Keeping pace with a single-chip 16-bit microprocessor

by ALAN J. WEISSBERGER
National Semiconductor Corporation
Santa Clara, California

INTRODUCTION

The emphasis in contemporary microprocessor development has been on 8-bit word lengths. Unfortunately, for many applications, the 8-bit microprocessor cannot provide the required accuracy, throughput, programming ease, or flexibility. The multichip 16-bit processor has been cost effective in many of these applications, but has provided unused flexibility or speed (at extra cost) in others. National Semiconductor has developed a single-chip 16-bit microprocessor, the Processing and Control Element (PACE), to provide the benefits of a 16-bit CPU with greater simplicity than the multi-chip design. The benefits accrue from integrating the functions of not only the multi-chip CPU, but also most of the functions that were previously implemented with TTL devices.

In addition, a group of compatible microcomputer chips has been developed to augment the basic processor. A complete microcomputer system, with 1024 words (16,384 bits) of read-only program storage, clocks, buffers and one 16-bit or two 8-bit peripherals is shown in Figure 1. Table I lists features and benefits of this microprocessor.

ARCHITECTURE

The PACE microprocessor, shown in Figure 2, provides 16-bit parallel data-processing capability in a 40-pin package. Functionally, the processor can be segmented into six blocks: Data Storage, ALU, Status, Control, Interrupts, Input, and Output.[1]

Four accumulators, two temporary registers, a program counter, and a 10-word Last-In/First-Out Stack (LIFO) provide ample storage for data manipulation, address formation, and arithmetic computations. Two of the accumulators (AC0, AC1) are principal working registers, while the two others (AC2, AC3) may be used as index registers or auxiliary working registers. The LIFO stack is used primarily to save the program counter during subroutine execution or interrupt servicing. It can also be used to store status information or data. External read/write memory may be used as a stack extension by provision of stack-full and stack-empty interrupts, allowing implementation of a simple stack-service routine.

Arithmetic Logic Unit (ALU) operations include AND, OR, XOR, complement, shift left, shift right, mask byte, and sign extend. Both binary and 4-digit BCD addition capability are provided, thus eliminating the program storage and execution time required to perform BCD to binary conversion. A unique feature of the PACE ALU is the ability to operate on either 8- or 16-bit data, as specified by the programmer through the use of a status flag. This feature allows character-oriented and other 8-bit applications to be implemented and executed using an 8-bit peripheral data bus and read-write memory, while address formation and instruction storage are implemented in the more-effective 16-bit data length.

All status and control bits for PACE are provided in a single Status flag register, whose contents may be loaded from or to any accumulator or the stack. This allows convenient testing, masking and storage of status. In addition, a number of status bits may be tested directly by the conditional branch instruction, and any bit may be individually set or reset. The byte flag is used to specify an 8-bit data length for data processing instructions, while arithmetic operations for address formation remain at the 16-bit data length. In the 8-bit data mode, modifications of the carry, overflow, and link flags are based on the 8 least significant data bits only. Four flags (bits 11-14) that may be assigned functions by the programmer are provided. These flags drive output pins and may be used to directly control system functions or as software status flags.

Six levels of prioritized vector interrupts are available. This allows automatic identification of an interrupting device's level by trapping to a dedicated location in an interrupt pointer table. The pointer specifies the starting address of the interrupt service routine for that particular level. All devices on a given level can be enabled or disabled as a group, independent of other interrupt levels. This permits a fast responding peripheral device on a high level to interrupt a slower peripheral device on a lower level. An individual interrupt enable is provided in the status register for each level (IE1 to IE5), and a master interrupt enable (IEN) is provided for all five lower priority levels as a group. The level-zero interrupt is an exception to this procedure. It is the highest priority interrupt in the system and cannot be locked out by the master interrupt enable. This interrupt level is typically used by the control panel, which can then interrupt the

