**
Brad Hartman

LSI processor mirrors high-performance minicomputer **134**
Michael Druke, Ronald Gusowski, Edward Buckley, Dean Carberry, Roger March, and Richard Feaver

16-bit microprocessor enters virtual memory domain **142**
Yohav Lavi, Asher Kaminker, Ayram Menachem, and Subhash Bal

Part 4: High-Speed Bipolar Processors

One-chip bipolar microcontroller approaches bit-slice performance **150**
John Nemec

Bit-slice parts approach ECL speeds with TTL power levels **156**
Dale Mrazek

High density raises sights of ECL design **162**
William R. Blood Jr.

How bit-slice families compare
W. Thomas Adams and Scott M. Smith
- Part 1, evaluating processor elements **171**
- Part 2, sizing up the microcontrollers **179**
- Postscript: Microcontrollers serve many needs **187**

ECL accelerates to new system speeds with high-density byte-slice parts **188**
Paul Chu

Part 5: Peripheral Support Chips

Peripheral chips shift microprocessor systems into high gear **196**
John G. Posa

Tough mathematical tasks are child's play for Number Cruncher **210**
Alan J. Weissberger and Ted Toal

Slave microcomputer lightens main microprocessor load **216**
Don Phillips and Allen Goodman

MOS processor picks up speed with bipolar multipliers **220**
Douglas J. Geist

Expandable FIFO buffers improve processor efficiency **223**
Krishna Rallapalli

One chip controls keyboard and display **224**
Lorne Trottier and Branko Matic

Second-generation microcontrollers take on dedicated-function tasks **230**
John Beaston

Single-chip computer scrambles for security **236**
Robert Budzinski

One-chip data-encryption unit accesses memory directly **241**
John Beaston

Two-chip data-encryption unit supports multi-key systems **245**
Thomas Humphrey and Frank L. Toth

Chips make fast math a snap for microprocessors **249**
Krishna Rallapalli and Joe Kroeger

Making mainframe mathematics accessible to microcomputers **254**
John Palmer, Rafi Nave, Charles Wymore, Robert Koehler, and Charles McMinn

Part 6: Signal Processors

Getting the most out of the 9900 for real-time control **263**
Henry Davis

V-MOS chip joins microprocessor to handle signals in real time **267**
Richard W. Blasco

Single-chip n-MOS microcomputer processes signals in real time **275**
M. E. Hoff and Matt Townsend

Software makes a big talker out of the 2920 microcomputer **281**
M. E. Hoff and Wallace Li

Packaging a signal processor onto a single digital board **287**
Louis Schirm IV

Part 7: Board-Level Microcomputers

Microcomputer families expand: Part 2, the new boards **295**
Raymond P. Capece

16-bit single-board computer maintains 8-bit family ties **304**
Robert Garrow, Jim Johnson, and Les Soltesz

Microprocessor bus standard could cure designers' woes **310**
Gordon Force

Memory finds and fixes errors to raise reliability of microcomputer **316**
Alan Heimlich and Joel Korelitz

Special-function modules ride on computer board **321**
Gary Sawyer, Jim Johnson, David Jurasek, and Steve Kassel

Part 8: Software for Microcomputers

Design microcomputer software like other systems—systematically **328**
William F. Dalton

Factoring in software costs avoids red ink in microprocessor projects **333**
Phillip Hughes

High-level languages ease microcomputer programming **338**
J. Lynn Saunders and Larry E. Lewis

Programming microcomputer systems with high-level languages **342**
John G. Posa

Using assembly coding to optimize high-level language programs **350**
Pat Caudill

Forth's forte is tighter programming **354**
Stephen M. Hicks

Language extensions, utilities boost Pascal's performance **359**
Roger R. Bate and Douglas S. Johnson

Pascal software supports real-time multiprogramming on small systems **365**
Roger R. Bate and Douglas S. Johnson

Microcomputer software satisfies conflicting programming needs **370**
James Isaak

Structured assembly language suits programmers and microprocessors **374**
Morris Krieger

M6800 program performs cyclic redundancy checks **379**
S. V. Alekar

8080 program counter makes relative jumps **380**
Prakash Dandekar

8080's stack pointer transfers data blocks fast **381**
Prakash Dandekar

Data-block transfer program is efficient and flexible **382**
Chris Lusby Taylor

8080 program computes 32-by-16-bit quotient **383**
G. W. Switt and J. P. Eisenstein

Programming a microcomputer for d-a conversion **384**
Richard T. Wang

Improved processor program boosts a-d conversion efficiency **386**
Tomasz R. Tanski

Unspecified 8085 op codes enhance programming **388**
Wolfgang Dehnhardt and Villy M. Sorensen

8085 program rapidly computes 8-by-16-bit product **390**
Gary A. Sitton

6500 program automatically sets communications chip bit rate **391**
Michael R. Corder

C language's grip on hardware makes sense for small computers **393**
M. S. Krieger and P. J. Plauger

Part 9: Applications

How to expand a microcomputer's memory **399**
Howard Raphael

Microprocessor makes alphanumeric display smart **402**
William Otsuka

Designing fail-safe microprocessor systems **407**
Dan R. Ballard

Writable control store saves microprogramming time and expense **412**
Joseph A. Oberzeir

Interface processor has two minds to transfer data faster **417**
Alton B. Otis Jr.

