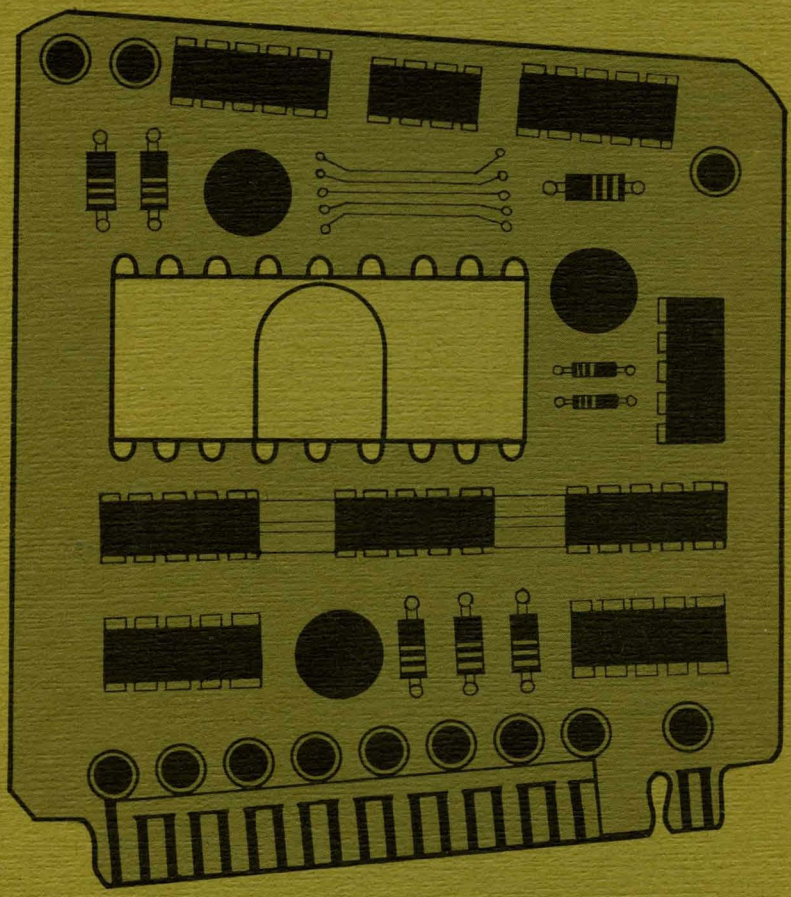
Electronic Engineering
TIMES



Integrated Circuit Applications

JERRY EIMBINDER
CONFERENCE DIRECTOR

PRINTED IN THE U. S. A.

ELECTRONIC ENGINEERING TIMES
280 COMMUNITY DRIVE
GREAT NECK, N.Y. 11021

Electronic Engineering
TIMES

Microprocessor/Memory Proceedings

Integrated Circuit Applications Conference

Part One

See Last Page for Table of Contents

TABLE OF CONTENTS

MICROPROCESSORS

LINEAR/DIGITAL

1.

INTRODUCTION:
EVOLUTION OF THE MICROPROCESSOR

SAM DAVIS
Semiconductor Editor
Electronic Engineering Times
Great Neck, NY

Comparing microprocessor history to a 24-hour clock, it has been said that we are still in the first 30 seconds. The clock started counting in 1969 and since then no fewer than 20 semiconductor companies have entered the microprocessor arena. Some backed into it, some entered with a methodical and planned approach, others have started and abandoned approaches.

"The electronics industry will undergo a significant positive upheaval due to the microprocessor," comments Gordon Moore, president of Intel. "It is yet another step in the increasing pervasiveness of electronics. It is the most significant, revolutionary device since the IC, and will cause a dislocation of roles between the user and the semiconductor manufacturer. The semiconductor supplier has now assumed the role of system designer as he controls the architecture of the microprocessor."

"Father of the microprocessor" is the title bestowed by Moore on Ted Hoff, manager of applications research at Intel. "We were a fairly small company founded in 1968, when Busicom of Japan contacted us in June, 1969," Hoff recalls. "They wanted us to design a group of custom LSI chips for a calculator that would have required 11 36-to 40-pin packages. If we had followed their approach, it would have taxed our design capabilities and their economic goals would have been hard to meet."

Hoff had been working with a DEC-8 at that time and was interested in its use as a scientific calculator. This led to the idea of developing a family of general-purpose LSI that could be programmed as a calculator, or for many other applications. After determining this approach was feasible, the project started in September, 1969. The result was the first general-purpose, LSI, parallel processor family, the now-famous MCS-4 or 4004. The first devices of the three-chipset family were delivered in April, 1971 and the first public announcement came in November, 1971.

Heading up the design effort for Intel was Federico Faggin (pronounced Fa-geen), now president of Zilog, Los Altos, CA. Faggin remarked, "The technology was ready in 1970, and the silicon-gate MOS LSI process (which he co-invented) allowed us to get the curcuit density required for the CPU chip. The idea had been around before, but Intel implemented the first cost-effective, practical family of microprocessor devices. The key was the successful marriage of technology and function. The 4004 had the equivalent of 2,000 transistors; other LSI at that time had a maximum of 1,000

transistors per chip.

A third member of Intel's team at the time was microprocessor department manager Hank Smith, who left Intel in 1974 to become a general partner in a New York City venture capital firm. "It would have been great for us to say we envisioned success for the microprocessor when we started," Smith commented. "We knew there was something there and it was significant, but I don't think any of us ever foresaw how quickly it would catch on and how fast it would grow. Initially it was true that we just let things happen, but after we got started we put together quite an extensive game plan."

The 8008 family was next to be developed by Intel, in conjunction with the then Computer Terminals Corp., now Datapoint. Vic Poor, senior vice president for research and development at Datapoint, recalls that it was about Thanksgiving, 1969 when he and a colleague worked out the architecture for a single MOS chip processor.

"This was to be the heart of a processor the company was designing for what was to become the Datapoint 2200 CRT terminal," Poor explained. "We first brought out the 2200 as a bipolar machine with the idea that we would move into the MOS chip when it became available. We entered into a contract with both TI and Intel late in 1970. TI was well into development before dropping out, while Intel was later in developing the processor than we had planned. In the meantime, TTL prices dropped precipitously and TTL speeds were still faster than MOS LSI. We declared a truce with Intel and said in effect, 'we really don't want the parts anymore, why don't you go market it on your own and call it square with us. In return, we'll let you have the rights to it.' We have stayed with bipolar MSI processors and have not used MOS LSI in two versions of the 2200 and the newer 5500 CRT terminals.

"Work on the 8008 was an on and off proposition," Federico Faggin remembered. "After we started on it, we had to pull some people off to work on a pattern generator to test the 4004. When the tester was finished, work on the 8008 started again in April, 1971 and the first samples were available in January, 1972."

Faggin pointed out that the 8008 chip size was about 146 x 146 mils, taking advantage of the experience gained on the 4004, which had a 136 x 136 mil chip size. "We were careful not to go too far all at once," Faggin added. "In 1971, 170 mils on a side was kind of a 'no man's land.' Today, over 200 mils is not uncommon."

Faggin said he proposed the 8080 as the next generation microprocessor in the summer of 1972 and by November had developed the specs and architecture. The 8080 was to employ N-channel silicon-gate MOS technology, whereas the 4004 and 8008 were P-channel MOS devices. First samples of the 8080

became available in November, 1973. "Compared to the 8008, the 8080 has double the instruction set and is about eight times faster, providing almost 20 times the improvement in throughput. It also has a better interrupt capability," Faggin observed.

"After the 8080 was introduced, things really started happening at Intel," he continued. "From a handful of people who worked on the 4004 and 8008, we had expanded to about 80 people in the microprocessor department when I left in November, 1974."

Next on the list for Intel was the first complete microprocessor family of bipolar devices, the 3000 series introduced in September, 1974. MMI came out with their 5701/6701, four-bit slice microprocessor prior to the 3000 series, but Intel was the first with a complete family of compatible devices for their two-bit slice, bipolar microprocessor. This device was followed later that year with the 4040, an improved version of the 4004.

Microcomputer marketing manager Hal Feeney describes Intel's hardware efforts in 1975 as directed toward reducing chip count and improving throughput of existing microprocessor systems. New devices include a programmable interval timer, programmable DMA controller and programmable interrupt controller. They are also working on devices that will allow four-bit and eight-bit systems to be interfaced for distributed processing applications.

Reflecting on his experience, Hank Smith noted, "It is significant that with microprocessors you build up a group of loyal customers. Once they use your microprocessor, they are not likely to change and use someone else's because of their software investment. This is unlike a memory device where once you design it, you can use anybody's. I feel strongly that the most important thing was to get it designed in as many places as possible. Being first with the microprocessor was much more important than with a memory device."

One overlooked but significant contribution by Intel is their high-level language, PL/M, according to Smith. "It brings programming to the point where anyone can learn it in a short period. Someone who has no experience with computers can program these systems on an increasing level starting with the 8008, going on to the 8080 and then to the 3000. You can write one program and use it on different kinds of systems and products; you don't have to start from scratch each time, only recompile."

Gordon Moore credits Smith with originating the Intellec program development system concept that followed the use of simulation cards for microprocessor system design. "The Intellec family was developed to make it as easy as possible to use the microprocessor family," Smith explained. "We felt it was difficult for a circuit designer to make the transition

to computer user, and the Intellec simplified this. It was a new thing for a semiconductor company to produce complete hardware systems of this kind."

Feedback from customers told Intel that a development system with more capability than the Intellec would be desirable. Out of this grew the MDS, or microcomputer development system, started in mid-1974. "The idea of the MDS was to extend the development system directly into the prototype or production system environment," according to Mike Maerz, who was involved in the MDS design. "It is a single development system with architecture flexible enough to incorporate development capabilities in all new products. With it, we have an interface flexible enough to allow a wide variety of peripherals to be used. For a low-cost system, a teletype could be used, while a more sophisticated system could employ a floppy disk, line printer or CRT terminal."

"The big difference between older development systems and the MDS is that the MDS can be used without present microprocessors or future systems," Bill Broderick, Intel's MDS product manager pointed out. "The older development systems were for a specific microprocessor. The MDS has 14 spare card slots to accommodate future systems, and allows software debugging through use of interactive, in-circuit emulation (ICE) software. Also, the user has the capability to build and modify his hardware external to the development system. In older systems, he had to change cards within the box when he wanted to modify his system."

Not only did Intel pioneer the microprocessor hardware, they also pioneered documentation support. Gordon Moore noted, "My view is that, in most of the aspects of the microprocessor, being there first meant we had to do everything first. We recognized the need for adequate documentation and have shipped more user's manuals than we have CPU chips. No longer could a semi manufacturer supply a two-page data sheet. With microprocessors, 100-page manuals are commonplace."

The best analogy made by Intel president Moore for the future of the microprocessor was a comparison with the fractional horsepower motor. In the Industrial Revolution, the fractional horsepower motor replaced the water wheel with all its belts as a source of power. Today, the average home might have 40 to 50, he said. Moore sees the home of the future as having at least a dozen microprocessors in everything ranging from refrigerators to stoves and electronic games.

2. RECENT DEVELOPMENTS IN THE DESIGN AND APPLICATION OF A BIPOLAR CONTROL STORE SEQUENCER

STEVE LAU
Technical Staff Member
Signetics Corporation
Sunnyvale, California 94086

INTRODUCTION

The basic structure of a high performance Central Processing Unit (CPU) or a "Smart" controller can be typically classified into two distinct but interactively related functional sections. One section is generally referred to as the Processing Section and the other the Control Section.

With the state of the art in bipolar Schottky technology, high-performance microprocessors are designed to perform functions of the Processing Section. Due to the limitation on pin numbers and chip size, the overall Processing Section is partitioned into several functionally equivalent slices. In today's bipolar microprocessor market, 2-bit and 4-bit slice architecture predominates. Each architecture type has its uniqueness but, in general, a slice contains a group of general-purpose registers, an accumulator, special-purpose register(s), ALU and related status flags. All of these elements constitute the Processing Section of a CPU.

The Control Section of the CPU is more complex in design. Typically this section includes the macro-instruction decode logic, test-branch decode, microprogram sequencing logic and the control store where the microprogram resides. Aside from the microprogram, the remaining portion of the Control Section (macro instruction decode, test-branch decode and sequencing), does not lend itself to efficient partitioning into vertical slices. This is due to the random nature of the logic usually found in the Control Section. However, horizontal functional grouping is possible. For example, the macro-instruction decode and test-branch decode logic can now be replaced by the Signetics FPLA (Field Programmable Logic Array); the random logic traditionally needed to implement the microprogram sequencing can now be replaced by the Signetics Control Store Sequencer; and, of course, the microprogram can be stored in a high-density PROM or ROM such as the 82S115.

GENERAL DESCRIPTION

The Signetics Control Store Sequencer is a low-power Schottky LSI, designed for use in high-performance, microprogrammed systems. The basic function of this device is to set up the microprogram address from which the microinstruction is fetched. All microinstructions are assumed to reside in the Control Store (ROMs, PROMs, or RAMs, in the case of Writable Control Store).

The fundamental philosophy behind the design of the sequencer evolved around three points:

(1) To design an LSI that can handle most of the essential sequencing functions normally required for efficient microprogramming.
(2) To design an LSI that is easy to use.
(3) To design an LSI in a 28 pin DIP and yet maintain a high addressability of 1024 words.

Additional address requirement can be easily met by providing external page

registers, which can be either entirely or partially controlled via the microprogram.

FUNCTIONAL DESCRIPTION

The Control Sequencer architecture is shown in Figure 1. The address register is a 10-bit D-type FF which holds the current address. The register changes state when the CLOCK is in a Low-to-High transistion (edge triggered). The address register can be loaded with different address sources under control of the three Address Control (AC_{2-0}) lines and one Test input line. These sources are:

All 0's for reset
Current address +1 for simple increment
Current address +2 for skip
10-bit branch address from outside
Stack register file output

There is a 4-level Stack Register File and a 2-bit Stack Pointer, both of which respond automatically to operations requiring a PUSH (write to Stack Register File) or POP (read Stack Register File).

The file is organized as a 4-word-by-10 bit matrix and operates as a LIFO. The Stack Pointer operates as an UP/down counter.

A cross section of the activities that take place within various logic elements is shown in Figure 2. The AC_{2-0} and the TEST INPUT are the variables in the chart. This chart may be used to gain a better understanding of the device so that its capability can be fully utilized.

Following is a detailed description of all the possible functions performed by the Control Store Sequencer. Note that the mnemonics are in parenthesis, and the logic state is defined as TRUE = 5V and FALSE = 0V:

AC_{2-0} = 000: TEST & SKIP (TSK)
PERFORM TEST ON "TEST INPUT": LINE

IF TEST IS
FALSE: NEXT ADDR = CURRENT ADDR +1, STACK POINTER UNCHANGED.

IF TEST IS
TRUE: NEXT ADDR = CURRENT ADDR +2, i.e. SKIP NEXT MICROINSTRUCTIONS: STACK POINTER UNCHANGED.

This function is used to facilitate transfer of control based on the result of a test on the "Test Input" line.

AC_{2-0} = 001: INCREMENT (INC)
NEXT ADDR = CURRENT ADDR +1

This function is used to serially sequence the address register by 1. This simple function eliminates the need for providing 10 external address lines to do a BRANCH to next address.

AC_{2-0} = 010: BRANCH TO LOOP IF TEST CONDITION TRUE (BLT) PERFORM TEST ON "TEST INPUT" LINE

IF TEST IS
TRUE: NEXT ADDR = ADDR FROM REG FILE (POP), STACK POINTER

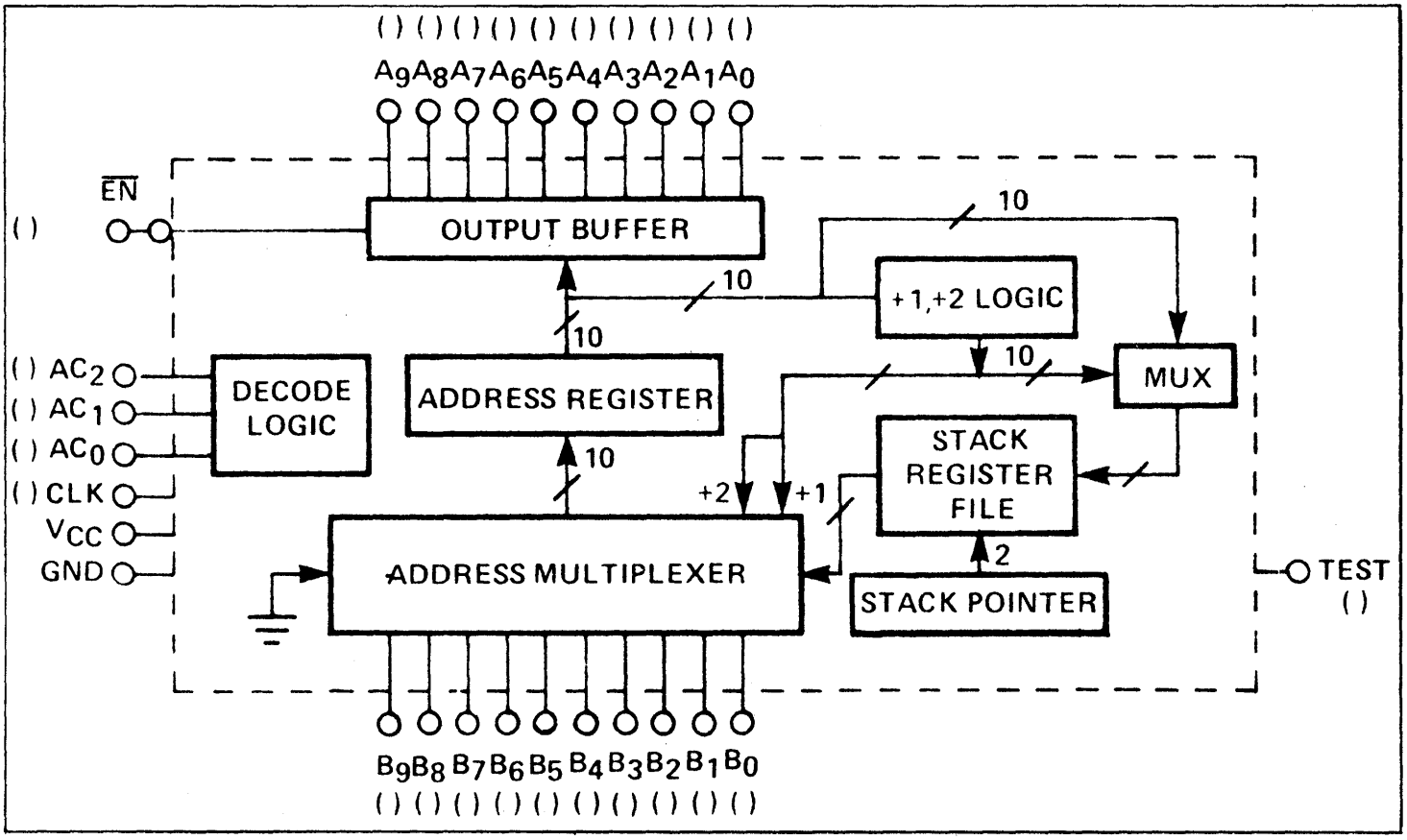


Figure 1 - Control Store Sequencer Architecture

MNEMONIC	DESCRIPTION	FUNCTION $AC_{2\,1\,0}$	TEST	NEXT ADDRESS	STACK	STACK POINTER
TSK	TEST & SKIP	0 0 0	FALSE TRUE	CURRENT +1 CURRENT +2	N.C. N.C.	N.C. N.C.
INC	INCREMENT	0 0 1	X	CURRENT +1	N.C.	N.C.
BLT	BRANCH TO LOOP IF TEST INPUT TRUE	0 1 0	FALSE TRUE	CURRENT +1 STACK REG FILE	X POP (READ)	DECR DECR
POP	POP STACK	0 1 1	X	STACK REG FILE	POP (READ)	DECR
BSR	BRANCH TO SUBROUTINE IF TEST INPUT TRUE	1 0 0	FALSE TRUE	CURRENT +1 BRANCH ADDR.	N.C. PUSH (CURR +1)	N.C. INCR
PLP	PUSH FOR LOOPING	1 0 1	X	CURRENT +1	PUSH (CURR. ADDR)	INCR
BRT	BRANCH IF TEST INPUT TRUE	1 1 0	FALSE TRUE	CURRENT +1 BRANCH ADDR.	N.C. N.C.	N.C. N.C.
RST	SET MICRO-PROGRAM ADDR. OUTPUT TO ZERO	1 1 1	X	ALL O's	N.C.	N.C.