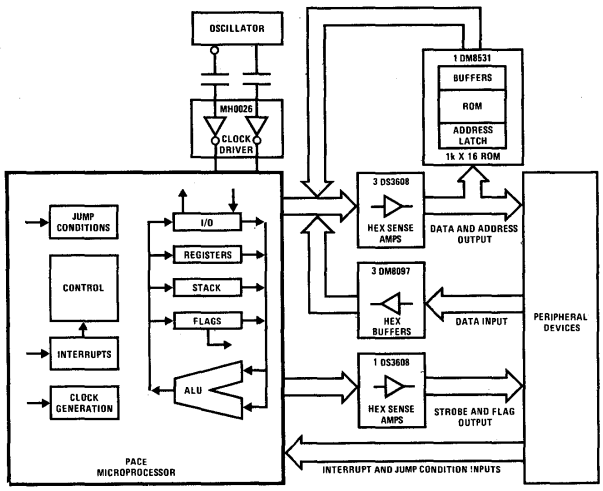

Figure 1—PACE system with 1k×16 ROM—A complete data processing system can be built with PACE, a clock driver and input and output buffers. The control program is stored in a one-chip ROM

application program without affecting system status. It could also be used as an indication of a catastrophic condition such as a power failure. In this case the processor would save its internal registers in a non-volatile or battery supplied memory and halt execution in an orderly fashion.

The minimal package count required to implement a microprocessor system using PACE and its support chips may be important in some applications independent of the associated lower cost. Hand-held or portable equipment may have physical constraints that can only be met by a processor component family of parts. Low power dissipation may also be important in some applications, and the use of a MOS microprocessor with CMOS or lower power TTL support chips may be required.

Some applications that might benefit from the small size, weight, and power requirement of the PACE microcomputer system include remote sensing systems, weather-monitoring stations, and natural-gas pipelines. In each case, a minimum PACE microcomputer system could be installed at an unmanned site. Information could be sensed, collected, and processed locally before being sent to a central computer or recorded on a cassette. Local control and preprocessing reduces data transmission costs because only tested and verified data is sent.[4] These unmanned microprocessor-based systems could also run calibration and diagnostic tests of the remote instrumentation to determine whether or not it is functioning properly.

The ability to operate on either 8- or 16-bit data can be a great advantage in terminals and communication processors. Eight-bit characters can be extracted and processed in the 8-bit mode of operation without packing and unpacking overhead software. Line monitoring, statistical tabulations and error control may be implemented using 16 bits. The PACE CPU can be conveniently interfaced to a byte-oriented peripheral (CRT) and to equipment that has a data length exceeding 8 bits (card reader).

Command outputs and external status inputs are implemented very efficiently using the PACE CPU. The flag outputs can be utilized for control functions, such as start reader, rewind, and others in a tape controller. Similarly, the user jump conditions can be used to sense system status conditions, such as end of tape or inter-record gap. A flag and jump condition can be used together as a serial I/O port, eliminating the hardware required to interface to the data bus and to decode the device address. Several flags and jump condition inputs can be used to provide a keyboard scanning function, modem control, or character synchronization in a smart terminal.

The PACE interrupt system can save considerable hardware and software in applications having several interrupts. The on-chip priority logic and vectored branch to the interrupt routine save logic required external to other microprocessors to resolve priority and jam an address vector onto the data bus, or the program storage and execution time required for the alternative scheme of sequentially polling the interrupt status of all devices. Interrupts are essential in applications where alarm conditions or transient conditions must be serviced immediately, such as automobile, process or machine tool control, or plant monitoring. They are useful in many other systems to eliminate the program overhead required to scan asynchronous system inputs, such as a controller for multiple terminals or an intersection traffic-light controller.

The ability to add BCD data eliminates execution time and program storage overhead required to convert BCD to binary data. This is useful in BCD-oriented applications, such as display controllers, electronic cash registers, billing systems, accounting machines, navigation aids, and industrial controllers and test systems.

The compatibility of PACE with the microprogrammable IMP-16 is beneficial in applications where the IMP-16 could serve as a host processor with the PACE being used as a lower-level processor, such as an automated assembly line. Applications where a microprocessor controlled product is available in several models may use the IMP-16 for the more-sophisticated models and the PACE for the less demanding tasks, allowing common software and peripheral interfaces.