Microcomputer-based control smoothes universal motor performance **422**
Tom Slade

Applying the Hamming code to microprocessor-based systems **427**
Ernst L. Wall

Microprocessor reads BCD on only three lines **435**
Darwin T. Scott

Nonmaskable interrupt saves processor register contents **437**
Ivars P. Breikss

Adapting the M6800 processor for automatic telephone dialing **438**
Moshe Bram

One-button controller issues step, run, and halt commands **440**
Robert Dougherty

Hardware breakpoints aid 8080 program debugging **442**
Guy Sundman

Single-step exerciser aids 8085 debugging **443**
Scott Nintzel

Microprocessor adds flexibility to television control system **444**
Kaare Karstad

Avoiding missteps in programming and memory **451**
Norman E. Peterson

Module minimizes repair time of process-control systems **454**
Ralph Foose

Single-loop process controller adapts gains to operating level **458**
Frank Hermance and David Farwell

Monitoring system optimizes apple-tree spray cycle **462**
P. David Fisher and Sigurd L. Lillevik

Interrupts call the shots in scheme using two microprocessors **465**
Thomas A. Harr Jr. and Robert Phillips

High-frequency operation with the AM9513 controller **470**
Terence J. Andrews

Tricked interrupts speed processor's data transfer **472**
G. Rodriguez-Izquierdo

Managing memory to unloose the full power of microprocessors **474**
Jeffrey J. Roloff