X = DON'T CARE
N.C. = NO CHANGE

Figure 2 - Next Address Control Function

DECR BY 1.

IF TEST IS
FALE: NEXT ADDR = CURRENT ADDR +1, STACK POINTER DECR BY 1.

This function is used as the last microinstruction of a loop (assuming that the beginning microinstruction is a PUSH FOR LOOPING AC_{2-0} = 101). By means of this function, the loop is re-executed or exited depending on the result of the test on the "TEST INPUT" line. If the test is TRUE, the loop will be re-executed by using the address supplied by the Stack Register File. If the test is FALSE, the control exits the loop by moving to the next address. In either case, the Stack Pointer is kept current automatically.

AC_{2-0} = 011: POP STACK (POP)

NEXT ADDR = STACK REG FILE
STACK POINTER DECREMENTED BY 1

This function is used to POP or read the Stack Register File unconditionally. It is usually used as the last microinstruction of a subroutine where the control will be returned to the main microprogram.

AC_{2-0} = 100: BRANCH TO SUBROUTINE IF TEST INPUT TRUE (BSR)

IF TEST IS
FALSE: NEXT ADDR = CURRENT ADDR +1, NO PUSH ON STACK, STACK POINTER UNCHANGED.

IF TEST IS
TRUE: NEXT ADDRESS = BRANCH ADDRESS (B_{9-0}), PUSH CURRENT ADDR +1 → STACK REG FILE, STACK POINTER INCREMENTED BY 1.

This function facilitates the transfer of control based on the result of the test on "TEST INPUT" line. If the test is FALSE, no branch will take place and the next instruction will be executed; if the test is TRUE, the address register is loaded with the B_{9-0} (Branch Address) lines and, at the meantime, the (current address +1) is written or pushed into the Stack Register File. The latter condition allows branching to a micro-subroutine whose beginning address is supplied by B_{9-0} and, at the meantime, the return address is saved in the Stack Register File.

AC_{2-0} = 101: PUSH FOR LOOPING (PLP)

NEXT ADDR = CURRENT ADDR +1
STACK POINTER INCR BY 1
PUSH (CURRENT ADDR)→ STACK REG FILE

This function is generally used as the first microinstruction of a program loop. The current address is saved in the Stack Register File. This function works hand in hand with the BLT function:

AC_{2-0} = 110: BRANCH IF TEST CONDITION TRUE (BRT)

IF TEST IS
FALSE: NEXT ADDR = CURRENT ADDRESS +1.

IF TEST IS
TRUE: NEXT ADDR = BRANCH ADDRESS (B_{9-0}).

This function is used to facilitate transfer of control based on the result of the test on the TEST INPUT line. If the test is TRUE, the next address is supplied by the B_{9-0} lines; if the test is FALSE, the control proceeds to the next address.

AC_{2-0} = 111: RESET TO 0 (RST)

NEXT ADDR = 0, FOR RESET

This function is used to reset the address to all 0's. The state of the B_{9-0} lines has no bearing on the next address setup.

The following additional functions can be performed by the Sequencer, although they are not related to the state of the AC_{2-0} and TEST INPUT:

1. The device will power up to a known state, which is all 0's. This feature can be used to initiate the "power on reset" subroutine.
2. When the external clock is inhibited, all internal register contents are undisturbed. This is a means of retaining the current address (and therefore executing the current microinstruction) for timing delay purposes, where the unit time is equal to the microinstruction cycle time. The clock inhibit signal can be supplied directly by the micro-code or it could be the status condition of certain control logic.
3. The three state output buffers can be disabled (placed in a High-Impedance State) when an external address source is used to access the microprogram. This external address can be a micro-interrupt vector, which directly accesses the starting microinstruction of the interrupt handling subroutine. If Address 0 is to be reserved for initialization, the micro-interrupt vector can be asserted on any address lines other than A_0, such as A_1 A_2 A_3 for 8 levels of interrupts.

HOW TO DESIGN WITH THE 8X02

The 8X02 is totally compatible with all bipolar TTL logic elements. A typical hardware setup is shown in Figure 3. This example generally represents the control loop of a 16-bit CPU. The various major block functions shown in Figure 3 can be described as follows:

1. One FPLA is used to decode the macroinstruction.
2. One FPLA is used to decode the hardware and program status condition.
3. A multiplexer is used to channel one of eight conditions to the "Test" input of the 8X02. The multiplexer select control is directly supplied by the microcode. These conditions may vary from system to system. The multiplexer approach is a simple way to accommodate the multitude of conditions that need to be tested. Note that to force a 1 and 0 can be accomplished by tying the inputs to 5V and 0V, respectively.
4. The 8X02 is used to sequence the microprogram.
5. The eight 82S115 PROMS are used to implement a 1K-word-by-32 bit micro-

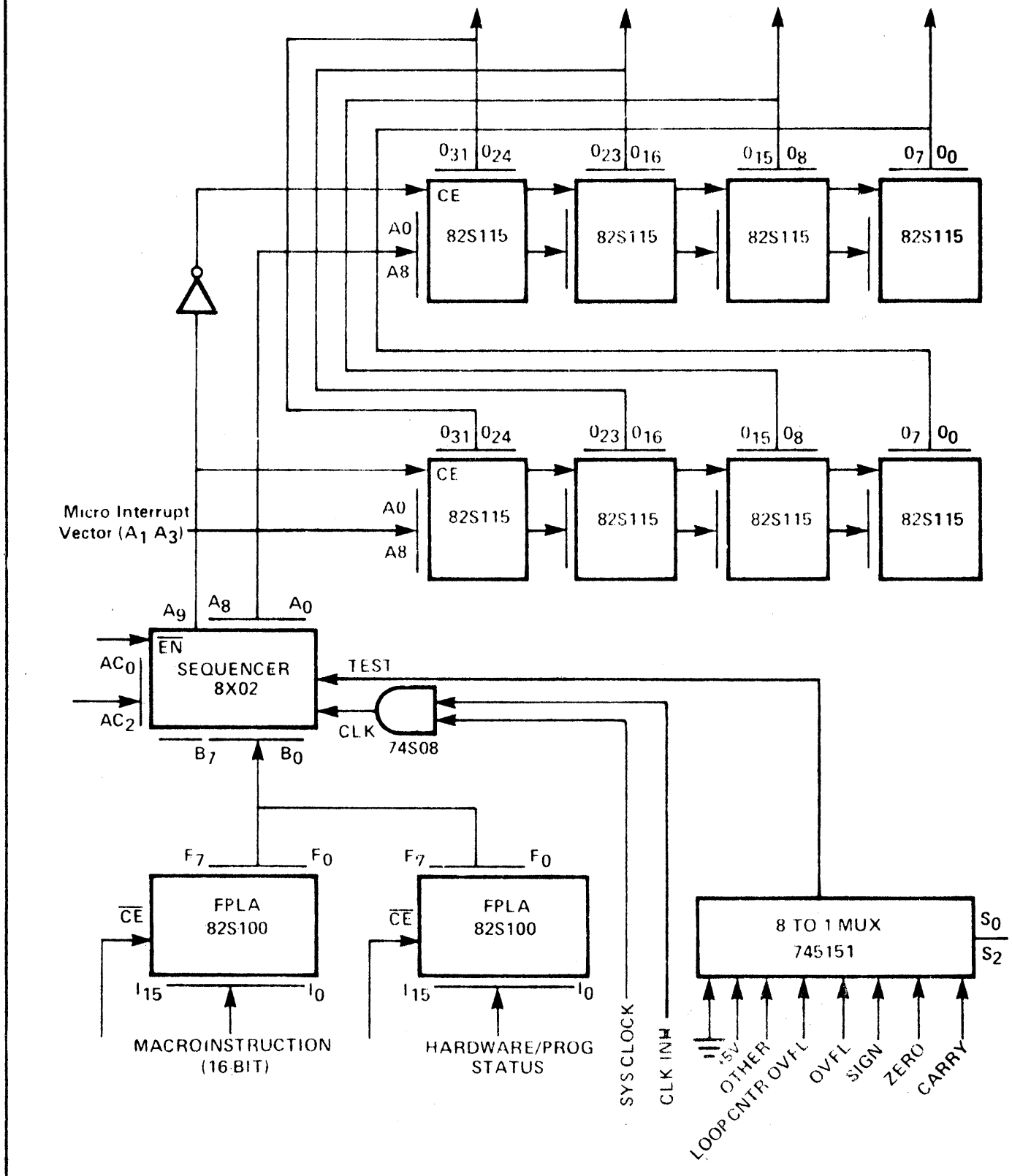


Figure 3 - 8X02 as Part of Control Loop Configuration

program. Additional PROMS may be added as required. The address output signals (A_9 - A_0) of the 8X02 can drive up to 8MA.

To control the 8X02 as it is configured in Figure 3, the firmware basically has to provide fields for:

AC_{2-0}: 3 bits for address control

ACK INH: 1 bit for clock inhibit

S_{2-0}: 3 bits for multiplexer select. In a simpler design, a 1-bit field connected directly to the "TEST" input pin of 8X02 may satisfy the design requirement.

MICROPROGRAMMING CONSIDERATIONS AND EXAMPLES

During the design phase of the firmware (or microprogram), the firmware engineer may find it necessary to allocate certain addresses in the microprogram to handle specific functions which are hardware dependent. For example:

(1) One address each may be assigned as the entry point for subroutine handling, depending on the way that the interrupt vector is connected to the address bus (A_{9-0}).

(2) Address 0 may be assigned to handle system initialization functions.

(3) A convenient number of addresses may be allocated to take care of memory fetch functions as well as sampling (enabling) interrupts.

Example: Test and Skip Programming Technique (Figure 4)

(1) The TSK instruction is used to facilitate transfer of controls.

(2) When executing the TSK instruction, the "Test" input is checked and, if the TEST is TRUE, skip the next instruction and go to address (X+3). If the TEST is FALSE, go to the next address (X+2).

(3) The RST instruction is used to bring the control to address 0 where micro-interrupts can be enabled.

NOTE: Addresses are shown in parenthesis and instructions are shown inside the blocks.

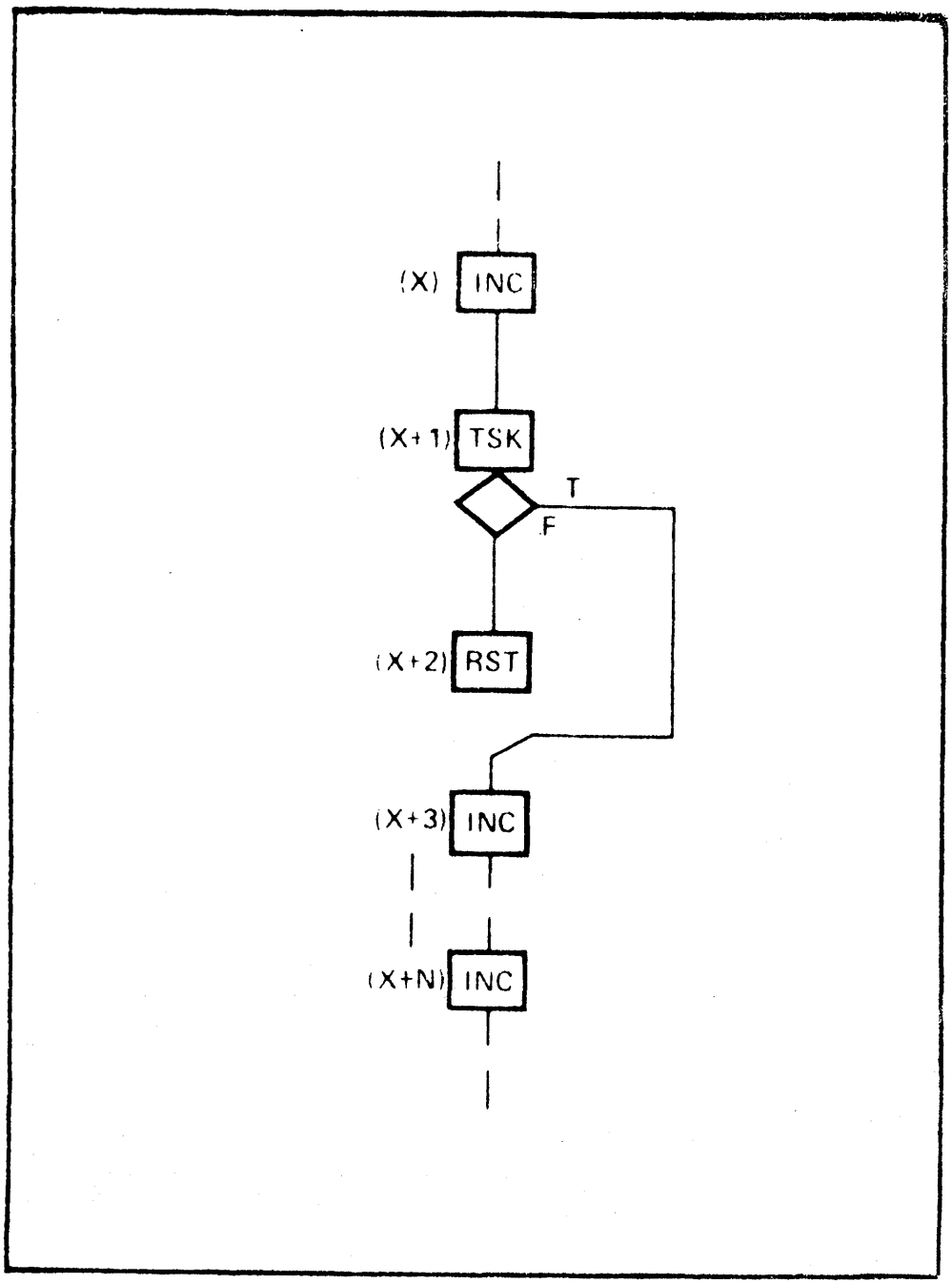


Figure 4 - Test and Skip Programming Technique Example

Example: Conditional Branching Technique (Figure 5)

(1) N-WAY BRANCH within the same 1024 word page is possible.

(2) When the "Test" condition is true, the branch will be taken. The branch address is supplied via B_{9-0}. In the example it is (Y).

(3) When the "Test" condition is FALSE, the control will proceed to the next instruction at address (X+2).

NOTE: Addresses are shown in parenthesis and instructions are shown inside the blocks.

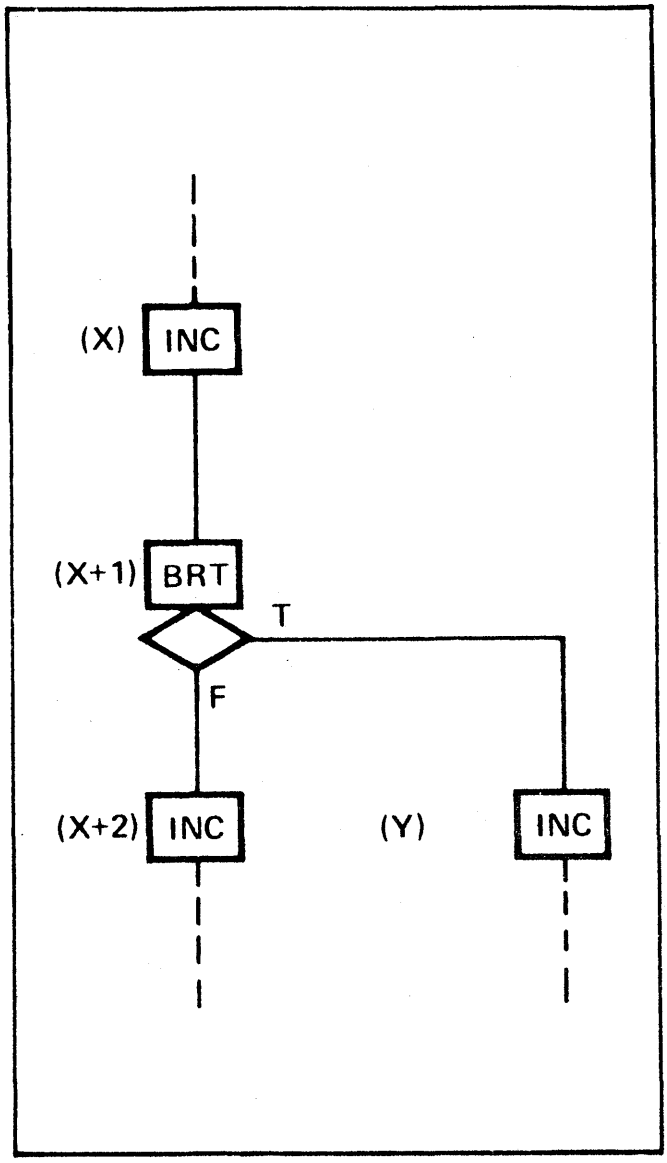


Figure 5 - Conditional Branching Technique Example

Example: Subroutine Nesting Technique (Figure 6)

In this subroutine the beginning address (Y) must be presented to B_{9-0} inputs during the BSR instruction. If the BRANCH is taken, the return address (X+2) will be saved in the Stack Register File. When the subroutine is done, issue a POP instruction to return the main program to address (X+2).