Data transfers between PACE and external memories or peripheral devices take place over the 16 data lines (D00-D15); are synchronized by 4 control signals (NADS, IDS, ODS, and EXTEND); and use common instructions. This

TABLE I—PACE Features

• 16-bit instruction word	Addressing flexibility, speed
• 8- or 16-bit data word	Wide application
• 45 instructions	Efficient programming
• Common memory and peripheral addressing	Powerful I/O instructions
• Shares instructions with National's IMP-16	Allows software compatibility
• 4 general purpose accumulators	Reduces memory data transfers
• 10-word stack	Interrupt processing/data storage
• 6 vectored priority interrupt levels	Simplifies interrupt service and hardware
• Programmer accessible status register	May be preserved, tested, or modified
• Typical 10μsec instruction execution	High speed
• Can utilize DM8531 1k-by-16 ROM	Single memory package
• Single-phase true and complement clock	Minimum external components

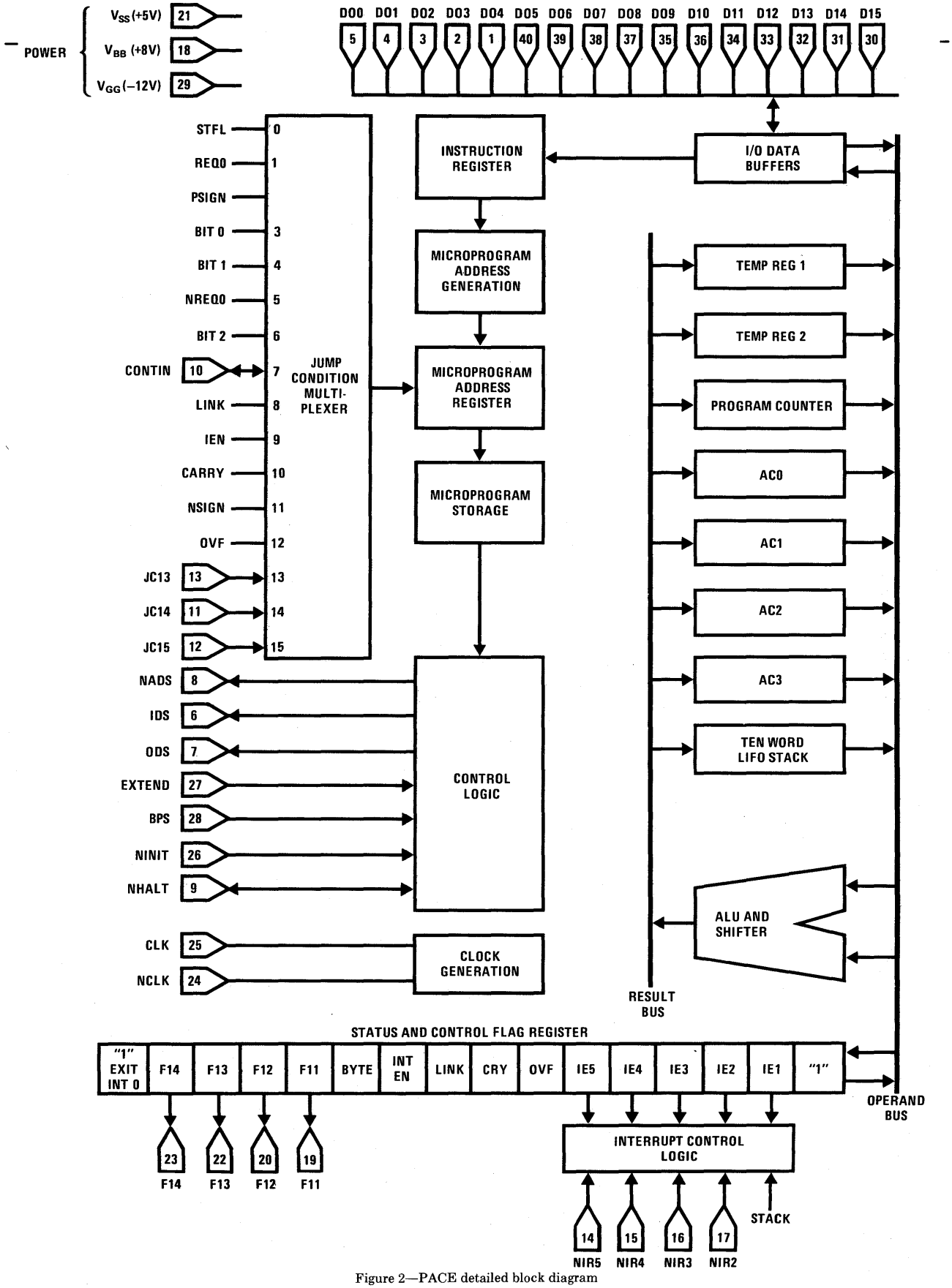

Figure 2—PACE detailed block diagram

```
CONST:  .WORD  X'FFFF     ; CONSTANT FOR DOUBLE PREC. ADD
START:  LI     R1, 0      ; CLEAR RESULT REGISTER
        LI     R3, 16     ; LOOP COUNT TO AC3
        CAI    R0, 0      ; COMPLEMENT MULTIPLIER
LOOP:   RADD   R1, R1     ; SHIFT RESULT LEFT INTO CARRY
        RADC   R0,R0      ; SHIFT CARRY INTO MULTIPLIER
                          ; AND MULTIPLIER INTO CARRY
        BOC CARRY, TEST   ; TEST FOR ADD
        RADD   R2, R1     ; ADD MULTIPLICAND TO RESULT
        SUBB   R0, CONST  ; ADD CARRY TO H.O. RESULT
TEST:   AISZ   R3, -1     ; DECREMENT LOOP COUNT
        JMP LOOP          ; REPEAT LOOP
```

Figure 3—Multiply routine

unified bus architecture is in contrast with many other microprocessor or minicomputers that have one instruction type (I/O class) for communication with peripheral devices and another instruction type (memory-reference class) for communication with memories. The advantage of the approach used by PACE is that all memory-reference instructions are available for communication with peripherals. For example, the DSZ (Decrement and Skip if Zero) instruction can be used to decrement and test a peripheral device register; the SKAZ (Skip if And is Zero) instruction can be used to test the contents of a status register; LD (Load) and ST (Store) instructions may be used for simple data transfers. This technique can improve throughput and simplify programming.