Implementing interrupts for bit-slice processors **479**
Vern Coleman

Subject index **481**
Index of devices by number **483**

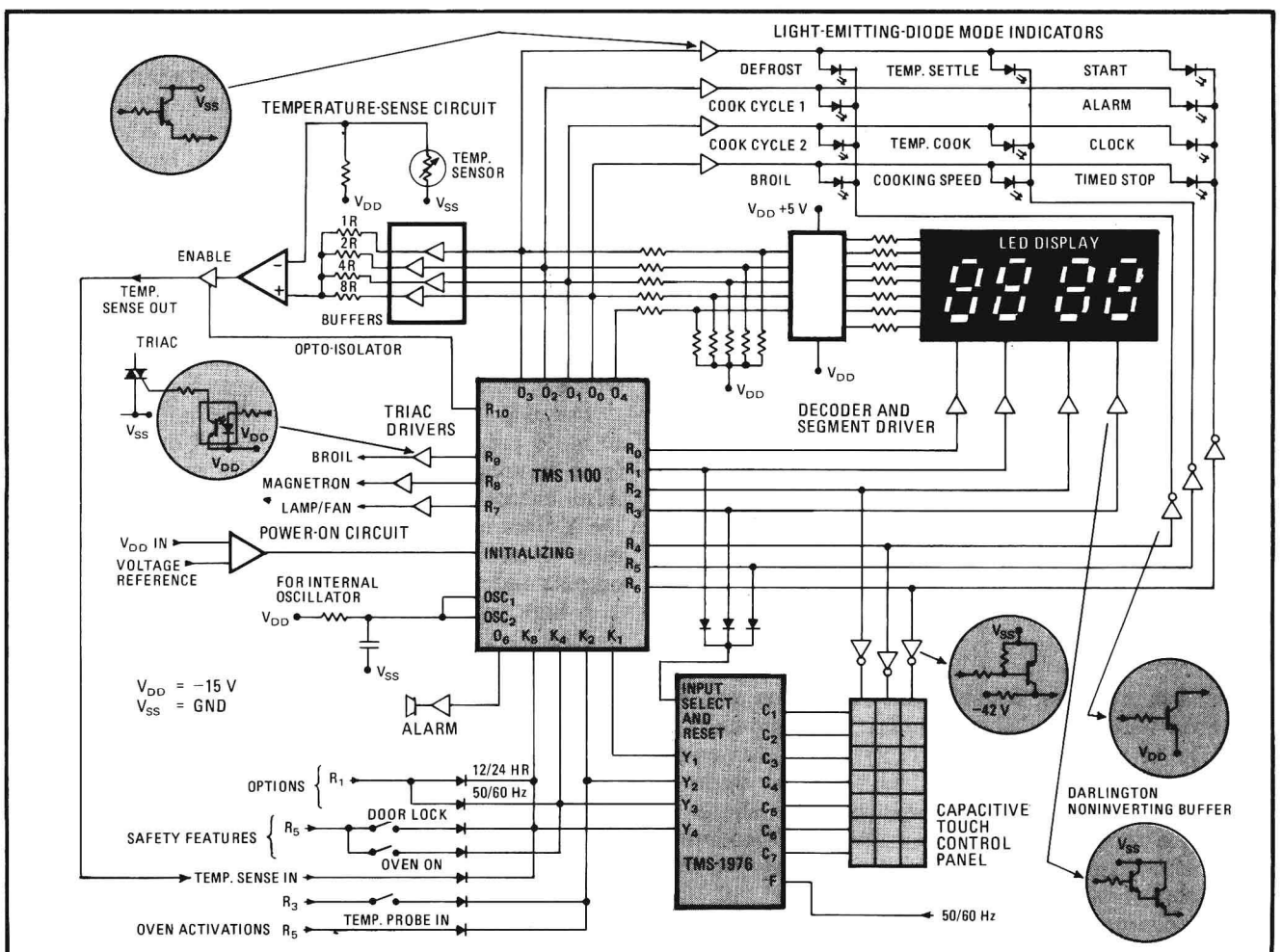
Part 1
Low-End Microcontrollers

Single-chip microprocessor rules the roast

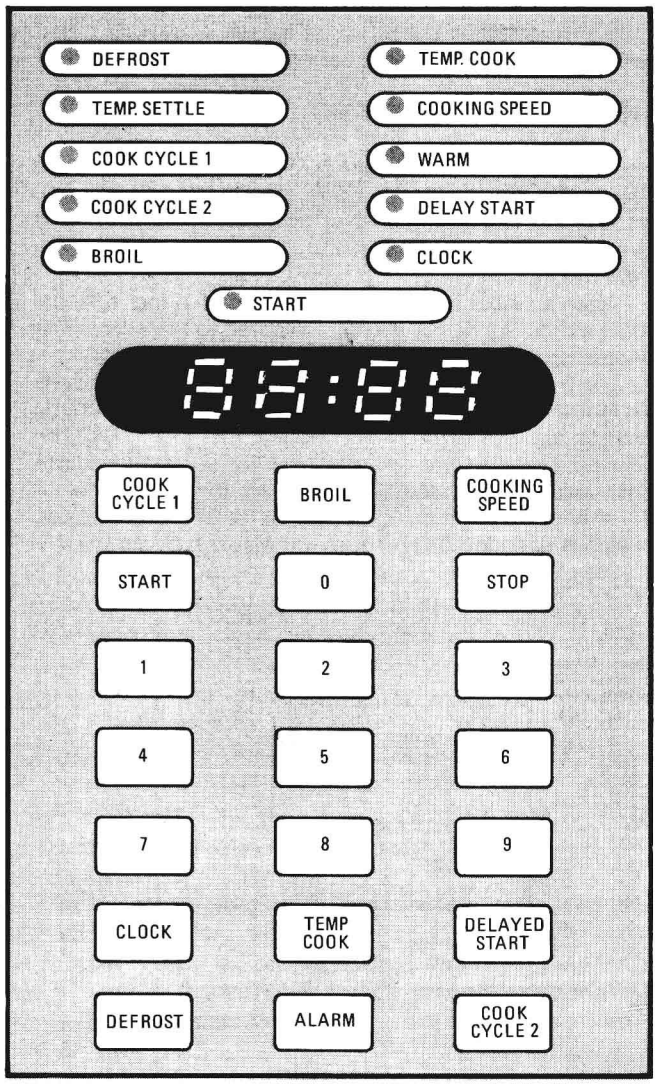
Device controls microwave oven and allows the cook to program cooking cycles, temperatures, and speeds

☐ Millions of golden-brown Christmas turkeys will pop out of microwave ovens this year, and some of them will owe their flavor to microcomputers. These versatile components are displacing conventional electromechanical timers for controlling microwave ovens. They provide the same control functions—timing and heat settings—but they also make life easier for the cook by offering programable sequences of cooking cycles and programable temperature-controlled cooking, as well as programable cooking speeds for these two modes.

The TMS 1100, a one-chip, 4-bit microcomputer (see "Six chips make up the TMS 1000 family,"), is typical of the low-cost microprocessor units controlling microwave ovens. In combination with the TMS 1976 capacitive-touch interface chip, it provides the minimum-chip-count circuitry of Fig. 1. As well as controlling the cooking, the circuitry controls a clock, a digital display and mode lights, the magnetron micro-



1. Oven electronics. The TMS 1100 controls the circuitry for time and temperature cooking methods, generates a precise time base, and controls triacs for supplying ac power to the oven's magnetron, fan, light, and broil element.



2. Touch control. A glass control panel with 21 capacitive switches, a 4-digit LED display, and 11 LED mode indicators is used to enter program commands to place the microwave oven in one of a number of cooking modes.

wave power oscillator, a fan, a light, and the broil element.

The key to the oven's operation is the control panel (Fig. 2). A large, four-digit, seven-segment light-emitting-diode display shows the time of day, temperature settings, or timer settings. Above this display, 11 LEDs indicate the operation the oven is performing or the activation of the clock.

How it works

The cook programs through the 21 capacitive switches below the LED display. For example, to set the clock to 10:20, he or she touches the CLOCK switch, then enters the time by touching 1, 0, 2, 0 (the clock can be set at the factory for 12- or 24-hour cycles). Then she or he pushes the START switch to start the clock's counting. The time is displayed on the four-digit display and may be temporarily displaced during cooking.

Capacitive-touch switches are formed by placing two capacitors in series for each switch on the plate-glass control panel, which also seals the circuitry from the outside environment. Application of the ac-grounded

TABLE 1: FUNCTIONAL OPERATION OF THE TMS 1976

Input set and reset	Detected input	Outputs
L	C_1	L L L H
L	C_2	L L H L
L	C_3	L L H H
L	C_4	L H L L
L	C_5	L H L H
L	C_6	L H H L
L	C_7	L H H H
L	C_8	H L L L
L	C_9	H L L H
L	no key	L L L L
L	reset	L L L L

body capacitance of the cook to the junction of a switch's two capacitors alters the net capacitance, thereby lowering the voltage to the capacitive input lines. The interface chip (Fig. 3) detects the voltage change and encodes it into a 4-bit binary word, which goes to the microcomputer.

What the interface chip actually detects is the absence of a scan pulse from the R-output lines of the microcomputer. The scan pulses generate transitions on the interface chip's input lines, C_{1-9}. These transitions are 0.5 volts more negative than the reference voltage applied to the chip's V_{ref} pin, which permits their detection and then the latching of the level detector's output. When a switch is touched, the input voltage becomes at least 0.3 V more positive than V_{ref}, sending a new logic condition to the interface chip's decoder.

The microcomputer's R lines are buffered in order to drive the keys with a large voltage (-42 V) to make level detection of a switch touch as reliable as possible. Since the R lines make a positive transition when scanning, external inverting buffers drive the control switches, which drive the interface chip's input buffers.

The chip takes the inputs from the buffers by priority so as to prevent generation of invalid encoder outputs from simultaneous touching of two or more switches. After encoding as shown in Table 1, the output goes out the Y lines into the K inputs of the microcomputer. No other interface circuitry is required between the Y outputs and the K inputs.

A high level on the interface chip's ISR (input select and reset) will reset the latches and will maintain the reset until a low level is sensed. The reset is accomplished by the R_1, R_3, and R_5 scans from the microcomputer; the R_2, R_4, and R_6 scans address the control-switch inputs.

The microcomputer's O-output and R-output lines control the four-digit display and the 11 indicator lights. The R outputs strobe the O-output data to the LED display and LED indicators.

The interface chip also provides a time-base input for the clock, the four cooking timers, and the alarm timer. The time-base reference is the frequency of the ac line. A line pulse is recorded every 16.6 milliseconds at 60 hertz or every 20 ms at 50 Hz. This signal is tied to the

Six chips make up the TMS 1000 family

The TMS 1000 series is a family of p-channel metal-oxide-semiconductor 4-bit microcomputers with read-only and random-access memories, arithmetic/logic unit, oscillator, and clock generator fabricated on a single chip. The TMS 1000 (28 pins) and the TMS 1200 (40 pins with two additional outputs) are the basic family members. They have 1,024 instruction words of ROM and 256 bits of RAM. The TMS 1070 (28 pins) and TMS 1270 (40 pins with four additional outputs) interface directly to high-voltage displays of up to 35 volts. Otherwise they are functionally identical to the TMS 1000/1200. The TMS 1100 (28 pins) and TMS 1300 (40 pins with five additional outputs) are extensions of the basic TMS 1000/1200 with 2,048 words of ROM and 512 bits of RAM.