NOTE: Addresses are shown in parenthesis and instructions are shown inside the blocks.

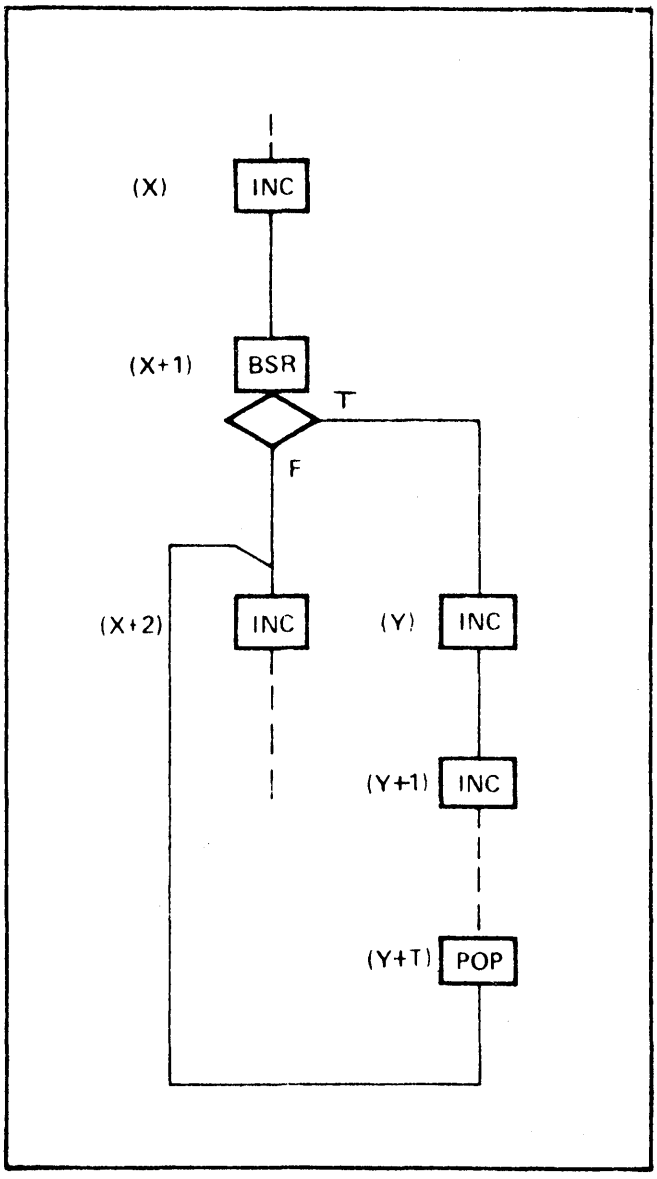


Figure 6 - Subroutine Nesting Technique Example

Example: Program Looping Technique (Figure 7)

(1) The first instruction of the LOOP must be a PLP. During the execution of a PLP, the sequencer pushes the current address (X) into the Stack Register File.

(2) The last instruction of the LOOP must be a BLT instruction. When the BLT is executed, the sequencer checks the "TEST" input which is normally connected to a loop count overflow signal. If the loop counter does not overflow, the loop will be re-executed. If the loop counter does overflow, the next instruction will be automatically accessed and executed.

NOTE: Addresses are shown in parenthesis and instructions are shown inside the blocks.

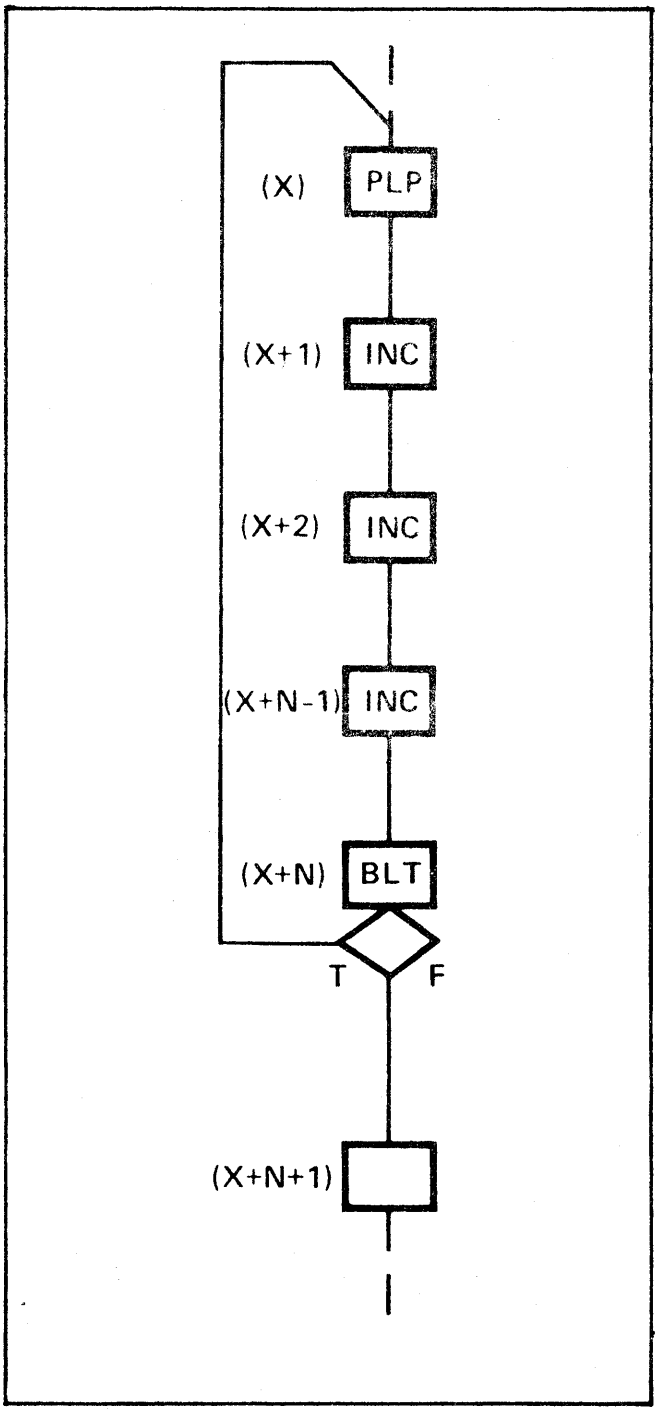


Figure 7 - Program Looping Technique Example

Example: 4-Level Subroutine Nesting Technique (Figure 8)

When applying the subroutine nesting technique, the following can be used as a guideline:

(1) Use the BSR instruction to branch to the subroutine if the "TEST" input of the sequencer if HIGH.

(2) Use a POP instruction to return from a subroutine.

(3) Caution must be exercised to avoid stack overflow or underflow.

(4) A 10-Bit address (beginning address of subroutine) must be supplied to the B_{9-0} during BSR instruction.

NOTE: Addresses are shown in parenthesis and the instructions inside blocks of flowchart.

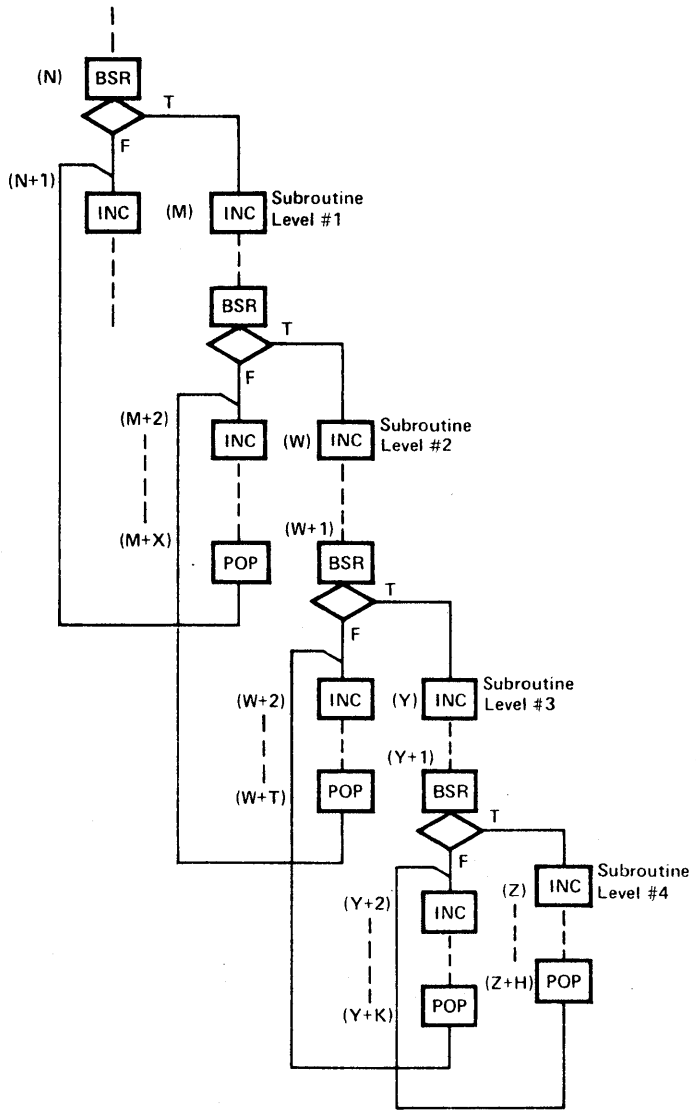


Figure 8 - 4-Level Subroutine Nesting Technique Example

3.

SINGLE CHIP MICROPROCESSOR WITH MINICOMPUTER PERFORMANCE

BY: JOHN D. BRYANT
Microprocessor Systems Engineer
Texas Instruments Incorporated
Houston, Texas

INTRODUCTION

As technology keeps advancing, integrated circuits with lower and lower cost/performance ratios continue to emerge. The introduction of the TMS 9900 microprocessor by TI maintains this trend by offering the performance of a minicomputer at the cost of a single chip. The TMS 9900 is a full 16 bit central processor utilizing advanced high speed N channel MOS technology, and is system engineered to provide the most cost-effective solution to a wide variety of applications.

The philosophy which led to the 9900 was to provide a complete family of total software compatible computers. Current family members include the TMS 9900 single chip microprocessor, the 990/4 microcomputer on a board, and the 990/10 high-performance TTL minicomputer. A program written for one family member will run without reassembly on any other. Thus, the family can provide a single solution to a wide range of applications and reduce significantly the expense involved with training and maintenance. The family concept also lends itself to the efficient design of hierarchical and distributed processing systems.

PRODUCT SIGNIFICANCE

Advanced Features

The TMS 9900 offers several advancements over previously announced components. These include:

1) Single Chip 16 Bit CPU

   A full 16 bit architecture that incorporates the power and efficiency of a 16 bit instruction set on a single chip.

2) Minicomputer Instruction Set

   The instruction set contains 69 basic instructions which include some previously available only on minicomputers (for example, multiply and divide).

3) Memory to Memory Architecture

   This advanced architecture provides 16 general purpose registers (accumulators, index registers, etc.) while maintaining the flexibility of fast and efficient context switching.

4) 17 CPU Vectored Interrupts

   The interrupt structure is designed to allow the interface of 2 special purpose interrupts (RESET and LOAD) and up to 15 user specified interrupts with minimal external circuitry.

5) Simplified Bus Structure

I/O data are transferred on dedicated I/O buses using standard TTL interfaces. The memory data and address buses are separate and need no external demultiplexing circuitry.

6) Standard Peripherals

All signals (excluding clocks) are completely TTL compatible. The architecture is configured to interface directly with standard memories and TTL.

Target Applications

The applications for which these advanced features will be significant are:

1) Interrupt Driven Systems

The TMS 9900 offers very fast context switching and efficient interrupt handling for the design of high performance/low overhead systems.

2) Limited Memory Systems

The power and efficiency of the 16 bit architecture and instruction set tend to require fewer memory bits to perform a given function.

3) High Throughput Systems

The powerful instruction set coupled with the fast context switch enables the processor to achieve high speed task operation with a 3 MHZ clock.

4) Control Applications

The bit-oriented I/O system allows the implementation of versatile control systems with minimal program and interface hardware.

5) 8 Bits or Greater Word Systems

The 16 bit architecture along with the comprehensive subset of byte instructions insures that the 9900 will operate efficiently in 8 - 16 bit data applications (A/D, D/A, Control, Communications).

In other words, the 9900 is designed to perform effectively in a wide variety of applications while minimizing program and hardware overhead.

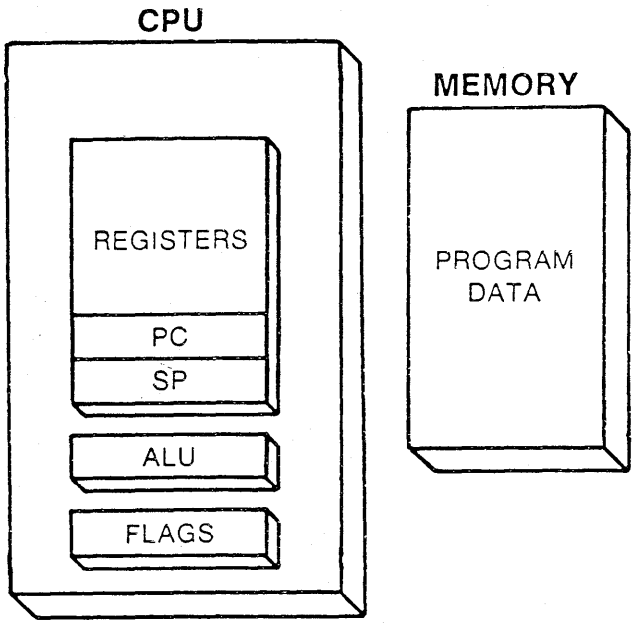
SYSTEM ARCHITECTURE

Machine Architecture

The machine architecture is shown in Figure 1 compared to a currently popular 8 bit microprocessor (8080). The 8080 employs a conventional stack architecture with an 8 bit parallel arithmetic unit, a status flag register, a 16 bit program counter register (PC), a 16 bit stack pointer

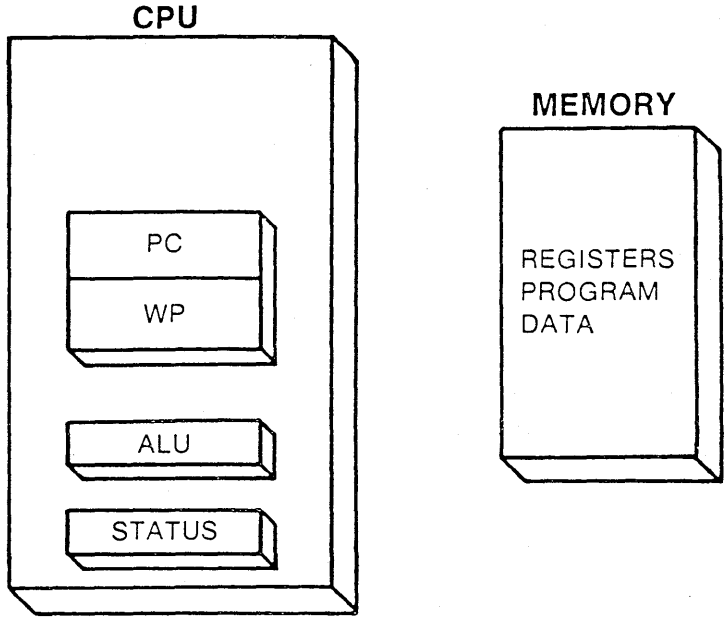
MACHINE ARCHITECTURE

TMS 8080



PROGRAM AND DATA REGISTERS IN CPU PACKAGE

TMS 9900



PROGRAM AND DATA REGISTERS IN MEMORY

- NUMBER OF WORKSPACE REGISTERS LIMITED ONLY BY MEMORY SIZE
- PROVIDES FAST CONTEXT SWITCHING

FIGURE 1

register (SP), and a set of program data registers. The operating program and data are stored in external memory. The TMS 9900, on the other hand, uses an architecture commonly called memory-to-memory. In the processor are a 16 bit arithmetic unit, a status flag register, a PC, and a 16 bit register (WP) that defines a 16 word location in external memory called the workspace. The workspace registers are the program data registers and are stored in the external memory along with the operating program and data. Two advantages of this architecture are (1) the number of workspace registers is limited only by the memory size and (2) since there are no program data registers on the chip that have to be stored, context switching is very fast.

A block diagram of the 9900 is shown in Figure 2. Interface to the outside world consists of the 16 bit bidirectional memory data bus (D0-D15), the 15 bit address bus (A0-A14), the I/O data bus (CRUOUT, CRUIN), interrupt inputs (INTREQ, IC0-IC3), miscellaneous control signals, and power and clocks. Internally the circuit contains the three user accessible registers (PC, WP, and STATUS) along with six additional housekeeping registers. The ALU is a full 16 bit parallel arithmetic unit, and all internal buses are full 16 bit parallel buses. The control logic consists of a microsequence ROM controller employing 8.5K bits which is not user microprogrammable.

Memory Organization

All memory addresses are 16 bits and specify the location of a byte of data. The memory is organized into 16 bit words (2 bytes/word), and thus the 16 bit address can directly specify 64K bytes or 32K words. Since each memory access results in 2 bytes, the least significant bit of the address is not needed outside since it merely specifies which byte in the word is being requested. Therefore, the address bus is only 15 bits wide. The sixteenth bit (LSB) is maintained internally by the processor so that if a byte operation is performed, the unspecified byte will remain untouched.

The memory interface is designed to interface directly with standard RAM's and ROM's, and the data bus is not multiplexed to simplify system design. READY/WAIT control signals are available to slow down the processor to mate with any memory access time and allow the use of mixed memory speeds or slow memory (at 3 MHZ the processor requires 500 ns memories for full speed operation). HOLD/HOLD ACKNOWLEDGE is also provided to simplify the design of asynchronous DMA systems.

An example of a small memory system is shown in Figure 3. This system uses 1K words of ROM and 256 words of RAM. Note that READY is held high since the TMS 4700 and TMS 4042 memories are fast enough to run with the processor at full speed. All signals are of the right polarity and magnitude to interface directly with the memory system, and even the inverter shown on CS for the 4042 may not be required, depending on how the memory map is defined.