Data transfer operations are initiated by an address data strobe (NADS), which gates the address to the memory or peripheral. An input or output data strobe (IDS or ODS) follows on the next clock cycle. The appropriate strobe is used to gate the data into or out of the processor. The memory device shown in Figure 1 provides address latches on the chip. Two 8-bit bidirectional TRI-STATE data latches may be provided for the peripheral(s). The EXTEND input allows the I/O cycle time to be extended by multiples of the clock cycle to adapt to a variety of memory and peripheral devices or for DMA bus interfacing. Further functional details are provided in References 2 and 3.

Programming

An 8-bit processor must manipulate multiple registers to form 16-bit addresses, make several memory accesses to fetch multi-byte instructions or 16-bit data and use double precision arithmetic routines to obtain accuracy greater than two decimal places. A 16-bit processor does not suffer from these limitations so that faster, shorter and simpler programs may be written. This is clearly evident in minicomputers where the 16-bit word length is standard.

The sample program of Figure 3 illustrates the efficiency of PACE in data processing applications. The complete instruction set, divided into eight instruction classes, is listed in Table II.

The program multiplies the 16-bit value in AC2 (multiplicand) by the 16-bit value in AC1 (multiplier) and provides a 32-bit result in AC0 (high order) and AC1 (low order). Worst case execution time is under one millisecond.

UNIQUE FEATURES

Many of the features of the PACE microprocessor prove beneficial for a wide range of applications, while some provide direct benefits in certain classes of application. The 16-bit instruction and address word lengths and multiple accumulator architecture make programming easier and more efficient. Instructions and operands are fetched in single memory cycles rather than the multiple memory references required for byte-oriented data or instructions. This enhances system throughput and improves program execution times. Program storage requirements and development cost reductions sometimes allow more hardware functions to be implemented in software, reducing system cost and making more of the system reconfigurable by software modification.

Certain functions implemented on the chip simplify interfacing by minimizing the number of external components for a microcomputer system.