Customers' application programs are reproduced on the internal ROM during wafer processing by a single-level mask. The ROM program controls data input, storage, processing, and output, plus branching, looping, and subroutines. The RAM is used to store input data, flags, and results for later use.

When an input instruction is executed, the four external data inputs, K_1, K_2, K_4, and K_8, are gated to the adder. The inputs can be stored in the RAM for subsequent use.

The R outputs are individually latched and can be used to multiplex inputs, to strobe O outputs or other R outputs, and to address external devices such as memories. The O-output latches are set when a transfer-data-to-output instruction is executed. This transfers the 4-bit accumulator and 1-bit status-register contents to the O register, which is decoded by a user-programable logic array.

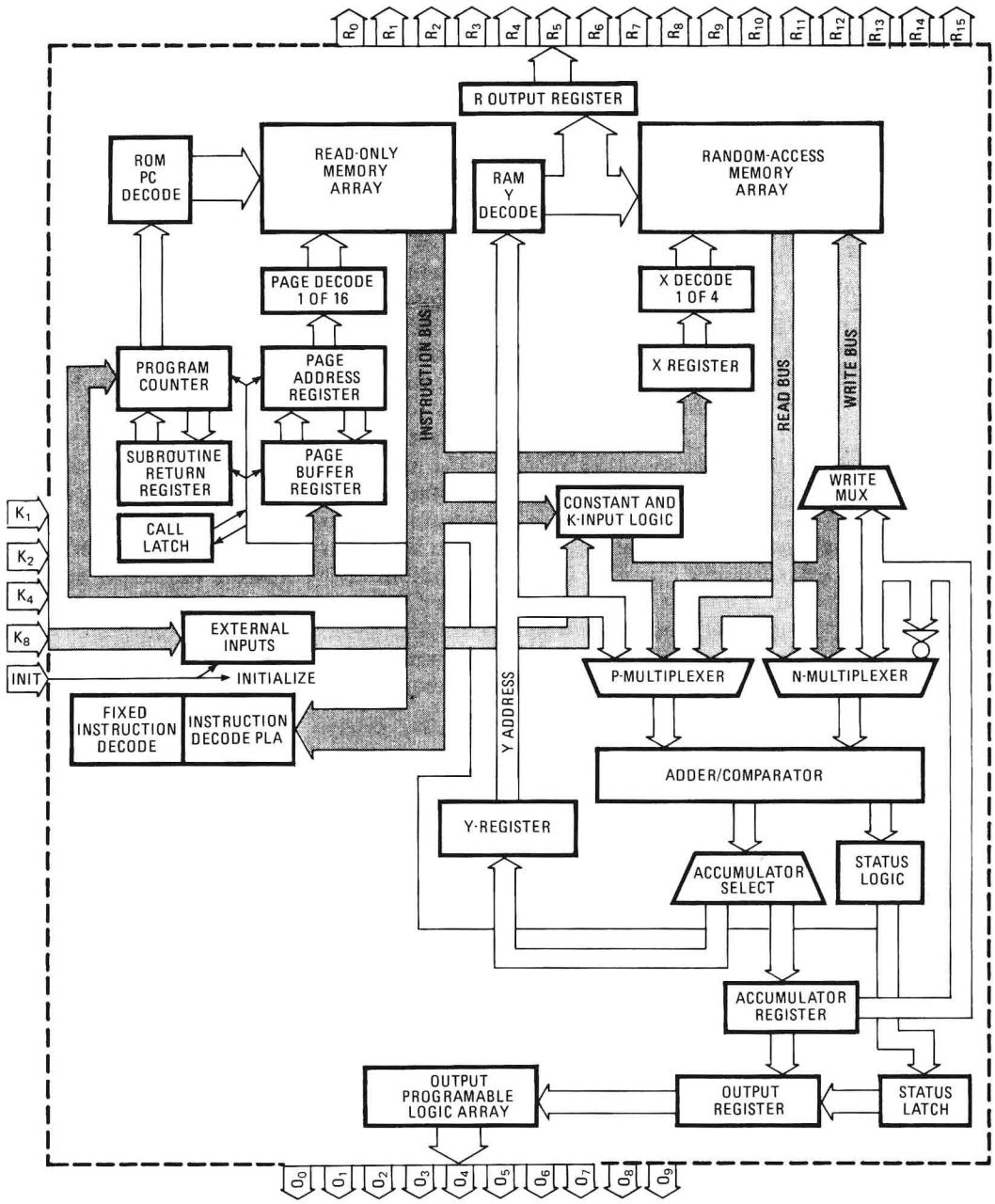
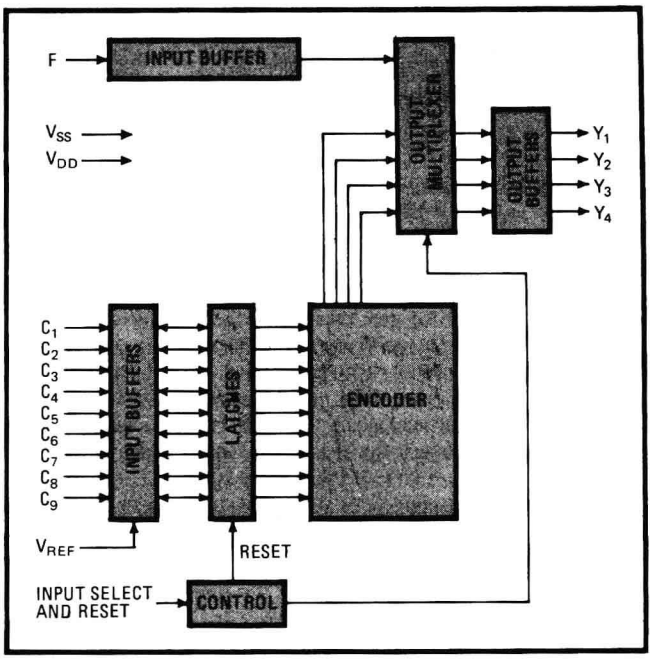