Context Switching

A context switch can be defined as a change in the program environment due to either an external (interrupt) or internal (subroutine call or XOP sequence) stimulus. This change can be effected by storing all

CPU BLOCK DIAGRAM

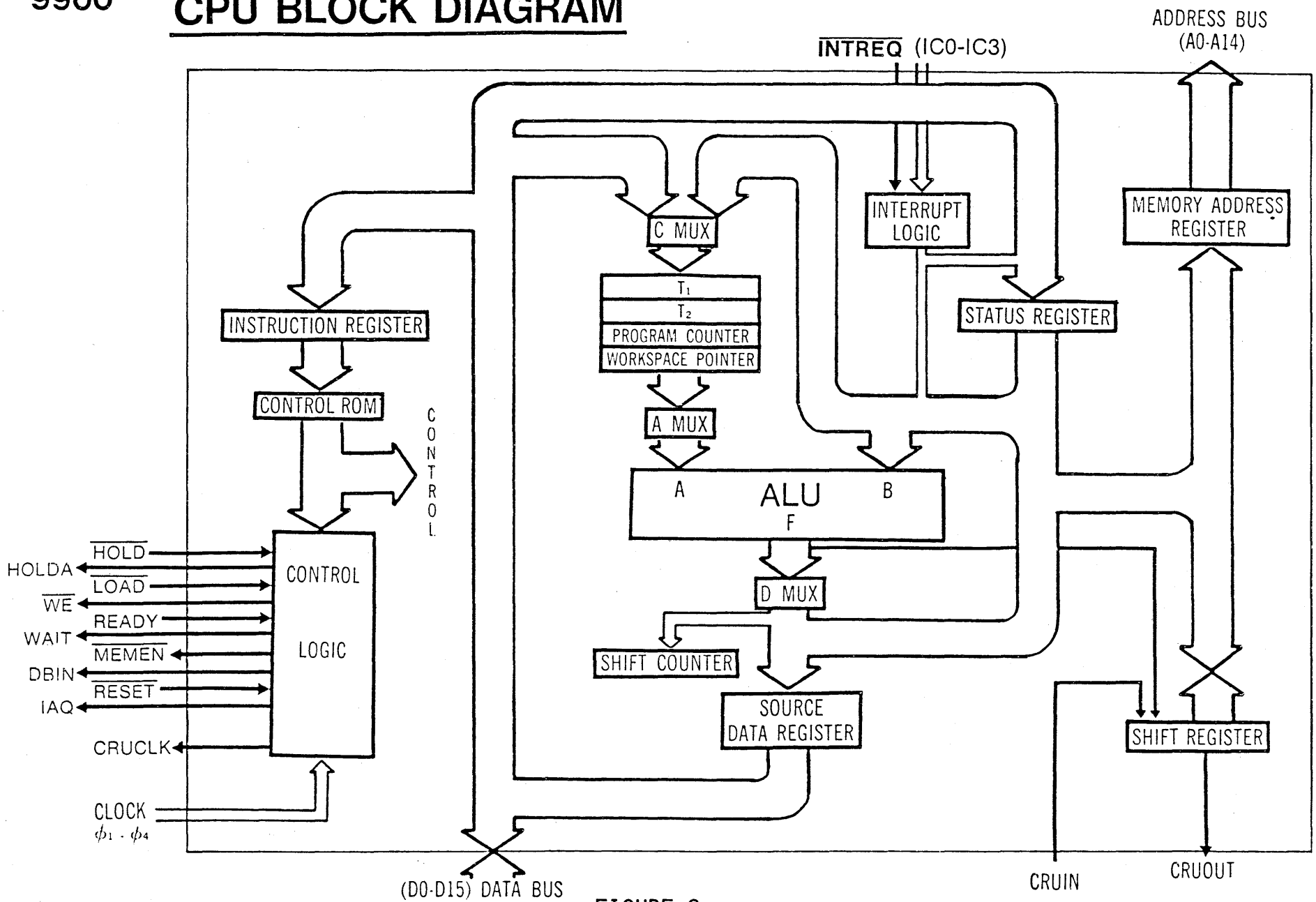


FIGURE 2

SMALL MEMORY SYSTEM

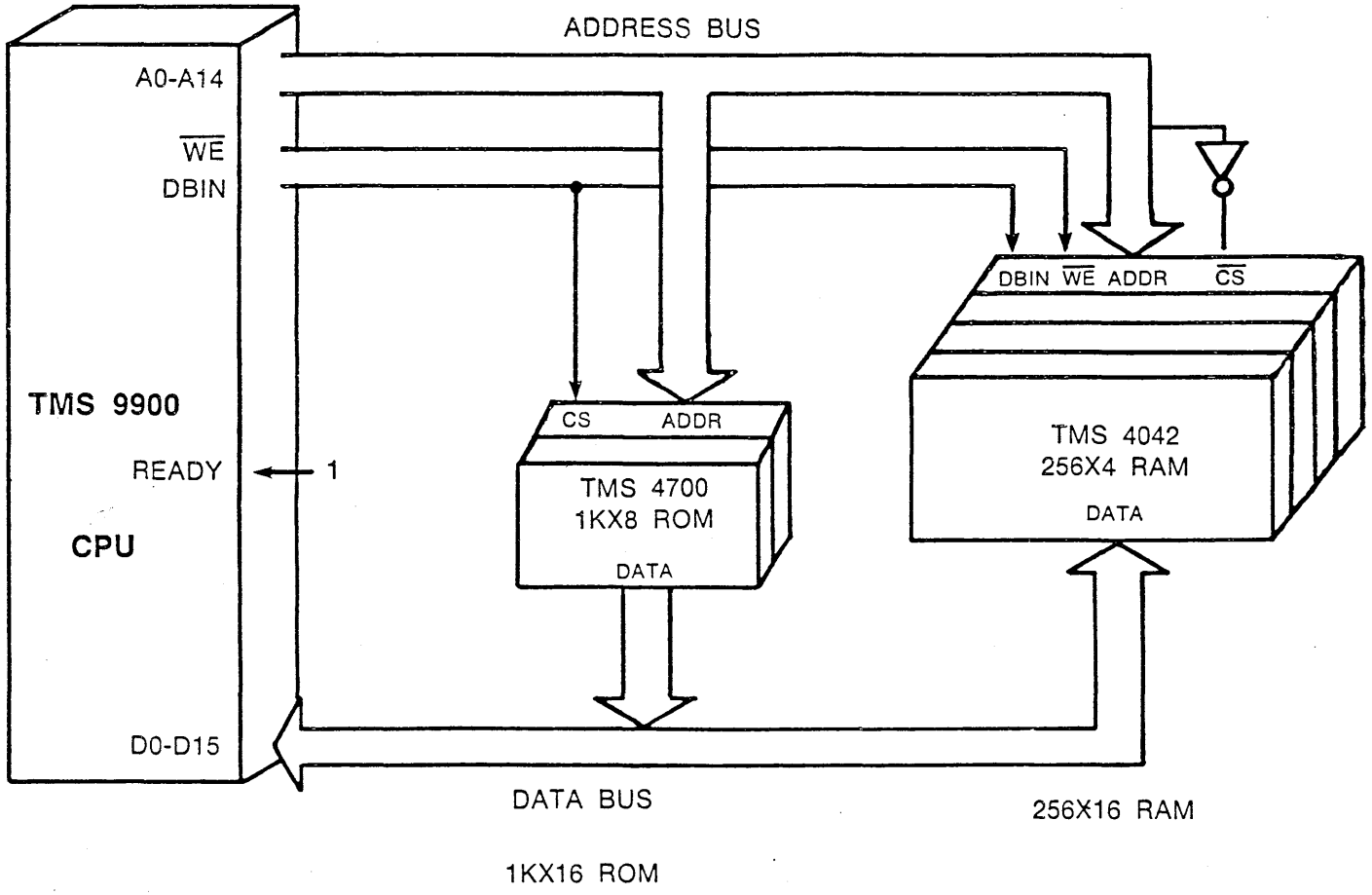


FIGURE 3

information contained in the processor relative to the active program before the change and replacing with any needed information for the new active program. The return will then take place when the stored information is reloaded into the processor. For example, to service an interrupt the TMS 9900 must store its 3 internal registers (PC, WP, STATUS) and load two (PC, WP). The return is then accomplished by reloading the three stored registers. The time required to perform this function is pure overhead and limits the interrupt service capability of the processor. Figure 4 shows the amount of time required for interrupt overhead for four current processors, and Figure 5 illustrates the effect of the required times. The bar charts represent the number of context changes per second that would completely saturate the processor. For example, the TMS 9900 saturates at 80K and the 8080 at 19K. The line at 15360 is representative of an application using the processor as a data concentrator for 16 9600 baud lines. The percentages represent the amount of processing time used as pure overhead. As the chart illustrates, the TMS 9900 offers a significant advantage in applications of this type.

Interrupt Structure

The TMS 9900 offers 17 prioritized interrupt levels, and the interrupt sequence includes an automatic register save and load of the PC and WP with vectors from memory. The two highest priority interrupts are the RESET pin and the LOAD pin, and neither are maskable. The remaining 15 interrupts are interfaced with 5 interrupt input pins (INTREQ, IC0-IC3) and are masked by a 4 bit code stored in the status register. The mask is a "fence" mask in that all interrupts above it are allowed, and all below are not. The mask is automatically changed whenever an interrupt is serviced such that only higher priority interrupts can be accepted. The interface is very simple (see Figure 6), and has been designed to require only standard TTL circuits.

I/O Interface

I/O data is transferred by the TMS 9900 serially using 3 dedicated lines: CRUIN is a serial input line, CRUOUT is a serial output line, and CRUCLK is an output synchronizing signal. The data is addressed by bit using 12 bits of the address bus. The I/O instructions can specify that any number of bits from 1 to 16 be transferred, and several data addressing modes can be used to minimize the software overhead involved. Parallel to parallel data transfers are very easily implemented using standard TTL components as shown in Figure 7 where an 8 bit input port and an 8 bit output port are implemented, and addressing by bit gives an automatic interface to single or odd bit numbers.

Instruction Set

Perhaps the most significant feature offered by the TMS 9900 is the instruction set. The instruction set is a complete minicomputer instruction set containing 69 16 bit instructions including multiply and divide. The instructions can specify two separate operands, each of which can be specified independently by several addressing modes. The processor allows 7 different addressing modes, and sub-groups of instructions are oriented toward words, bytes, and bits to maintain

INTERRUPT OVERHEAD

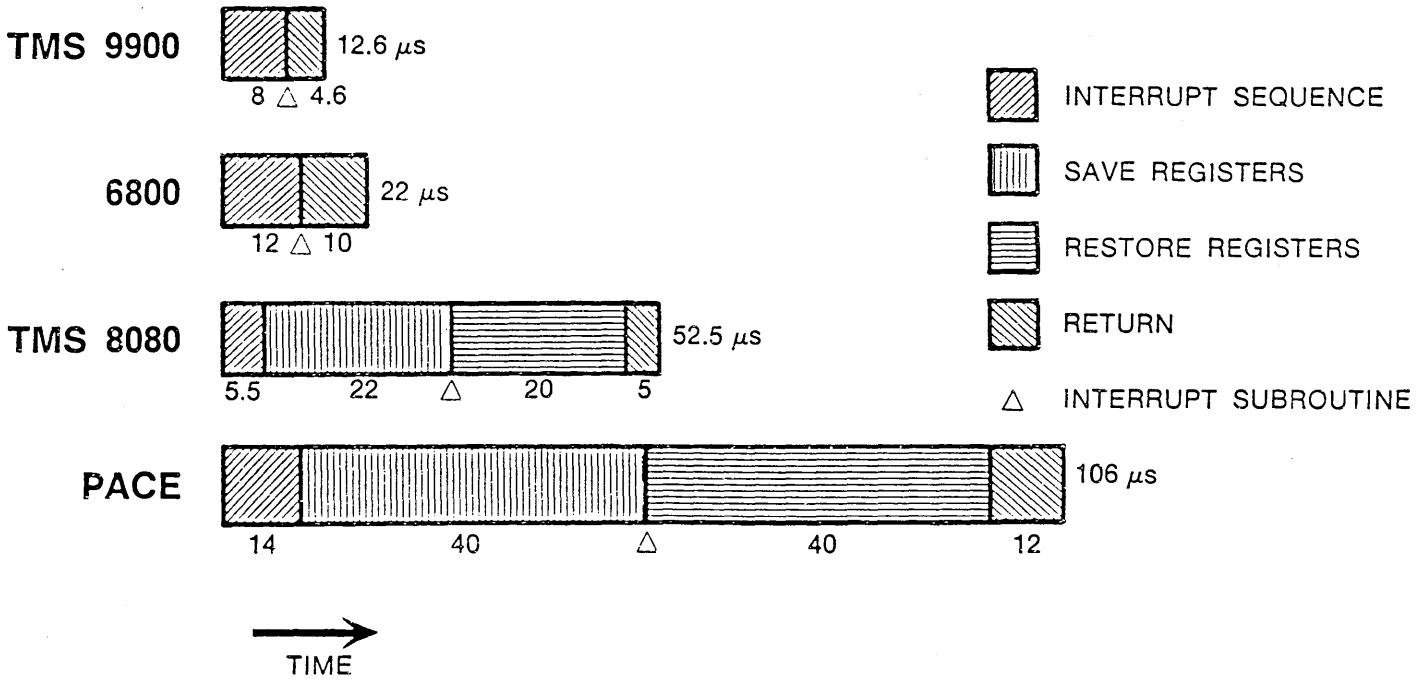


FIGURE 4

9900

SYSTEM CAPACITY
FOR CONTEXT SWITCHING OVERHEAD

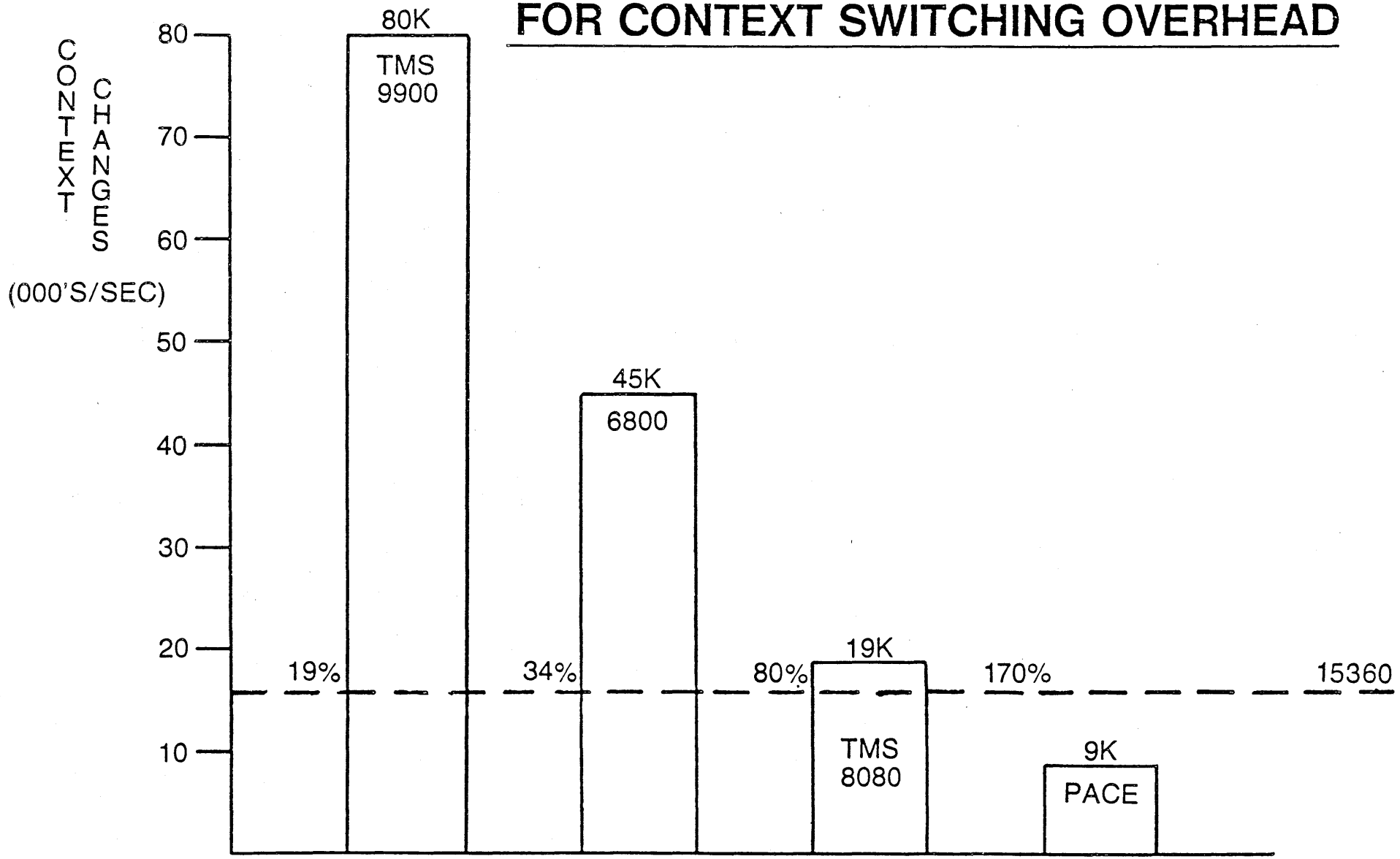


FIGURE 5

INTERRUPT STRUCTURE

- SEVENTEEN PRIORITIZED INTERRUPTS ARE AVAILABLE
- HIGHEST PRIORITY INTERRUPTS ARE RESET AND LOAD FOLLOWED BY 15 AVAILABLE FOR EXTERNAL USE
- EACH INTERRUPT HAS A TWO WORD VECTOR STORED IN MEMORY
- INTERRUPT INTERFACE:

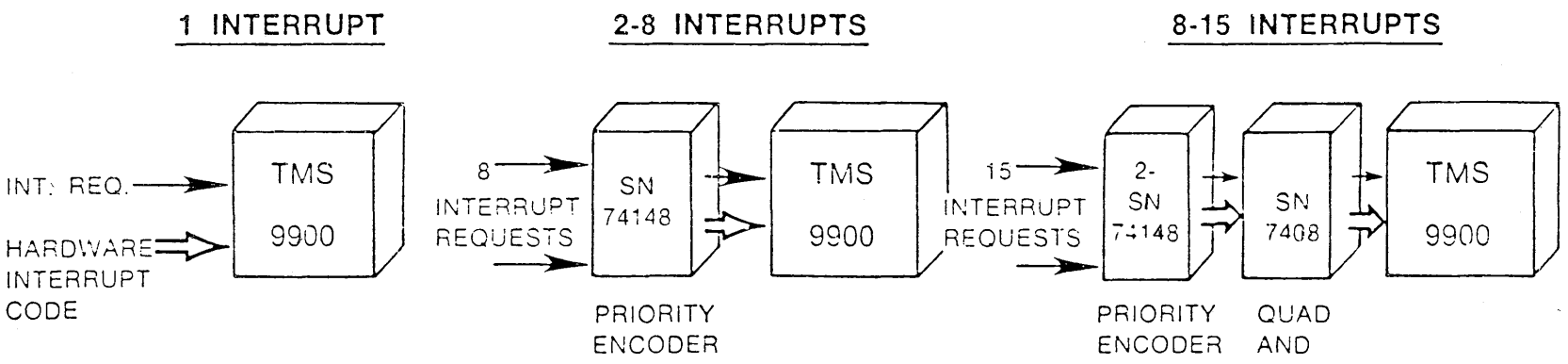


FIGURE 6

8 BIT I/O INTERFACE LOGIC

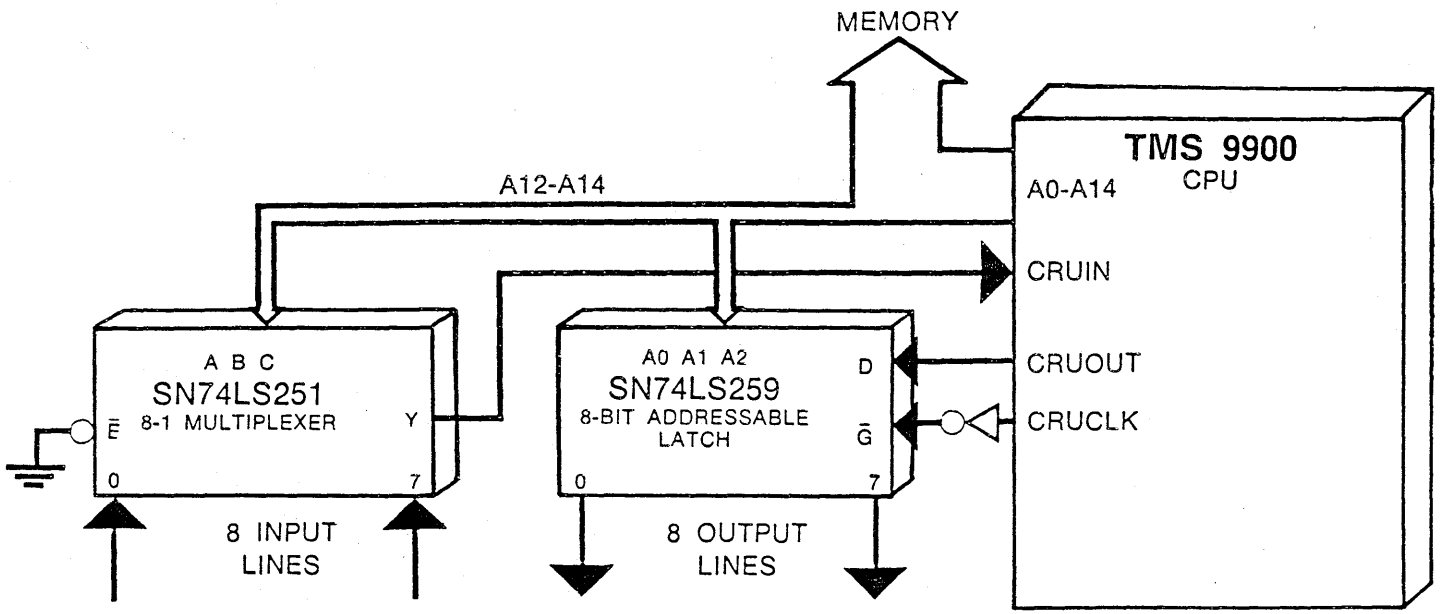


FIGURE 7

effectiveness in all applications.

A complete summary of the instructions is shown in Figure 8. The 16 instructions in the arithmetic sub-group include the instructions to add (for example) words, bytes, or immediate data, and to increment or decrement by 1 or 2, depending on whether one is operating on words or on bytes. The program control (20), data control (14) and logical (6) sub-groups are similarly comprehensive. The shifts (4) allow left and right, arithmetic circular and logical, and can specify shifts of from 1 to 16 locations. The I/O (5) instructions are used to communicate with the I/O interface, and the external (4) instructions do nothing except cause a unique code to be output to the outside world. The external instructions can be used to initiate external actions.

The seven addressing modes allowed by the TMS 9900 are:

1) Work Space Register
2) Indirect
3) Indirect with Auto Increment
4) Indexed
5) Symbolic (Direct)
6) Immediate
7) Program Counter Relative

Application of this instruction set has been shown to result in fewer instructions than comparable 8 bit architectures, and in fact to result in fewer memory bits for program storage. This in turn leads to a significant savings in design cycle cost due to:

1) Less programming time
2) Simplified debug
3) Lower hardware cost (fewer bits)

COST/PERFORMANCE ANALYSIS

Benchmark Evaluation

Figure 9 shows a comparison of the TMS 9900 with the PACE system, the 8080, and the 6800 for several different programs. The programs included are:

1) I/O Handler

   The processor accepts an interrupt, inputs a character, and checks to see if the character is an end-of-line character. If it is, the program jumps to another routine; if not, the character is output to a CRT for display and control is returned to the main program.

2) Character Search

   A forty character table is searched through for a specified character, and the address is returned. If the character is not found, a zero address is returned.

3) Computed Go To

   This routine tests a control byte which has one true bit. The

INSTRUCTION SET SUMMARY
69 INSTRUCTIONS

●ARITHMETIC (16)

ADD (W, B, Imm), SUB (W, B), COMPARE (W, B, Imm), INCR (1, 2), DECR (1, 2), ABS, NEG, MPY, DIV

●PROGRAM CONTROL (20)

BRANCH (LINK, LOAD WP), JUMP, JUMP CONDITIONAL (12) RETURN, IDLE, EXECUTE, EXTENDED OPERATION

●DATA CONTROL (14)

MOVE (W, B) LOAD (Imm, WP, ST), STORE (ST, WP), SWAP BYTES, CLR, SETO, SOC (W, B), SZC (W, B)

●LOGICAL (6)

AND I, OR I, INV, COC, CZC, XOR

●SHIFTS (4)

SRA, SRL, SRC, SLA

●I/O (5)

LDCR, STCR, TB, SBO, SBZ

●EXTERNAL (4)

RESET, CKON, CKOFF, LREX

FIGURE 8

BENCHMARK EVALUATION

	PROGRAM MEMORY REQUIREMENTS (BYTES)				ASSEMBLER STATEMENTS				EXECUTION TIME (μSEC)			
	9900	PACE	8080	6800	9900	PACE	8080	6800	9900	PACE	8080	6800
I/O HANDLER	32	44	34	23	9	18	17	10	76	150	73	54
CHARACTER SEARCH	22	24	19	18	8	10	12	8	655	1962	743	886
COMPUTED GO TO	12	12	17	18	5	5	11	9	99	352	155	162
$A_N - B_N = C_N$ (16)	20	26	36	41	6	10	21	20	513	1740	1155	1920
SHIFT RIGHT 5 BITS	10	8	22	21	3	3	15	10	22	58	167	91
$A_N - B_N = C_N$ (8)	20	26	30	32	6	10	15	14	513	1740	795	1380
MOVE BLOCK	14	18	16	15	4	9	9	8	7	27	19	19
TOTALS	130	158	174	168	41	65	100	79	1885	6029	3107	4512

FIGURE 9

position of the true bit determines which one of the 8 table vectors is used for control transfer.

4) Vector Addition

Two twenty-word vectors are added to produce a third twenty-word vector. Both eight and sixteen bit precision routines are shown.

5) Shift Right 5 Bits

A 16-bit word is shifted right by 5 places with zeros being shifted in.

6) Move Block

A block of N characters is moved to another location in memory. Both blocks can be anywhere in memory.

The comparison includes program memory requirements (in bytes), the number of assembler statements needed, and the execution times required to perform the various routines. Since all of the routines are independent, the results can be summed to give an indication of the processors' comparitive performances. As can be seen from the sums, the TMS 9900 offers significant advantages in all three areas.

Total System Costs

In order to analyze system costs, they have been broken down into five areas as shown in Figure 10 and compared with a typical 8-bit system.

1) Memory Costs

Memory costs relate directly to program memory requirements, and the benchmark analysis has shown that a 30% savings could be realized by using the TMS 9900.

2) Interrupt Interface

The TMS 9900 interrupt structure requires a minimal amount of standard TTL as opposed to complex or custom circuits.

3) I/O Interface

The dedicated I/O interface used by the TMS 9900 also allows the use of low-cost standard TTL.

4) Program Development

Program development can be related to the number of assembler statements required to perform the function. The benchmark analysis indicated that use of the TMS 9900 would allow significant savings in this area.

5) CPU and Support

The cost of the TMS 9900 chip itself will initially be more expensive due to the relative positions on the learning curve.

RELATIVE OVERALL SYSTEM COSTS

TMS 9900 AND 8-BIT MICROPROCESSORS

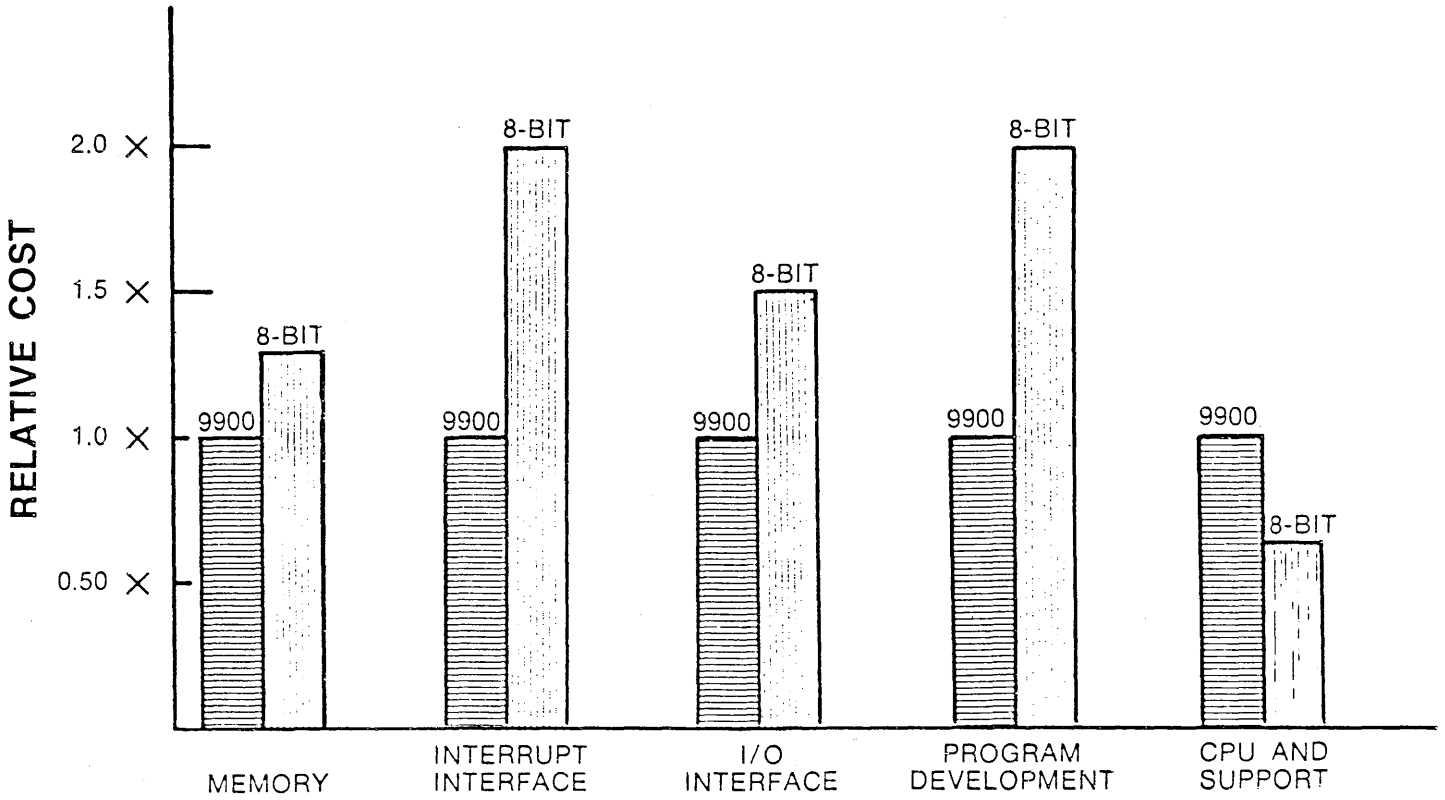


FIGURE 10

The five costs above can be summed to determine the total systems cost, and a comparison for three different system sizes is shown in Figure 11. As can be seen, the TMS 9900 is comparable for minimum system configurations and very quickly becomes cost effective as the system gets larger. In other words, the TMS 9900 system is the most cost effective solution to the majority of system applications.

SYSTEMS SUPPORT

In the area of systems support, the TMS 9900 is well supported both by hardware and software. The TMS 9900 has been designed to interface with standard memories and TTL, and thus, a huge number of inexpensive standard circuits are already available. In areas where custom circuits are needed (for example, the clock generator) development programs are underway to provide them. An assembler and simulator are now available on national timeshare services, and prototyping systems and stand-alone software development systems are under development. The high level languages, BASIC, COBOL and FORTRAN will all be supported. In addition, dedicated teams of systems and applications engineers are continually attempting to identify new areas and functions for which future support will be needed.

THE TI COMMITMENT

The TMS 9900 with its cost/performance ratio makes possible microprocessor based systems that were just not practical with the limited capability of 2nd generation 8-bit systems. TI feels that its future in microprocessors lies with the TMS 9900 system and will continue to provide the level of support needed to maintain its cost/performance leadership position.

RELATIVE SYSTEM KIT COST COMPARISON

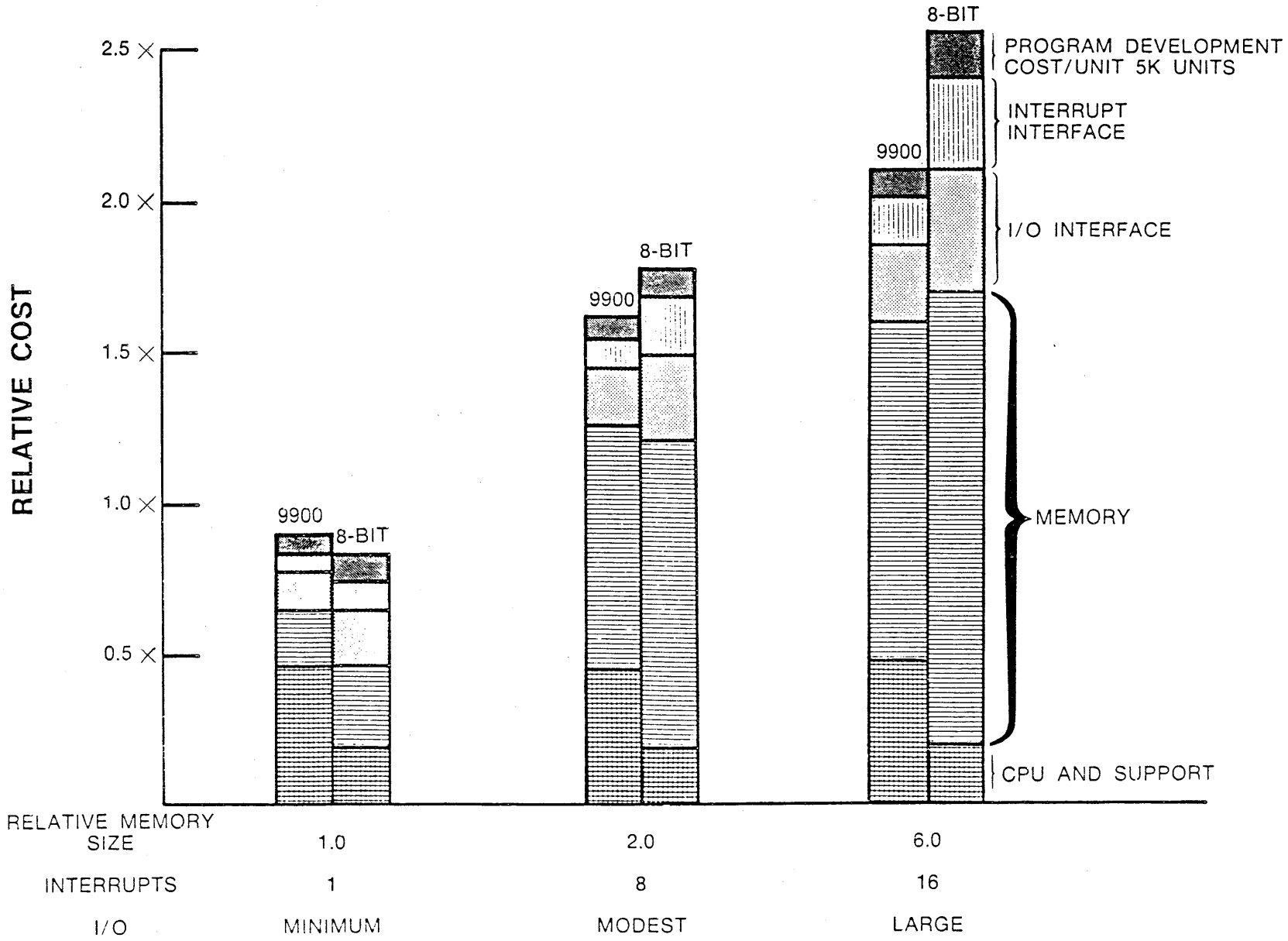


FIGURE 11

4. A MICROPROCESSOR DESIGNED WITH THE USER IN MIND January 22, 1976

WILLIAM E. WICKES
Manager, Microprocessor Developments
Electronic Arrays, Inc.
Mountain View, California

Electronic Arrays is in development with the EA9002 MPU, a high speed 8-bit parallel microprocessor designed primarily as a utility real time controller and data processor. It has been carefully conceived and engineered to fill a gap in the marketplace for a stand-alone, unbundled MPU not satisfied by the current proliferation of microprocessor products. The EA9002, as a result of full TTL compatibility, allows the user to select memory and supporting peripheral I/O components best suited to his individual application. Furthermore, the incorporation of a 64 byte (64 x 8) scratch pad RAM within the MPU will allow many system applications to be implemented without the need for external RAM components.

The EA9002 represents the first of what may be considered a new breed of speciality microprocessor products in that it is a stand-alone MPU with a clearly defined and unambiguous instruction set. This simplified instruction set along with an easily understood internal architecture and simplified approach to timing will accelerate product comprehension and software development time. Many of its outstanding features and attributes are depicted in the attached illustrations.

The MPU architecture, speed of execution and instruction set represents a subtle compromise between MOS/LSI logic, circuit and process technology. The internal 64 byte scratch memory is independent of external memory address and thus provides for rapid data manipulation. The eight 12-bit general purpose registers can be readily utilized as auxiliary accumulators, data pointers or indexing registers. They provide the capability of pointing to data anywhere within the range of the 12-bit address with one cycle time. The seven level 12 bit subroutine stack simplifies program control by providing automatic storage of current address location when servicing a subroutine. Internal flags, including an accumulator zero flag (A), are adequate for control of various modes of operation as well as bit testing. External control signals, including interrupt, have been provided to achieve maximum flexibility of selection of external components with minimum external discrete component requirements. The single phase clock input and +5V power supply further aid in reducing overall systems cost.