- Internal Clock generation from the true and complement clock inputs eliminates the need for a complicated timing generator.
- On-chip output buffers drive sense amplifiers with TRI STATE capability. This reduces power dissipation and chip size while improving speed.
- Interrupt control logic on the chip improves interrupt response time and saves 15-20 TTL packages that would ordinarily be required for the equivalent function.
- The jump condition multiplexer, status and control flag register are internal functions for sensing inputs and providing outputs directly to the user.

APPLICATION

The ability to efficiently operate on 8 or 16 bit data and perform binary or BCD arithmetic enables PACE to act as a controller or data processor in a complex system environment. In many cases a minicomputer or multiple dedicated microprocessors could be replaced with substantial savings in cost and complexity.

To illustrate the flexibility and power of the PACE microprocessor an application example has been developed. The Plant Security Monitoring System (PSMS), shown in Figure 4, acts as a watchdog by monitoring and in some instances controlling a plant's operation. One PACE CPU acts as a data acquisition/alarm scanner while another PACE CPU is utilized as a central control/acknowledgment terminal. The functions monitored are plant power (peak demand, total consumption, outage) and environmental quality (air contaminants,

TABLE II—PACE Instruction Summary

Mnemonic	Meaning	Operation	Assembler Format		Instruction Format
1. Branch Instructions					
BOC	Branch On Condition	(PC) ← (PC) + disp if cc true	BOC	cc,disp	0 1 0 0 | cc | disp
JMP	Jump	(PC) ← EA	JMP	disp (xr)	0 0 0 1 1 0 | xr | disp
JMP@	Jump Indirect	(PC) ← (EA)	JMP	@disp (xr)	1 0 0 1 1 0
JSR	Jump To Subroutine	(STK) ← (PC), (PC) ← EA	JSR	disp (xr)	0 0 0 1 0 1
JSR@	Jump To Subroutine Indirect	(STK) ← (PC), (PC) ← (EA)	JSR	@disp (xr)	1 0 0 1 0 1
RTS	Return from Subroutine	(PC) ← (STK) + disp	RTS	disp	1 0 0 0 0 0 | 0 0 | disp
RTI	Return from Interrupt	(PC) ← (STK) + disp, IEN = 1	RTI	disp	0 1 1 1 1 1
2. Skip Instructions					
SKNE	Skip if Not Equal	If (ACr) ≠ (EA), (PC) ← (PC) + 1	SKNE	r,disp (xr)	1 1 1 1 r | xr | disp
SKG	Skip if Greater	If (AC0) > (EA), (PC) ← (PC) + 1	SKG	0,disp (xr)	1 0 0 1 1 1
SKAZ	Skip if And is Zero	If [(AC0) ∧ (EA)] = 0, (PC) ← (PC) + 1	SKAZ	0,disp (xr)	1 0 1 1 1 0
ISZ	Increment and Skip if Zero	(EA) ← (EA) + 1, if (EA) = 0, (PC) ← (PC) + 1	ISZ	disp (xr)	1 0 0 0 1 1
DSZ	Decrement and Skip if Zero	(EA) ← (EA) – 1, if (EA) = 0, (PC) ← (PC) + 1	DSZ	disp (xr)	1 0 1 0 1 1
AISZ	Add Immediate, Skip if Zero	(ACr) ← (ACr) + disp, if (ACr) = 0, (PC) ← (PC) + 1	AISZ	r,disp	0 1 1 1 1 0 | r
3. Memory Data Transfer Instructions					
LD	Load	(ACr) ← (EA)	LD	r,disp (xr)	1 1 0 0 r | xr | disp
LD@	Load Indirect	(AC0) ← ((EA))	LD	0,@disp (xr)	1 0 1 0 0 0
ST	Store	(EA) ← (ACr)	ST	r,disp (xr)	1 1 0 1 r
ST@	Store Indirect	((EA)) ← (AC0)	ST	0,@disp (xr)	1 0 1 1 0 0