TABLE 2: CONVERSION TABLE FOR TEMPERATURE COOKING

Keyboard value	Cooking temperature (°F)	Type of cooking
1	120	extra rare
2	130	rare
3	140	medium rare
4	150	medium
5	160	medium well
6	170	well
7	180	high temperature 1
8	190	high temperature 2
9	200	high temperature 3



3. Capacitive keying. The TMS 1976 converts keyboard imbalance signals, caused by touch capacitance, to signals compatible with the TMS 1100. Another of the chip's functions is to convert a line-voltage frequency signal, fed in at pin F, to a precise time base.

chip's F (fixed) input pin. A high on the ISR gates the F input to the Y_1 output and resets the C_{1-9} inputs. A low on ISR gates the C inputs.

The microcomputer polls the interface chip, alternately checking the C inputs and the F input. If a high-level input is received on the F input, a high level appears on Y_1 and is transferred to the microcomputer's K_1 input. The microcomputer records the inputs in a subsecond counter used for the clock and the timers that are required for the various cooking cycles.

Interfacing with hardware

Since the system's power supplies settle slowly, an external reset in the power-on circuit avoids false programs when the oven turns on. A voltage comparator circuit holds the initializing input of the microcomputer to a source-voltage level until the power supply reaches the required drain-voltage level, then switches the input to the drain level to start program execution.

The LED digital display and indicators need a full source-to-drain 15-V swing, so interfacing for the microcomputer's O and R outputs is simplified by making the transistor-transistor-logic decoder's ground equal to the −15 V drain. This means the O lines require only a simple two-resistor voltage divider to drive the seven-segment decoder driver. The R lines require one resistor and a Darlington pair to strobe the digits.

Triacs driven by R lines control the magnetron, broiler, lamp, and fan. The R lines are buffered through optical isolators to the gates of the triacs.

The interface chip converts signals from the control panel and the time-base input to a signal compatible with the microcomputer. Other inputs are diode-OR-ed together and are gated to the microcomputer's K lines by its R lines. These inputs are the temperature-sensing circuit, the 50-/60-Hz line-frequency and 12-/24-hour clock options (set at the factory), and the power-on and oven-activation circuits. The microcomputer's algorithm (Fig. 4) controls the actual sequence of operations and subdivides the main program into three separate routines: scan, program, and cook.

After power-up, the program clears the internal RAM and enters the scan routine. This routine serves four basic functions: display, check for 50-/60-Hz input, check for a switch touch, and check for the cooking program chosen. When the scan routine detects a switch touch, it transfers control to the program routine. After entering a cooking sequence, the scan routine transfers control to the cook routine.

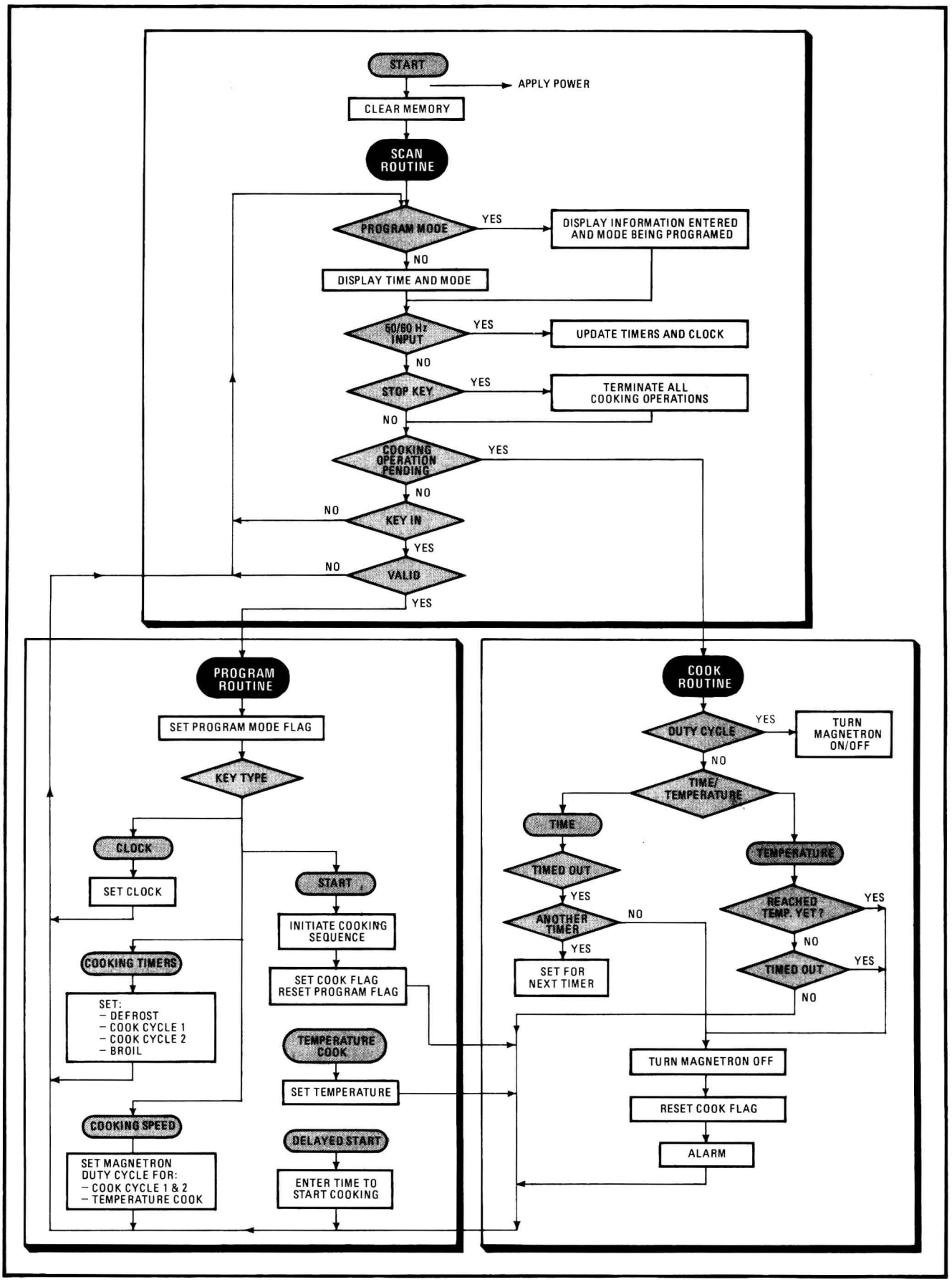
The microwave oven requires about 1,700 program instructions of the 2,048 in the microcomputer's instruction set, depending upon the efficiency in programing the ROM. Similarly, storing four programable timers, two programable power settings, the clock, flags, and other data requires 75% of the available RAM locations