The 55 basic instructions decoded by the EA9002 MPU have been organized for ease of understanding while providing complete control capability. In addition to normal address and data control instructions, a hard wired packed BCD arithmetic mode with automatic decimal correction is included. This simplifies the implementation of decimal arithmetic algorithms often required in even the most minimal systems.

Minimum system configurations can be as simple as one EA9002 MPU, one EA 4700 1K x 8 ROM and two MSI devices for 8 bit parallel input output ports. Expanded versions can be readily achieved by adding more ROM, RAM and I/O devices. A unique wait/sync input output control pin allows the user not only to synchronize the MPU with slow peripheral devices and suspend operation, but also provides a synchronizing pulse at the end of each instruction execution cycle. A further example of a minimum system configuration is illustrated utilizing Motorola's MC6820 PIA. This is by no means the limit to compatible smart peripherals. The EA9002 easily interfaces with many devices utilizing 8 bit TTL compatible bus structures. It is an interesting evolution in microprocessor

developments that now that the industry has achieved TTL compatibility in MOS products the user is able to mix and match those components from individual suppliers best suited to his need, whether it is that of a sophisticated communications interface or a simple data interface.

The instruction set utilizes all possible combinations within the 8 bit op code as illustrated in the op code map. There are no undefined codes. There are 14 two byte instructions where the second byte consists of either extended address information or an 8 bit literal with the one exception being the two byte DLY (Delay) instruction. All other instructions are defined by a single byte of instruction code. A generous distribution of instructions are allocated to operations utilizing one of the eight general purpose registers. An individual register is designated by the lower 3 bits of the code providing command and register identification with one byte. The instruction set and definition have been chosen with the logic engineer in mind as opposed to those systems that can only be understood and programmed by large computer programmers.

Relating individual instruction to the internal architecture of the MPU is straightforward. Data can be input or output over the 8 bit data bus to either the accumulator or one of the 8 general purpose registers. Arithmetic operations (binary or BCD) can be carried out between the accumulator and a designated general purpose register or a designated location in scratch.

Data within the accumulator can be rotated right or left, thru carry or no, for fast manipulation or test. Automatic jump on all zero or all 1 state within a designated general purpose register simplifies program generation and control.

Timing for each instruction execution is straightforward and consistent. Write the code and count the MPU cycle for a program execution without confusion or ambiguity. Every MPU cycle is the same, regardless of whether it is an internal operation or a data transfer. All peripherals are treated as memory locations, therefore, instructions and/or data can be obtained from any external device.

A very simple but typical programming example illustrates the advantages inherent in the EA9002 over competitive products. Assume the need is to implement a real time controller which receives a 16 byte string (could be any length up to 63) of data from an external source. As a simplification it is assumed the data is available for transfer to the scratch at the rate specified by the 9002 software. If not, the WAIT input could be used for synchronization or the interrupt could be used and this whole routine becomes part of an interrupt service routine. However, for this example, the DOS (Data Out Strobe) is used to acknowledge receipt of data and outputs the received data for confirmation and error checking if desired. Thus, each time the DOS occurs, the external circuits will input a new byte of data. This example may relate to the input of information from a high speed A/D converter for purposes of performing a subsequent mathematical routine on the data block. Maybe the need is to compute a standard deviation. Maybe the data is from sensors in a machine or plant control environment where an alarm condition or control condition is to be tested or used for subsequent operations.

Designating address bit 11 true as the I/O address (bit 11 false is the ROM address location) simplifies address formating, decoding and control. General purpose register 5 is chosen to control locations within scratch memory where the incoming data is to be stored. General purpose register 5 is also chosen as the string length counter (in this case it is 16) as well. This allows one register to serve two purposes. General purpose register 4 is used to store the address of the I/O port from which data is to be received.

The sequence of instructions follows a normal pattern of events. That is, the accumulator is loaded with the page address of the I/O (in this case it is 08 Hex) which is then moved to page register 4. Next the lower 8 bits of register 4 is loaded with all 0's since the complete I/O address is 800 Hex. Register 5 is then loaded with the string length (10 Hex) which also serves as pointer to scratch memory where the data is to be stored. Initilization is now complete and the program is written to execute a loop consisting of fetching the data from I/O to accumulator, transferring data to scratch memory, generating a DOS strobe for acknowledgement, decrementing the string length and scratch memory pointer and, finally, jumping back to fetch the next byte of data.

It can be seen that this particular routine will allow the input of a byte of data every 10μs or a 100KHz byte transfer rate.

Comparing an input routine such as this with an Intel 8080 or Motorola 6800 quickly illustrates the superior speed and programming power of the EA9002.

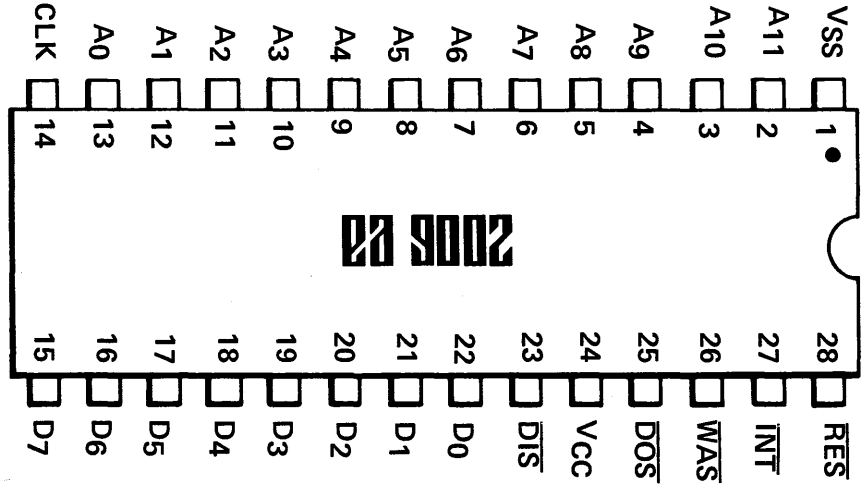
The EA9002 is schedule for sampling early this year with production by mid-year. The descriptive brochure is available with the Users Manual soon to follow. EA currently has excellent ROM and RAM supporting products and a programmable keyboard encoder circuit available. A macroassembler is on NCSS timesharing service and applications engineering is available to support systems designs.

A stand alone systems emulator "EASE" (Electronic Arrays Systems Emulator) is in development to be available in the 2nd quarter. "DEVELOP YOUR PROGRAMS WITH EASE."

It is not too early to review and consider this new microprocessor for product development programs. Review your old products for possible cost reductions. Review your new products for enhanced capabilities.

Introducing a New Microprocessor

ELECTRONIC ARRAYS, INC.



8 BIT PARALLEL MPU
WITH 512 BIT RAM

FIGURE 1

A New Microprocessor – Why?

EA Feels There is a need for a stand alone controller-type product which:

- Reduces systems costs
- Offers unbundled MPU
- Easy to understand and program
- Interfaces with readily available TTL components

Many real time controllers do not need computer type sophistication.

FIGURE 2

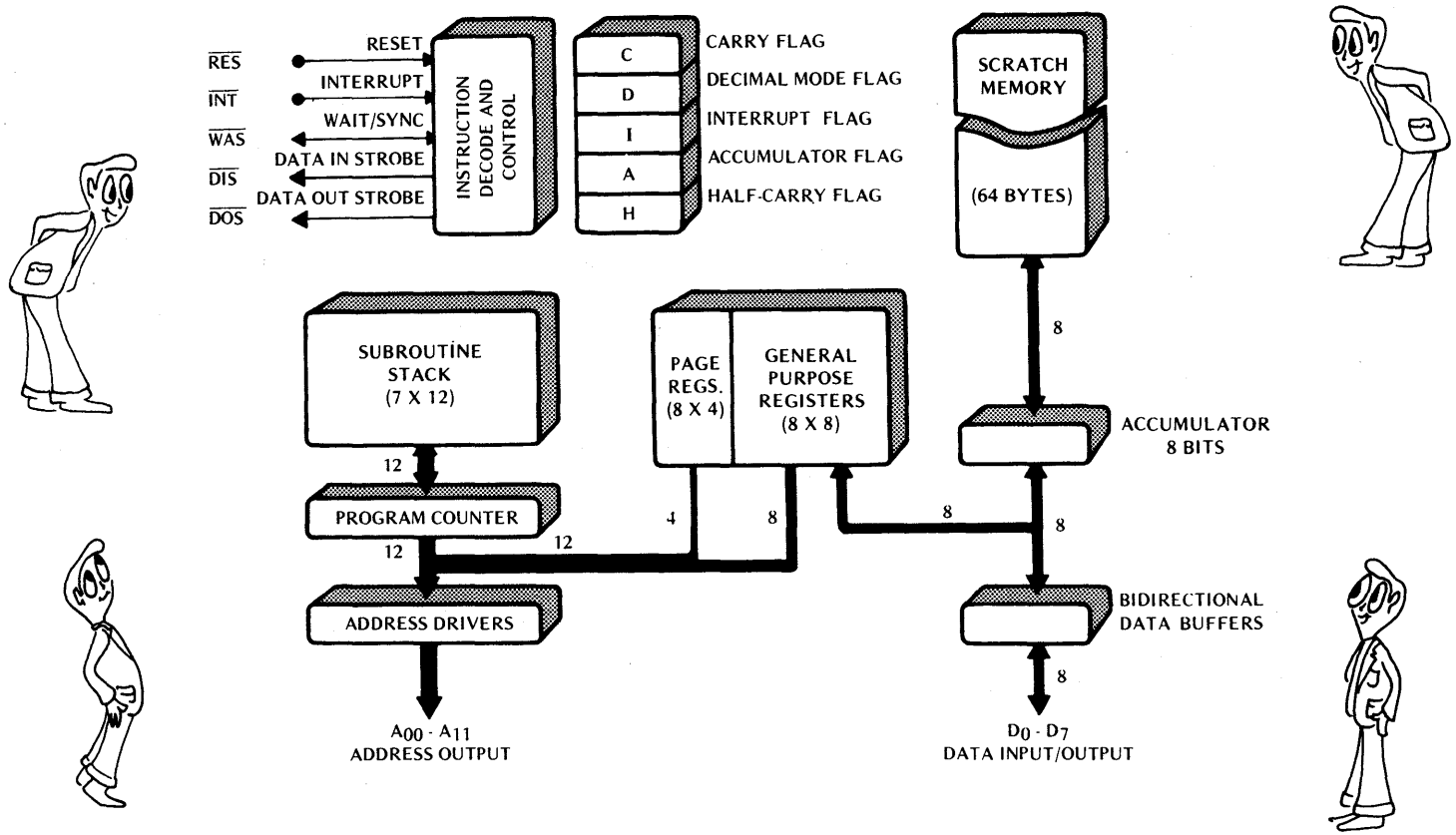
How Does This New Product Reduce Costs ?

1 Chip 8-Bit Fast Parallel Processor
On-Board 64 Byte RAM
Single +5V Power Supply
28 Pin Package
User Can Select Lowest Cost ROM, RAM, and Smart Peripherals
Minimizes Software Development Time

FIGURE 3

OK - What Is It ?

A 1 Chip 8 Bit Parallel MPU with 64 Byte Scratch



FAST – COMPACT – EFFICIENT

FOR REAL TIME CONTROLLER & INSTRUMENTATION APPLICATIONS

FIGURE 4

Look At These Features

64 Byte (512 Bit) Internal RAM
8 Bit Parallel I/O Data Bus
12 Bit Parallel Address Bus
8 12-Bit General Purpose Registers
7 Level Internal Subroutine Stack
Interrupt Input
Wait Input Control
Instruction Complete Sync Output
Binary or BCD Arithmetic
2μs Instruction Execution Time
Simple Timing Requirements

FIGURE 5

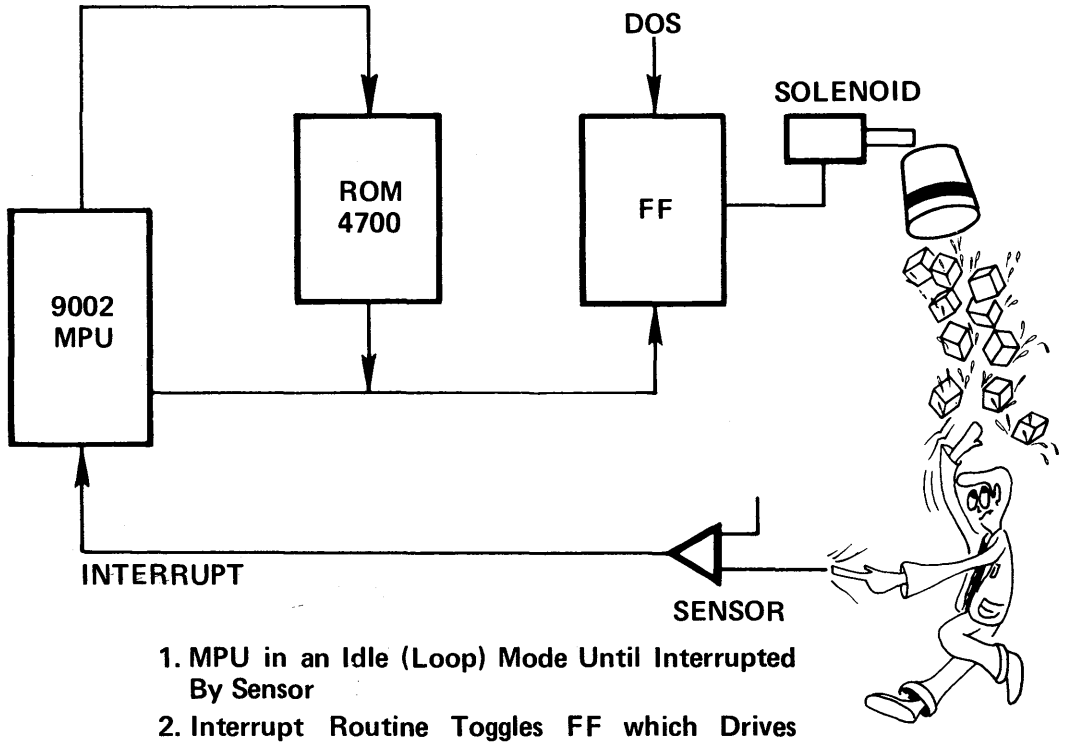
What About Instruction Power?

55 Basic Instructions
Add or Subtract in Binary or Packed BCD
8 GP Registers for Operand Addressing
or Data Handling
Input/Output Directly to Accumulator
or GP Registers
Scratch Memory Independent of External RAM
Jump Conditionally Anywhere Within a Page
Jump Unconditionally Anywhere
Wait on Slow Devices
Walk Across Page Boundaries

FIGURE 6

Excited !!

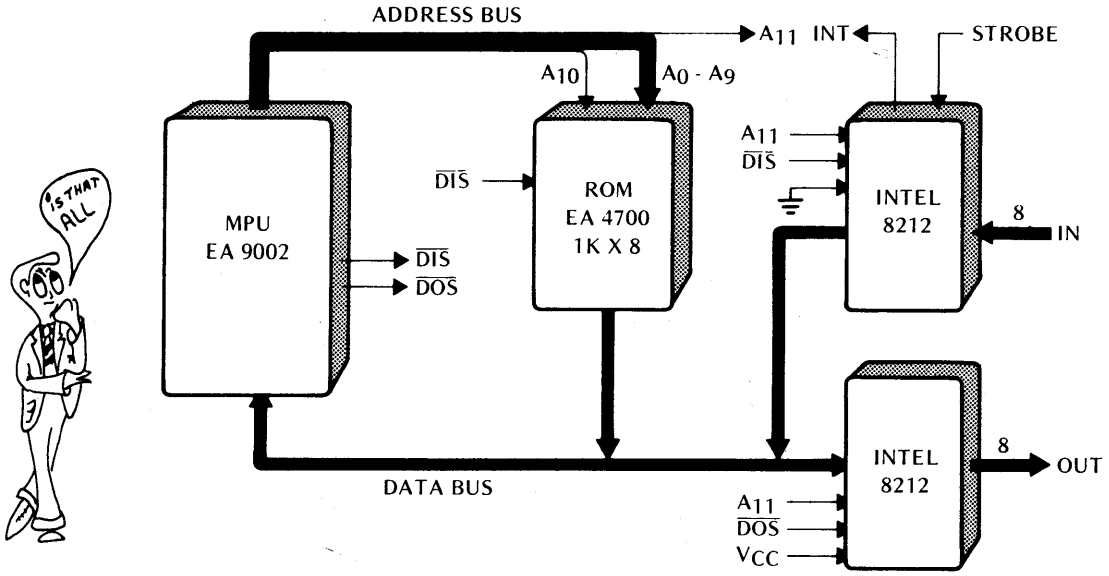
Then Sit Under This 9002 Controller & Get Cooled Off



1. MPU in an Idle (Loop) Mode Until Interrupted By Sensor
2. Interrupt Routine Toggles FF which Drives Solenoid and Dumps Ice on Excited Engineer
3. Routine Resets FF and Ice Bucket, Inhibiting Interrupt for 1 Minute
4. If Interrupt Still Present—Dumps More Ice

FIGURE 7

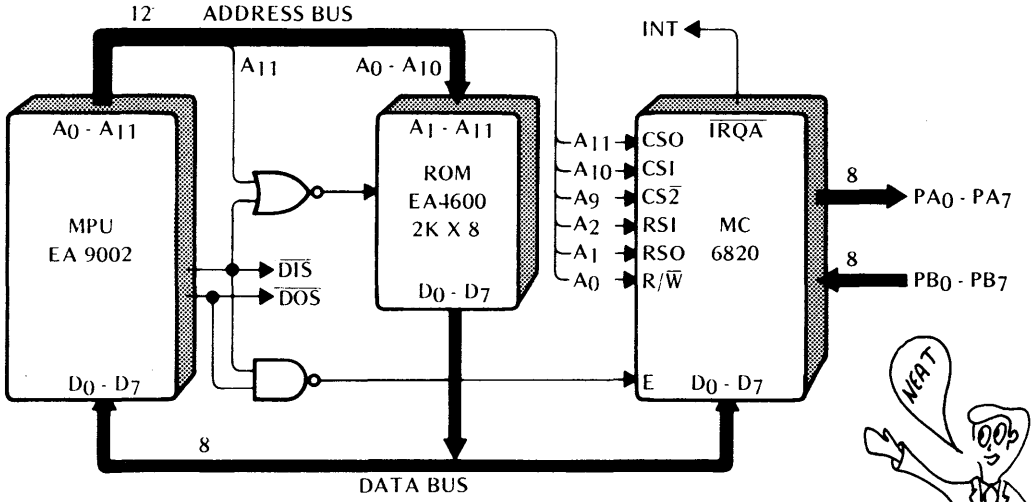
A Minimum System



UTILIZING INTEL MSI TTL DEVICES

FIGURE 8

A Minimum System



UTILIZING MOTOROLA'S PIA

USE WHATEVER PERIPHERAL IS MOST COST EFFECTIVE FOR YOUR APPLICATION

FIGURE 9

Instruction Set

4 MSB \ 4 LSB	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	DLY*	J — — CONDITIONAL JUMPS*							CPA				CSA	LAI*	DSI	ENI
1	JUN* — UNCONDITIONALLY															
2	JSR* — TO SUBROUTINE															
3	IRJ* — INCREMENT								DRJ* — DECREMENT							
4	XCH — EXCHANGE								CAP — ACC TO PAGE							
5	INP — INPUT								OUT — OUT							
6	LRI* — LOAD IMMEDIATE								JIN — INDIRECT							
7	INR — INCREMENT								DCR — DECREMENT							
8	ADD — ADD								SUB — SUBTRACT							
9	AND — AND								IØR — INC. OR							
A	XØR — EX. OR								CMP — COMPARE							
B	CAR — ACC → REG.								CRA — REG → ACC							
C	ADS — ADD SCRATCH								SUS — SUB SCRATCH							
D	RDS — READ SCRATCH								WRS — WRITE SCRATCH							
E	LRN — LOAD REG.								SRN — STORE REG.							
F	CLC	SEC	CLB	CMC	IAC	DAC	CLA	CMA	RAL	RAR	RLC	RRC	SED	SEB	RET	NOP