LSEX	Load With Sign Extended	(AC0) ← (EA) bit 7 extended	LSEX	0,disp (xr)	1 0 1 1 1 1
4. Memory Data Operate Instructions					
AND	And	(AC0) ← (AC0) ∧ (EA)	AND	0,disp (xr)	1 0 1 0 1 0 | xr | disp
OR	Or	(AC0) ← (AC0) ∨ (EA)	OR	0,disp (xr)	1 0 1 0 0 1
ADD	Add	(ACr) ← (ACr) + (EA), OV, CY	ADD	r,disp (xr)	1 1 1 0 r
SUBB	Subtract with Borrow	(AC0) ← (AC0) + ~ (EA) + (CY), OV, CY	SUBB	0,disp (xr)	1 0 0 1 0 0
DECA	Decimal Add	(AC0) ← (AC0) $+_{10}$ (EA) $+_{10}$ (CY), OV, CY	DECA	0,disp (xr)	1 0 0 0 1 0
5. Register Data Transfer Instructions					
LI	Load Immediate	(ACr) ← disp	LI	r,disp	0 1 0 1 0 0 | r | disp
RCPY	Register Copy	(ACdr) ← (ACsr)	RCPY	sr,dr	0 1 0 1 1 1 | dr | sr | not used
RXCH	Register Exchange	(ACdr) ← (ACsr), (ACsr) ← (ACdr)	RXCH	sr,dr	0 1 1 0 1 1
XCHRS	Exchange Register and Stack	(STK) ← (ACr), (ACr) ← (STK)	XCHRS	r	0 0 0 1 1 1 | r | not used
CFR	Copy Flags Into Register	(ACr) ← (FR)	CFR	r	0 0 0 0 0 1
CRF	Copy Register Into Flags	(FR) ← (ACr)	CRF	r	0 0 0 0 1 0
PUSH	Push Register Onto Stack	(STK) ← (ACr)	PUSH	r	0 1 1 0 0 0
PULL	Pull Stack Into Register	(ACr) ← (STK)	PULL	r	0 1 1 0 0 1
PUSHF	Push Flags Onto Stack	(STK) ← (FR)	PUSHF		0 0 0 0 1 1 | not used
PULLF	Pull Stack Into Flags	(FR) ← (STK)	PULLF		0 0 0 1 0 0
6. Register Data Operate Instructions					
RADD	Register Add	(ACdr) ← (ACdr) + (ACsr), OV, CY	RADD	sr,dr	0 1 1 0 1 0 | dr | sr | not used
RADC	Register Add With Carry	(ACdr) ← (ACdr) + (ACsr) + (CY), OV, CY	RADC	sr,dr	0 1 1 1 0 1
RAND	Register And	(ACdr) ← (ACdr) ∧ (ACsr)	RAND	sr,dr	0 1 0 1 0 1
RXOR	Register Exclusive OR	(ACdr) ← (ACdr) ∀ (ACsr)	RXOR	sr,dr	0 1 0 1 1 0
CAI	Complement and Add Immediate	(ACr) ← ~ (ACr) + disp	CAI	r,disp	0 1 1 1 0 0 | r | disp
7. Shift And Rotate Instructions					
SHL	Shift Left	(ACr) ← (ACr) shifted left n places, w/wo link	SHL	r,n,ℓ	0 0 1 0 1 0 | r | n | ℓ
SHR	Shift Right	(ACr) ← (ACr) shifted right n places, w/wo link	SHR	r,n,ℓ	0 0 1 0 1 1
ROL	Rotate Left	(ACr) ← (ACr) rotated left n places, w/wo link	ROL	r,n,ℓ	0 0 1 0 0 0
ROR	Rotate Right	(ACr) ← (ACr) rotated right n places, w/wo link	ROR	r,n,ℓ	0 0 1 0 0 1
8. Miscellaneous Instructions					
HALT	Halt	Halt	HALT		0 0 0 0 0 0 | not used
SFLG	Set Flag	$(FR)_{fc}$ ← 1	SFLG	fc	0 0 1 1 | fc | 1 | not used
PFLG	Pulse Flag	$(FR)_{fc}$ ← 1, $(FR)_{fc}$ ← 0	PFLG	fc	0 0 1 1 | fc | 0

temperature, air flow). Various transducers, thermocouples and sensing devices measure the required analog variables and provide inputs to an analog multiplexer. PACE scans these input points at operator selected time intervals by supplying a point address to the analog multiplexer and starting the Analog to Digital (A/D) converter. When the conversion is complete the data are read, processed, and checked against alarm limits. Critical deviations from normal operating conditions are detected and alarms are sent to the control/acknowledgment terminal. The PACE CPU at the terminal formats and routes the alarm data to an operators display panel. The operator