Cooking modes

With any oven, it's possible to cook by time or by temperature. But an oven with a TMS 1100 microcomputer takes over much of the work in both modes.

The timed mode consists of four programable cooking cycles controlled by the microcomputer's timers: defrost, cook cycle 1, cook cycle 2, and broil. The cook programs each of the four to the desired time. For example, to defrost for 25 minutes, he or she touches DEFROST, then 2 and 5, and then either START or another of the cycles. The cycles occur in the sequence listed above, but any of them may be skipped by skipping it in the programing.

Microcomputer control simplifies the task of figuring out total cooking times and turning on the oven at the proper time. For example, suppose a frozen roast is to be served at 6:40 p.m., and it needs 45 min defrosting, 50 min for cook cycle 1, 20 min for cook cycle 2, and 25 min for broil (since microwaves cook from the inside out,



4. Oven algorithm. The oven program is subdivided into scan, program, and cook routines. The scan routine continually monitors the other two routines and transfers control from the program to the cook routine.

Three other firms offer low-cost microprocessors

The dedicated, calculator-oriented, 4-bit microprocessor is rapidly finding itself a niche as the controller for home appliances and electronic cash registers. As well as the TMS 1000 series, three other families are available, from Rockwell International Corp.'s Microelectronic Product division, Anaheim, Calif., National Semiconductor Corp., Santa Clara, Calif., and ITT Semiconductors, Woburn, Mass.

Rockwell's PPS-4/1 family is composed of the MM-76, MM-77, and MM-78, all with read-only memory, random-access memory, and arithmetic/logic unit on one chip. The MM-76 is the firm's simplest and cheapest microprocessor, at less than $5 in volume. It has 640 bytes of ROM and 48 4-bit characters of RAM. The MM-77 has 1,344 bytes of ROM and 96 4-bit characters of RAM, and the MM-78 has 2,000 bytes of ROM and 128 4-bit characters of RAM. All three devices have 31 input/output lines. Rockwell has major contracts from microwave-oven makers for the MM-76 and MM-77, with the lower-cost device proving more popular. The MM-78 is suitable for electronic cash registers and word processors.

National's entry is its calculator-oriented process system (COPS), composed of the two-chip 5781/5782 and the single-chip 5799 and 57140. The 5781/2 is aimed at sophisticated applications such as electronic cash registers, rather than at appliances. The set, which costs around $12, has 23 I/O lines, a 2,048-by-8-bit ROM, and a 160-by-4-bit RAM. The 5799, which is suitable for a microwave oven, has 21 I/O lines, a 1,536-by-8-bit ROM, and a 96-by-4-bit RAM. In large quantities, it costs around $5. The 57140 costs less than $3 in large quantities and is intended for less complex products than multiprogram ovens. It has 18 output and 7 input lines, a 630-by-8-bit ROM, and a 55-by-4-bit RAM.

The newest entry is the ITT 7150, designed specifically for appliance control [*Electronics*, Sept. 16, 1976, p. 138]. It already is in extensive use as the controller of many European programable washing machines, and is just beginning to penetrate the same U.S. market. It has two ROMs with a total capacity of 2,000 bits and 17 input and 10 output lines. In order to keep costs down, it does not use a RAM. Price range is $4 to $8, depending on package, quantity, program, and so on.

Jerry Lyman, Packaging & Production Editor

broiling at the end will give the roast the desired outer browning).

The cook simply enters the cooking program, touches DELAYED START, enters the time it is to be finished, and touches START. The microcomputer will subtract the total program time of 2 hr, 20 min from 6:40 p.m. and will turn on the oven at 4:20 p.m. All the cook need do is put the roast in the oven when setting the program. If he or she has waited until after 4:20, the program is flagged as an error, and the panel's delayed start LED indicator and digital readout flash to alert the cook.

Temperature control

Turkeys, large roasts, and similar food items often are cooked by temperature, but with an MPU-controlled oven the cook need not monitor the cooking process. He or she inserts a temperature probe into the meat, and the circuitry in the upper left of Fig. 1 monitors the cooking. When the meat reaches the programed temperature, the oven turns off and the alarm sounds. For safety, the other cooking mode is disabled when the temperature sensor is plugged in.

The cook does not enter the temperature, but a number relative to the desired temperature taken from Table 2, which can be either on the touch panel or in an instruction book. For example, to cook a roast medium rare, the cook touches TEMP COOK, then the relative temperature setting, 3, then START.

As the temperature goes up, the sensor's resistance goes up. A resistor network converts this resistance charge to a voltage change. By feeding its O outputs to a digital-to-analog converter consisting of a 1R, 2R, 4R, 8R network, the microcomputer compares the two voltage changes to control the magnetron's power output. There are nine temperature settings. When the temperature setting reaches the programed value, the magnetron shuts down, and the alarm sounds.