*Two Byte Instructions

EASY TO LEARN & UNDERSTAND ALL POSSIBLE CODES UTILIZED

FIGURE 10

Think Logically

ALU-GP Registers

SUB	ADD
IØR	AND
CMP	XØR
CRA	CAR
CPA	CAP
	XCH

Address Control

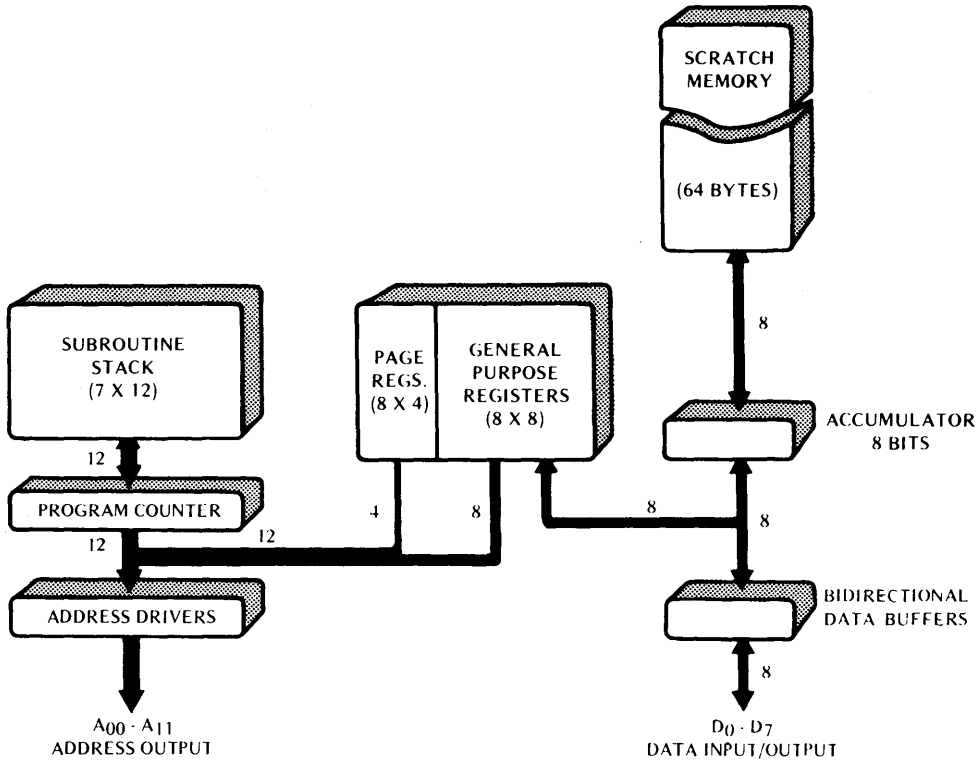
JVN*	J – –*
JSR*	RET
JIN	

GP Registers

INR	IRJ*
DCR	DRJ*

Miscellaneous

Set & Clear Flags
NOP
DLY



Input/Output

INP	LRN
OUT	SRN
LAI*	LRI*

ALU

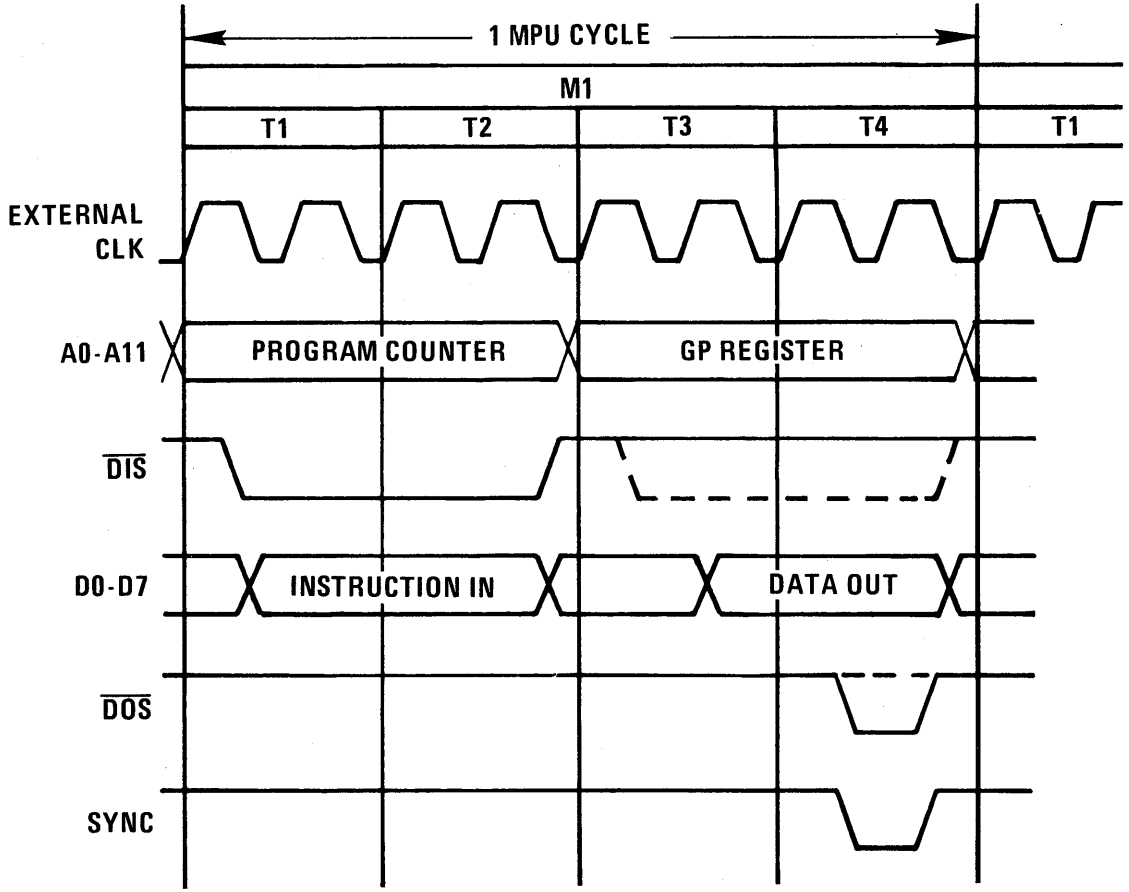
IAC	RAR
DAC	RRC
CMA	RAL
	RLC

ALU-Scratch

ADS	SUS
WRS	RDS

FIGURE 11

Timing

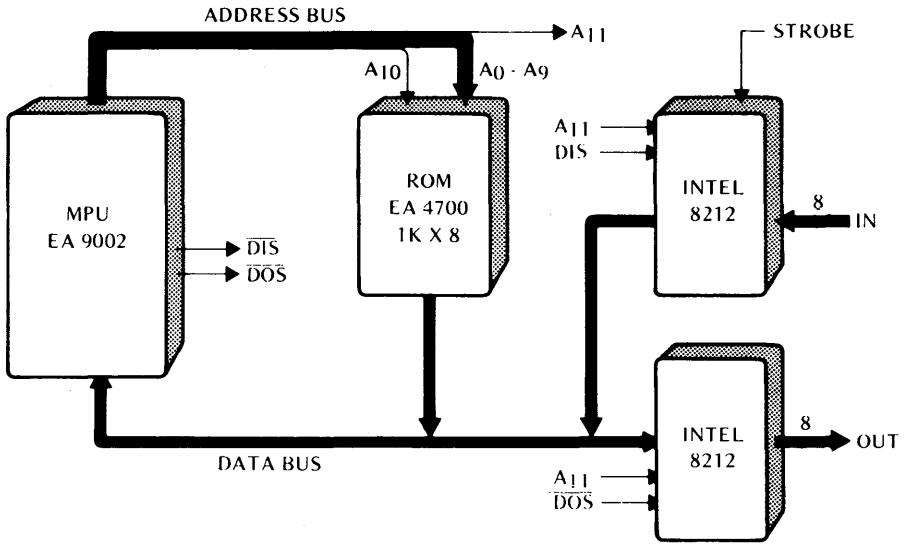


SIMPLE – CONSISTENT

FIGURE 12

Program Example –

Input a 16 byte string of data
Do not use interrupt
Use DOS (Data Out Strobe) to acknowledge
receipt and storage of data



SEQUENCE

1. Designate I/O Address as Bit 11 True (800 Hex) (Source of Data)
2. Designate Word String Length as 16
3. Designate Destination of Data as Scratch Memory Locations 1 to 16
4. Use Register 4 for I/O Address Pointer
5. Use Register 5 for Destination Address in Scratch Memory and String Length

FIGURE 13

Example Program Coding

OP CODE	INSTRUCTION			COMMENT
0 D 0 8	INPUT	LAI	PAGE	Load I/O page address
4 C		CAP	4	Load page reg 4 with I/O address
6 4 0 0		LRI	4, SOURCE	Load reg 4 with rest of I/O address
6 5 1 0		LRI	5, DEST	Load reg 5 with starting address in SM (also string length)
5 4	FETCH	INP	4	Fetch data from I/O & place in ACC
D D		WRS	5	Write data from ACC to SM
5 C		OUT	4	Generate DOS for acknowledgement
3 D 8 7		DRJ	5, FETCH	Decrement reg 5 (SM loaction & string length) & fetch next byte
	Fall thru on completion			

LABELS

PAGE	EQU	08	I/O address-bit 11 true is 800 Hex
SOURCE	EQU	00	
DEST	EQU	10	SM location & string length (10 Hex is 16 decimal)

INPUT DATA RATE = 10μs

FIGURE 14

Microprocessor Comparison – Program Loop For Inputing a 16 Byte Data String

	μs	BYTES		CODE		COMMENT
INTEL 8080	5	2	FETCH	IN	SOURCE	Input to ACC from I/O
	3.5	1		MOV	M, A	Move A to memory
	2.5	1		INX	H, L	Memory pointer
	2.5	1		DCR	B	String length
	5	3		JNC	FETCH	Fetch next data
	18.5	8				
MOT 6800	4	3	FETCH	LDA	A	Input to ACC from I/O
	6	2		STA	A, X	Store A to memory indexed
	4	1		DEX		Pointer & string length
	4	2		BGT	FETCH	Fetch next data
	18	8				
EA 9002	2	1	FETCH	INP	4	Input to ACC from I/O
	2	1		WRS	4	Store in SM
	4	2		DRJ	5, FETCH	String length & pointer
	8	4				

TWICE AS FAST WITH HALF THE CODE!!!

FIGURE 15

EA Support

components:	MPU	EA 9002	8 Bit Parallel
	ROM	EA 4700	1K X 8 Static
	ROM	EA 4600	2K X 8 Static
	RAM	EA 2101	256 X 4 Static
	I/O	EA 2000	99 Key Keyboard Encoder
software:	Descriptive Brochure		
	Users Manual		
	Macro Assembler for IBM 360		
	Macro Assembler on NCSS Timeshare		
hardware:	Start-up Board		
	System Emulator		

APPLICATIONS ENGINEERING

FIGURE 16

5. INTRODUCING THE 32K READ ONLY MEMORY

MICHAEL R. MCCOY
Manager, Memory Product Development
Electronic Arrays, Inc.
Mountain View, California

The advent of the MOS Microprocessor has been an undeniable boon for the MOS ROM industry. Once relegated to character generators, code converters and laboratory curiosities MOS ROMs have now found their way into hundreds and possibly thousands of applications attached to Microprocessors. A recent limitation to the pervasiveness of MOS ROMs has been that the performance of state-of-the-art Microprocessors has outstripped that of available MOS ROMs causing the user to choose a higher cost or less desirable alternative or else slow his system down. Recognizing this situation, Electronic Arrays is developing a new family of MOS ROMs intended for compatibility with present day and near term future microprocessors.

The EA3200 is the first in a series of compatible ROM and PROM products to be announced by Electronic Arrays. Key functional features of the EA3200 are:

- 32, 768 Bits Organized
  4096 words by 8 bits
- 3 Programmable Chip Selects
- High or Complemented Low-Level Chip Enable Clock
- Three-State Outputs
- Optional Output Latches
- Input Latches on Chip
- Standard Supply Voltages +12, ± 5, GND
- Simple Timing

Performance and electrical specifications are equally attractive:

• Access Time	300ns
• Cycle Time	470ns
• Input One Level	2.2 volts
• Input Zero Level	0.8 volts
• Output One Level	2.4 volts
• Output Zero Level	0.4 volts
• Power Dissipation	480mw Active 10mw Standby

The block diagram of Figure 1 illustrates the straightforward design of the 3200.

All addresses and chip selects are latched into the circuit with the Chip Enable signal. The addresses are then buffered, decoded and buffered again to address and read out one word of data. The data is detected by a sense amplifier triggered by a delayed Chip Enable signal followed by three-state output buffers. Two options are apparent from the diagram; high or low-level-inverted Chip Enable and output latching.

The timing diagram of Figure 2 shows the simple timing relationships of the EA 3200. Address and Chip Select signals need be present only for a short duration following application of the Chip Enable signal. Data access occurs in less than 300ns after Chip Enable reaches a valid one level provided a valid select code exists on the Chip Select Inputs. The output returns to open circuit condition less than 100ns after the Chip Enable signal returns to logic zero. Note that each access of data requires a Chip Enable cycle because the Chip Enable signal and its inverse are used to initiate several internal clocks.

Should the user select the latched mode of operation, data will remain on the outputs up to 2 milliseconds after Chip Enable returns to logic zero. Data may be altered or the chip de-selected only through initiation of another Chip Enable cycle.

Selection of the low-level Chip Enable option has a minor effect on chip timing. Due to the delay of the buffer circuit both access and de-select time will be increased about 50ns. Note that selection of the low-level version inverts the logic sense of Chip Enable.

It is not by chance that signal names and timing are similar to popular 4K RAMs. It is believed that many users are familiar with 4K RAMs thus making application of the 3200 easier. In addition, one will find that the 3200 and 4K RAM operation are sufficiently similar to be able to implement RAM/ROM memory systems using common signal lines.

Keys to the EA3200's Performance

In order to achieve the density and performance required by the objective specification, substantial improvements in both fabrication and circuit techniques had to be developed. The process selected was an Advanced Metal-gate process recently developed and exploited by Electronic Arrays in the EA4600 and EA4700 static ROM products. The process uses a masked oxidation technique to allow closely spaced geometries with a minimum of surface irregularities thus maximizing yield. The process is particularly suitable for ROMs as no yield-reducing contacts are required in the array, such contacts being essential in silicon-gate processes.

In the circuit design of the EA3200, we borrowed heavily from circuit techniques developed for high performance 4K RAMs. Most of the circuitry internal to the device is dynamic ratioless. In modern circuits such as this, the rather elaborate clocking required is made completely invisible to the user. The only penalty paid is the 2ms requirement placed on maximum cycle time.

One of the key circuits in obtaining the 3200's fast access time is illustrated in Figure 3. In previous designs it was common to use a simple buffer amplifier to read out the signal level from the ROM core. In contrast, the EA3200 uses a sense-amplifier technique similar to 4K RAMs. Here, only a very small signal is required from the memory core before the output can be correctly determined.

When Should I Consider A ROM?

In most Microprocessor applications there are several alternative methods of program storage. MOS, due to its low cost and compatibility with popular microprocessors is the technology that is almost universally used today. Even if one restricts the field to MOS devices, there are three distinct alternatives; RAM, ROM and EPROM (Erasable, Programmable Read Only Memory).

The chart of Figure 4 compares these three alternatives for features of interest. MOS RAM on the surface appears to be a winner due to its many desirable features. Closer inspection however, reveals that it falls down in three potentially crucial areas; density, cost in large quantity and volatility. The fact that it must be reprogrammed when powered down is probably the most frequent reason for dropping MOSRAM as an alternative.

The MOS EPROM's one outstanding advantage when compared to RAM is non-volatility. Once programmed it need not be reprogrammed again for years unless one exposes the device to ultra-violet light. Its disadvantages are its relatively slow access time and high comparative cost in anything but small quantities.

MOS ROM's such as the EA3200 share the non-volatility advantage with EPROMs but have the added advantage of high density and very low cost in medium to large quantities. Figure 5 illustrates the rather dramatic differences in ROM and EPROM pricing in various quantities. Here published prices are compared for an 8K EPROM and EA's compatible 8K ROM. Note that the cost of the ROM gets astronomical below 25 units due to the relatively high ($1500) one-time masking charge. Above 25 units the situation changes rapidly with the ROM having a 4 to 1 cost advantage at 1000 units. The comparison would even more dramatically favor the ROM if larger, more cost effective ROM's were compared with the 8K EPROM (currently the lowest cost device of its type available).

Unfortunately for ROMs, unit cost is not the only criteria for device selection. During early production, when program changes are likely, it may be intolerable to wait 6 weeks for a program revision. In such cases a PROM is often selected even at a substantial cost sacrifice.

Solving System Problems With The EA3200

The prime objective of the 3200 development was to realize a device that can be used effectively in microprocessor systems. How well this goal was achieved can be easily seen in Figures 6 and 7. Here the 3200 is shown connected in the two most popular microprocessor systems. Note the near absence of interconnecting devices, only a single TTL inverter is needed in the 8080 system and direct connection is used in the 6800 system. In both cases the system can operate at maximum rate unimpeded by the ROM and program capacity can be expanded to 16K words without any addition to supporting circuitry. The output drive capability of the 3200 allows direct connection to the 8080 data bus which is not possible with many existing ROMs. In both systems the user will find that selection of low level chip enable and non-latching outputs is most desirable.