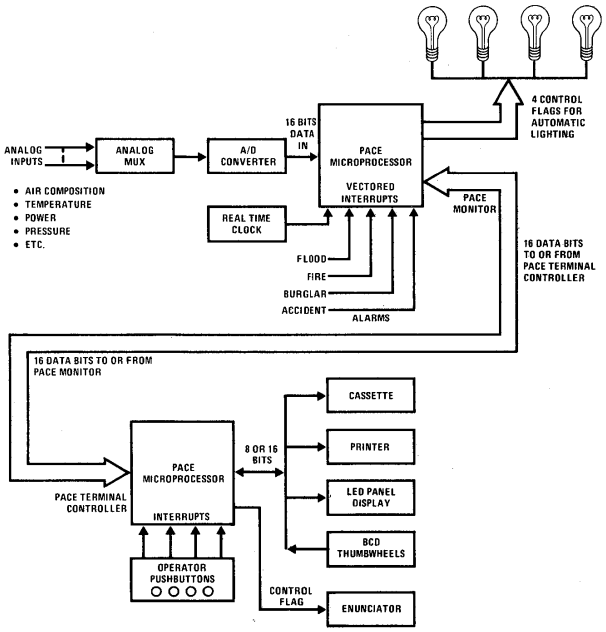

Figure 4—(Application example). PACE as a plant monitor and terminal controller

on duty observes the detected alarm and takes the necessary steps to correct the problem. Some alarms can be detected directly by limit switches, continuity breakage or by manually pressing a button. Examples include floods, fire, burglary, or accident alarms. These "crisis" conditions require immediate attention and would therefore be implanted as prioritized vector interrupts in the PACE monitor. Fast response and immediate operator notification are guaranteed by the sounding of an annunciator horn at the control terminal.

In addition to the above monitoring chores, one or more simple control functions could be provided. For automatic light control, shown in the example, a real time clock generates interrupt signals at fixed preset intervals. The processor recognizes the time of each interrupt and, if appropriate, dims the lights or turns them on or off. Light control commands are facilitated through the four user flags on the chip. This function would conserve energy by providing efficient allocation of electricity. Temperature control of the building by regulating heaters and air conditioners is another possible function that might be implemented as a dedicated application.

The operator at the central control terminal can select various status conditions to be displayed or he can change alarm limits through a set of BCD thumbwheels. Pushbuttons are used as interrupts to get the processors attention. A tape cassette or printer might be provided for record keeping or hard copy outputs. The PACE terminal controller works primarily with 8-bit character data for the supporting peripherals, but it can process 16-bit data from the thumbwheels or the monitor controller. This unique feature (selectable 8 or 16-bit data processing) can be used to efficiently adapt PACE to the function required. Auxiliary functions like trend analysis or signal averaging, could be provided by either PACE microprocessor, depending on the respective data load. Note also that binary and BCD data (thumbwheels and LED's) are processed directly.

CONCLUSION

The PSMS is a solution to a complex problem that is common to all industries. This application offers PACE as a system solution to a multitude of specific tasks. These tasks would ordinarily be done manually, with reduced efficiency, or electrically, with increased complexity and cost. The interfacing simplicity, benefits and low cost of LSI, and the convenience of working with 16-bits promise to make PACE a universal tool in many existing and new applications for microprocessors.

ACKNOWLEDGMENTS

The author would like to thank George Reyling and Gary Miller for their valuable contributions in preparation of this paper.

REFERENCES

1. Reyling, George F., *"Single Chip Microprocessor Employs Minicomputer Word Length," Electronics,* December 26, 1974, pp. 87-93.
2. *PACE Data Sheet,* IPC-16A/500D, National Semiconductor Corp.
3. *PACE Users Manual,* National Semiconductor Corp.
4. Weissberger, Alan J., *"Microprocessor as Intelligent Remote Controllers,"* WESCON 74, Session 23.