Whether using the time or temperature mode, the cook can vary the speed to control more accurately the extent of the cooking and to provide more uniform cooking. For example, if a 70% cooking speed is wanted, the cook touches COOKING SPEED, then 7. By removing ac power to the high-voltage dc supply of the magnetron at the proper time, the microcomputer keeps the magnetron on for 70% of its 30-second duty cycle. The microcomputer can calculate ten different speed settings in 10% increments.

For proper operation, the magnetron has to be turned on during the peak of the 50-/60-Hz ac line waveform. The microcomputer controls this by detecting the 0 voltage crossover of the ac input power, both on rising and falling. It divides the interval between the crossovers by two to determine the midpoint and switches on the magnetron at the appropriate instant in the next cycle.

Better than a multichip

The TMS 1000 series one-chip microcomputers have several advantages over multichip microprocessors in controlling microwave ovens. With a multichip MPU, an oscillator, clock generator, ROM, RAM, logic interfaces, and output latches all have to be added. Of course, a minimum chip count is essential for high reliability and low cost.

Since high production volume is a factor with microwave ovens, the multichip approach not only presents a substantial increase in parts cost, but also hikes the cost of inspecting incoming devices, stocking parts, and assembling and testing the system. The additional parts count also increases the times for design, printed-circuit layout, and debugging, thereby delaying production start-up. Finally, the relatively simple one-chip microcomputer does not need the software and hardware interrupts and larger program storage and memory area of more sophisticated MPUs. □

Four design principles get the most out of microprocessor systems

Careful attention to programing, partitioning by speed, adjunct-chip use, and networking produce more cost-effective applications

☐ Imagination was the magic ingredient in the creation of the microprocessor. Now, with the architecture standardizing, imagination is even more crucial in the optimal exploitation of these devices. Often a well-thought-out microprocessor application is many times more cost-effective than a brute-force solution.

Imagination can get a big boost from careful attention to certain key aspects of design. An analysis of some outstanding applications suggests that there are four fundamental approaches to the creative utilization of the microprocessor:

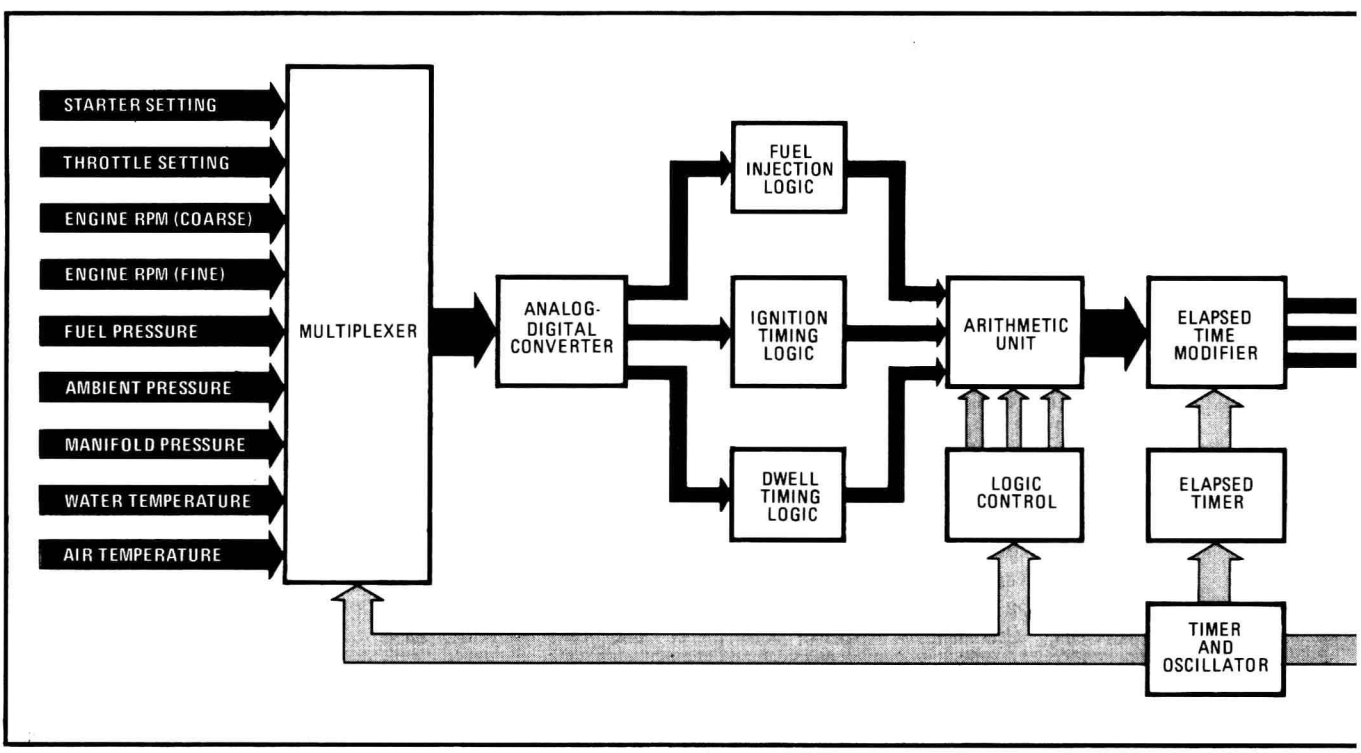
- Minimizing total hardware by clever programing.
- Partitioning the system by speed requirements.
- Using adjunct medium- or large-scale-integrated circuit chips astutely.
- Arranging microprocessors in networks.

These approaches point up the contrast between programing microprocessors and larger computers: specialized software programing of the dedicated chips can minimize hardware and reduce costs, while the goal for the bigger computers is to write as general as possible programs that allow for future options.

In the application of microprocessor technology to high-volume consumer goods such as automobiles, appliances, television sets, telephones, and others where production runs into millions of units per year, the prime factor is cost. Since microprocessors undoubtedly will follow the downward price trend of the calculator chip, the essential question is whether entire systems can be made simple and cost-effective. The answer is emphatically yes, provided that programs are efficiently written to capitalize on all the features of the processor.

Clever programing

Such a system is the automobile-engine controller in Fig. 1 designed to extract maximum energy from the fuel while minimizing pollutants. The strategy is monitoring in real time all pertinent parameters (engine



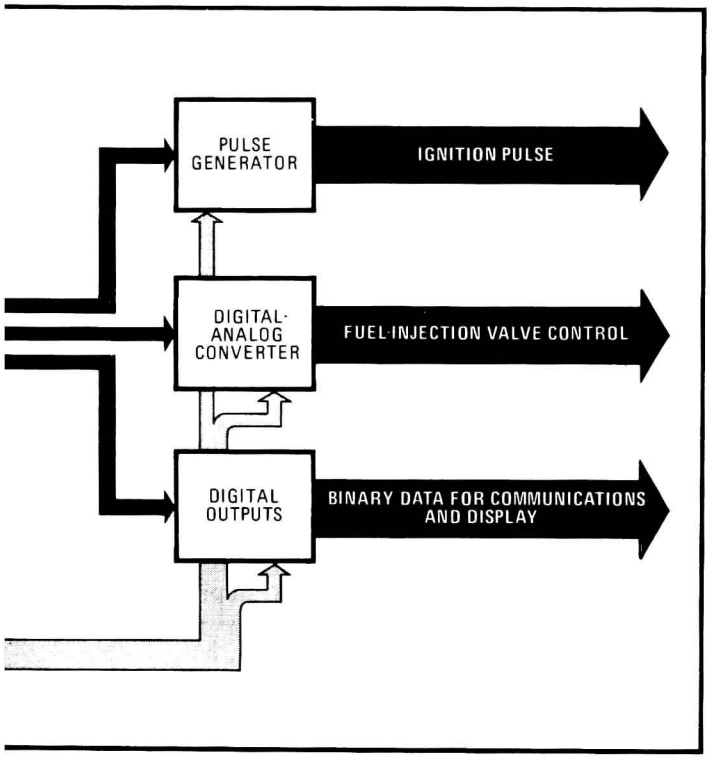
pressures, temperatures, speed, etc.) in order to arrive at the optimum gasoline-air mixture and precise instant of ignition. This process must be repeated at least 100 times a second.

The controller first multiplexes the analog inputs to permit sharing of an analog-to-digital converter. The digitized inputs go to the algorithmic portions for real-time solutions. One of these sections is for fuel injection, another for ignition timing, and the third for optimum spark energy (dwell time). All three blocks share the use of a single arithmetic unit.

These activities are coordinated by the logic control section and sent to the elapsed-time modifier, which provides the correct real-time distribution of the control functions. An elapsed timer supports the modifications. Lastly, the signals are converted to forms suitable for the driven devices: a pulse of precise timing and width is required for engine ignition; an analog signal of proper amplitude must be generated for fuel-injection control, and several digital lines for communications and display must be available.

If this concept is realized by a brute-force approach, the result could be a 200-chip system, hardly meeting the cost objective of a mass-consumer product. Use of a first-generation 4-bit microprocessor might reduce the count to the still-unacceptable level of 100, while a second generation unit can get the count down to 20. But, with a little ingenuity, a controller using the F8 microprocessor can be a four-chip system (Fig. 2). The two-chip F8 microcomputer system includes random-access memory, read-only memory, timer, interrupt hardware, and input/output ports, in addition to the traditional central processing unit.

To achieve minimum cost, the processor chips must assume analog-to-digital conversion, all computation, timing tasks, and digital-to-analog conversion. The microprocessor can perform the computation effortlessly, and a major savings in a-d and d-a circuits can be realized if pulse width is used as the intermediate analog signal at the inputs and the outputs. The most difficult task is keeping the timing under control.



1. Automotive controller. The large number of input and output variables, as well as the complicated sequencing of real-time events, detail what seems to be a microprocessor system with extensive hardware, if not a difficult programing task for a minicomputer.

Many functions must be timed simultaneously. The microprocessor must maintain elapsed time, perform a complete input/output cycle in less than 10 milliseconds, monitor the engine rotation, sample the analog input pulse widths, and generate the correct ignition time and spark pulse width. Perhaps the most stringent timing requirement is measuring the pulse widths of the analog inputs—in particular, the 200-microsecond engine pressure pulse—to an accuracy of ± 1 μs. This is a formidable task for any processor with a timer resolution of 15.5 μs and a tightest counting loop no better than 17 μs.

To accomplish all these functions, the microprocessor's role as a real-time controller must be enhanced. Several methods of enhancement have been devised, which provide valuable insight into next-generation processors.

Enhancing real-time control

First, a "timed binary-search" algorithm has been developed, which quickly resolves the high-speed pulse width in much the same manner as a sampling oscilloscope resolves a very fast waveform. In a standard sampling technique, the processor would sample the waveform every 17 μs, adding a time skew of precisely 2 μs with each "look," in the form of a NO-OP instruction. The information for a complete picture of the waveform would then be given after about a hundred samples.

This process is greatly speeded by using a standard binary search technique, in which the skew is added in binary multiples of 2 μs—17 + 8 μs, then 17 + 4 μs, and so on—resolving the pulse width to ±1 μs after just a few looks.

Second, a method has been devised for greatly improving the accuracy of the interrupt, which can occur randomly during any instruction. This method of registering the instant of interrupt improves timing accuracy by a factor of almost 10. The interrupt routine is able to determine exactly the variable time delay between the instant the interrupt occurs (during an instruction that must be completed) and the time it is activated. It is the precision of the time skew that determines the overall timer resolution.

A third method proves extremely useful when multiple timing functions are controlled by a single microprocessor. Called a time window, the technique makes use of the fact that a measurement will be within a given range of the last such measurement.

For example, if the automobile engine is running at 5,000 revolutions per minute, mechanical inertia ensures that there is no possibility the speed within the next revolution will be faster than 5,500 rpm or slower than