Figure 8 shows the EA1600 in an EA9002 system. The 1600 is a 16K version of the 3200 and was selected in order not to monopolize the 9002's 12 line address bus. A minor amount of additional circuitry can be added to the 9002 system to implement block addressing in which case multiple 3200 devices could be implemented in the system.

The EA3200 is certainly not limited to microprocessor applications, its high performance, high density and low cost make it ideally suited for many applications. Figure 9 illustrates an increasingly popular application-application program storage. Here the programs stored in ROM are packaged in Plug-in modules that the user can change at will. Several recent minicomputer announcements have featured compilers stored in ROMs resident in the system. Such applications appear to have high utility. The customer need no longer load his program into the device via paper tape, cassette or the like. On the supplier's end, there are many advantages. When a customer purchases a program, he gets hardware rather than a "soft" copy and expensively generated software would be much more difficult to copy and use without authorization.

Other potential applications of the 3200 are numerous. Its high capacity makes it ideal for high-resolution character generators and digital voice response systems. Its very low power dissipation (10mw standby) is sure to make the device a winner in applications where available power is limited. Its low cost per bit will allow its use in such cost sensitive areas as complex electronic games.

The 3200 is in final stages of development and will be available in the second quarter of 1976. Start planning now to use this highly effective device in your next system.

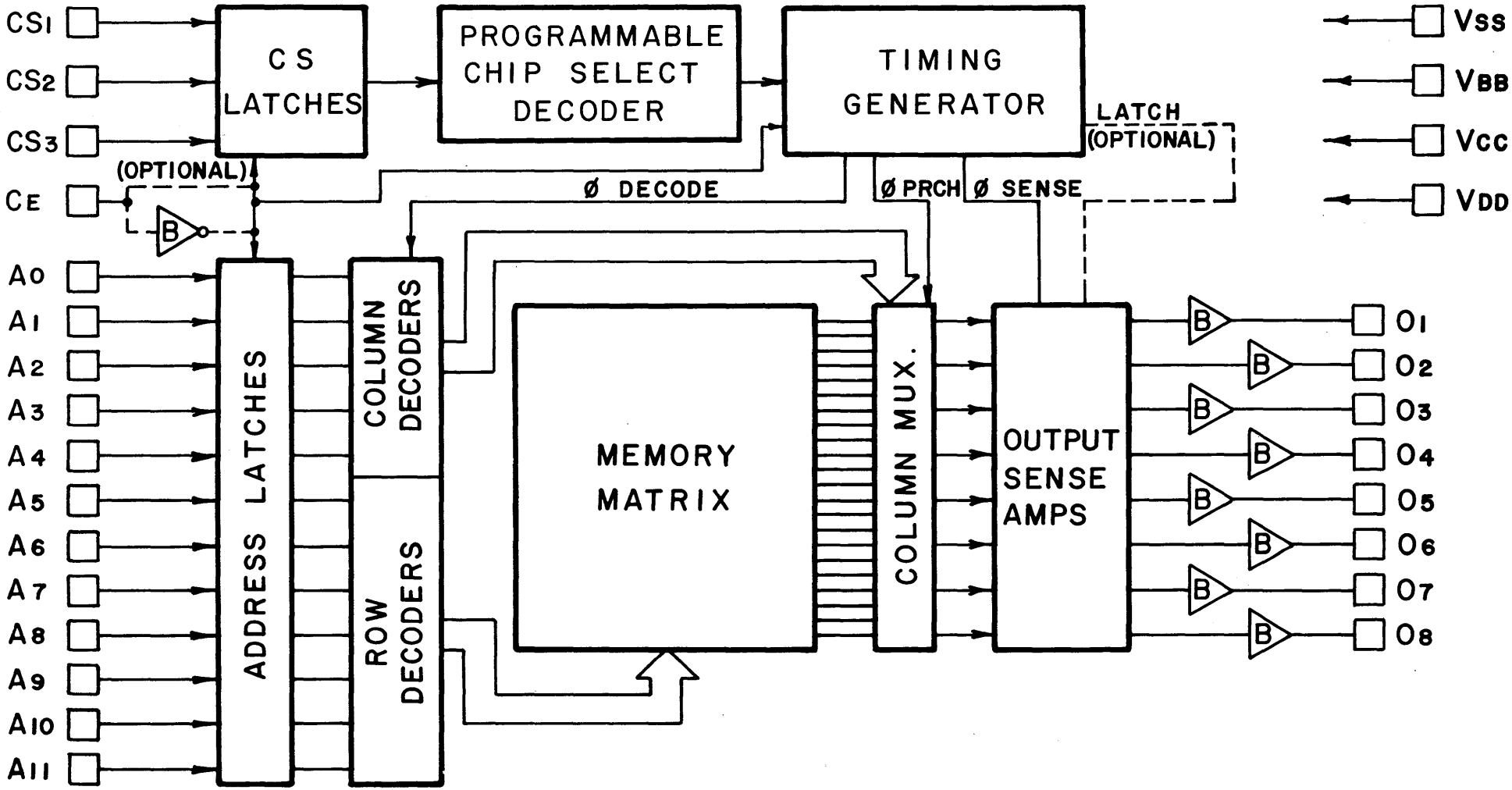
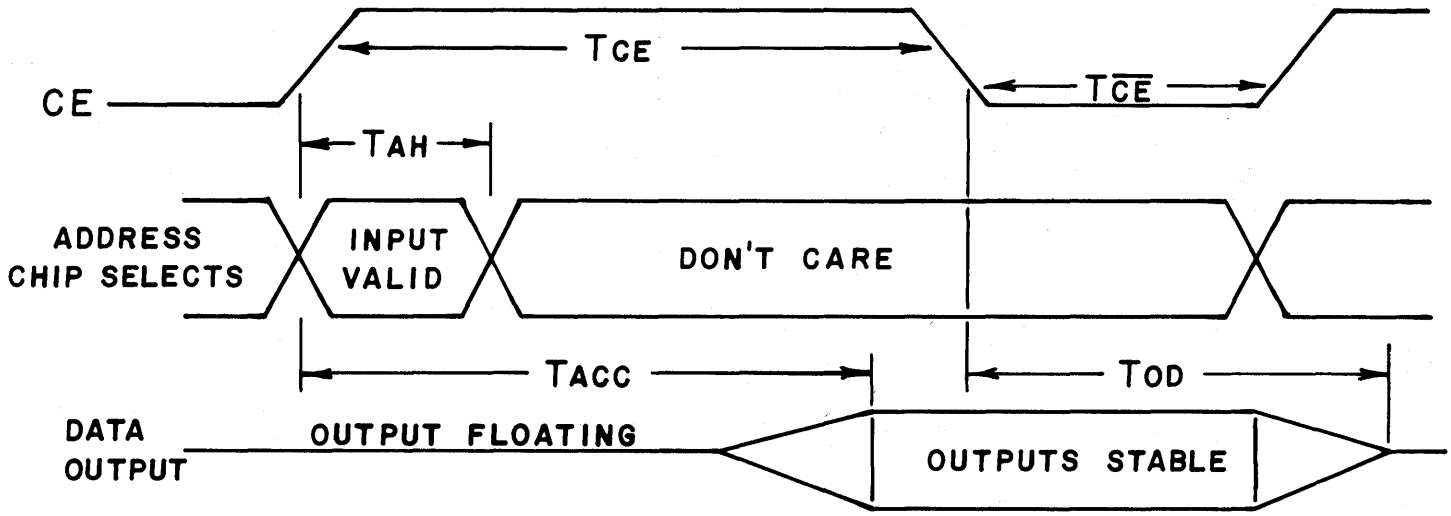


FIGURE 1 Block Diagram of the EA 3200



PARAMETER	SYM	MIN	MAX
ADDRESS HOLD TIME	TAH		90 ns
CHIP ENABLE ON TIME	TCE	280 ns	
CHIP ENABLE OFF TIME	$T\overline{CE}$	130 ns	
OUTPUT ACCESS TIME	TACC		300 ns
OUTPUT DISABLE TIME	TOD		100 ns

FIGURE 2 EA 3200 Timing

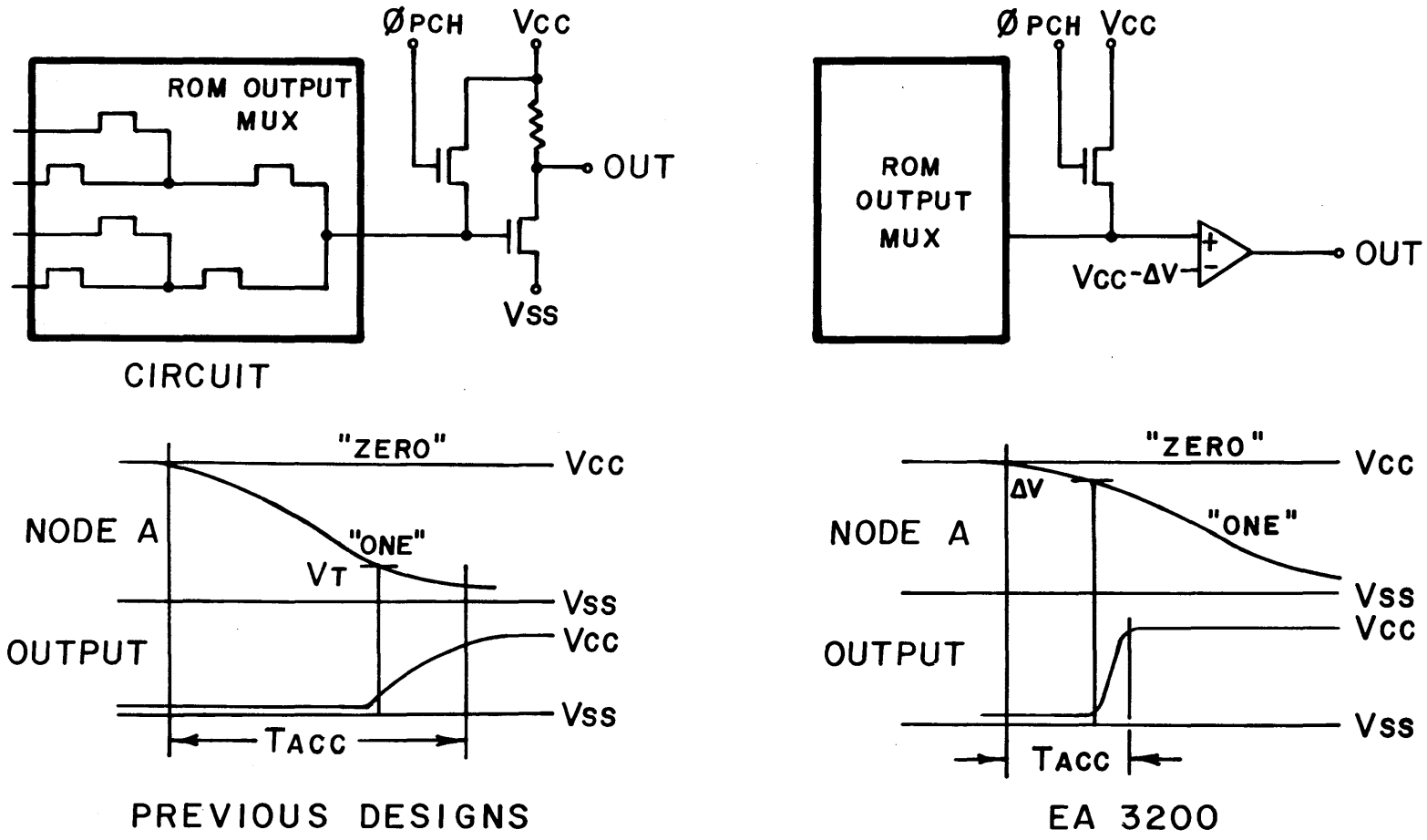


FIGURE 3

Comparison of ROM Sensing Techniques

FEATURE	MOS ROM	MOS EPROM	MOS RAM
COST-SMALL QUANTITY	HIGH	MEDIUM	LOW
COST-LARGE QUANTITY	VERY LOW	HIGH	MEDIUM
PROGRAMMABILITY	FACTORY ONLY	OFF LINE PROGRAMMER	IN SYSTEM
REPROGRAM TIME	6-8 WKS.	<1 HR	IMMEDIATE
VOLATILITY	NON-VOLATILE	NON-VOLATILE	MUST BE RELOADED AFTER POWER DOWN
DENSITY	32 K AVAIL. SOON	8 K	4 K
ACCESS TIME	300 ns	450 ns	250 ns

FIGURE 4 Comparison of Altenatives for Microprocessor Program Store

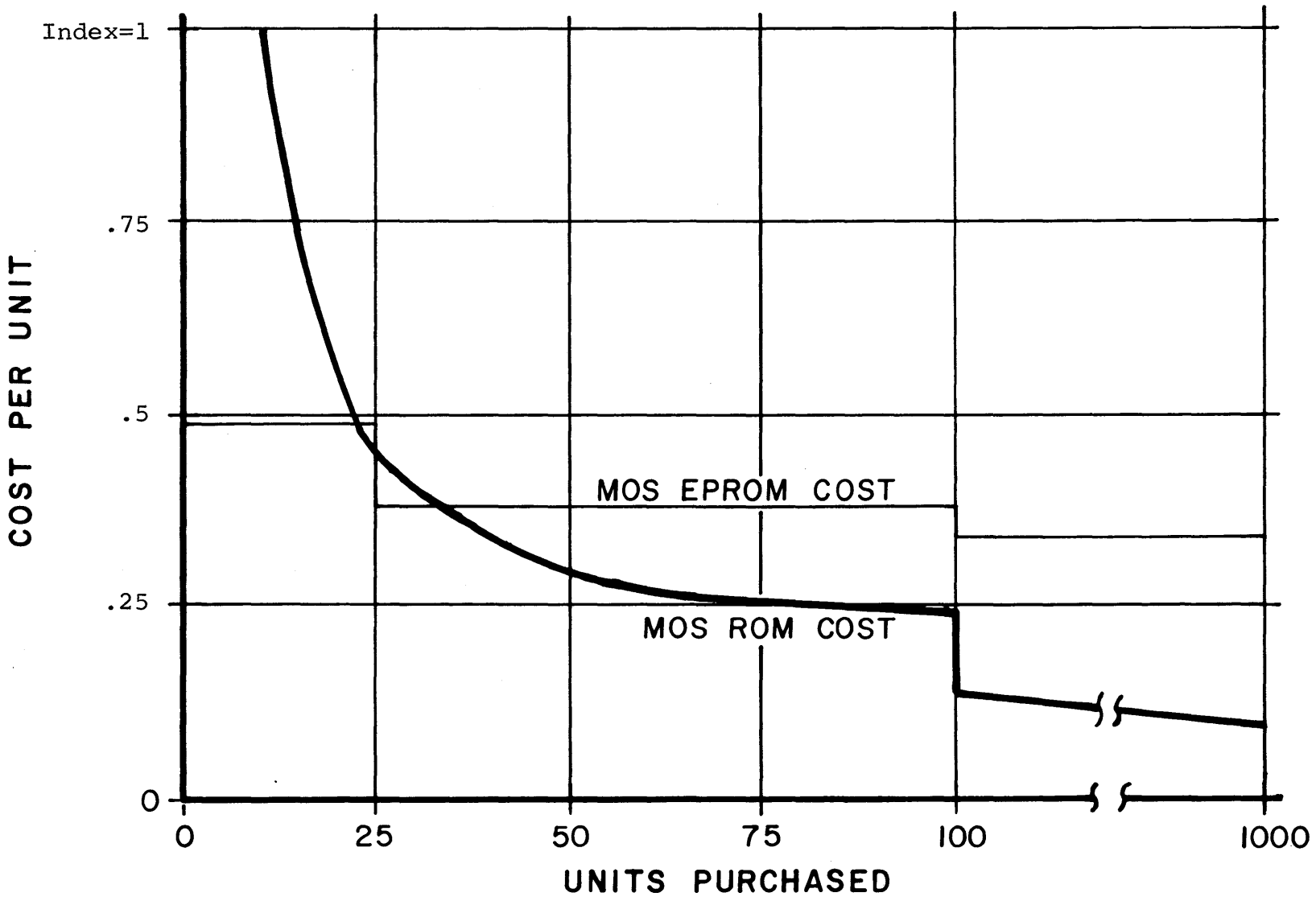


FIGURE 5 Unit Cost Comparison of 8K ROMs and 8K EPROMs

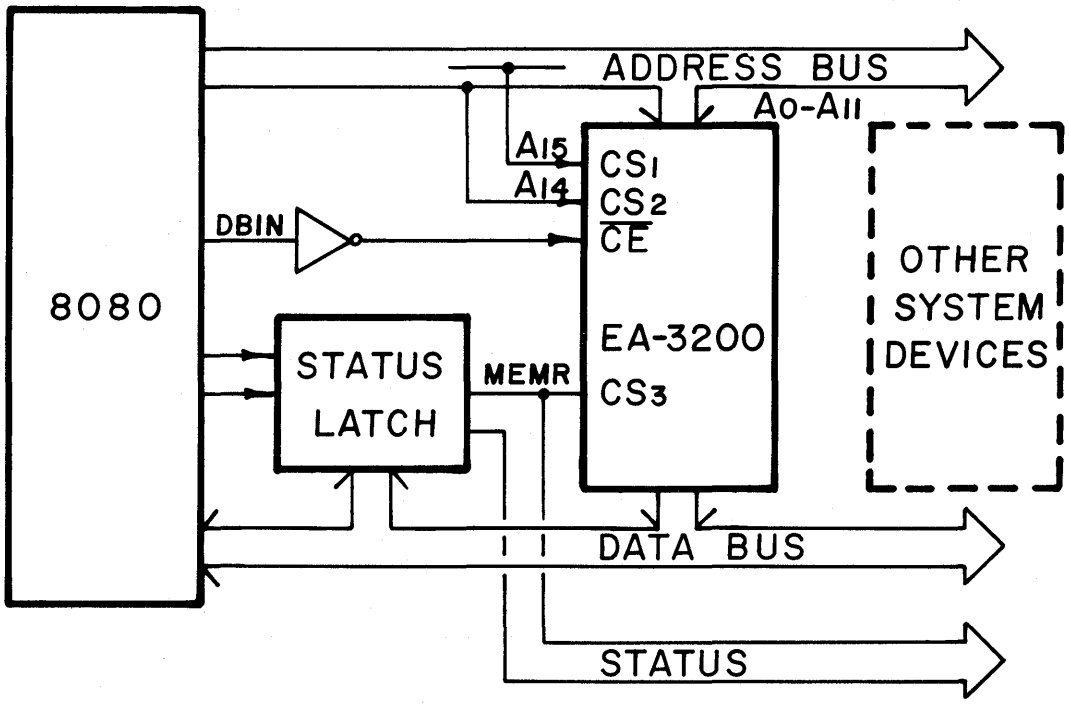


FIGURE 6

The EA 3200 in a 8080 Microprocessor System

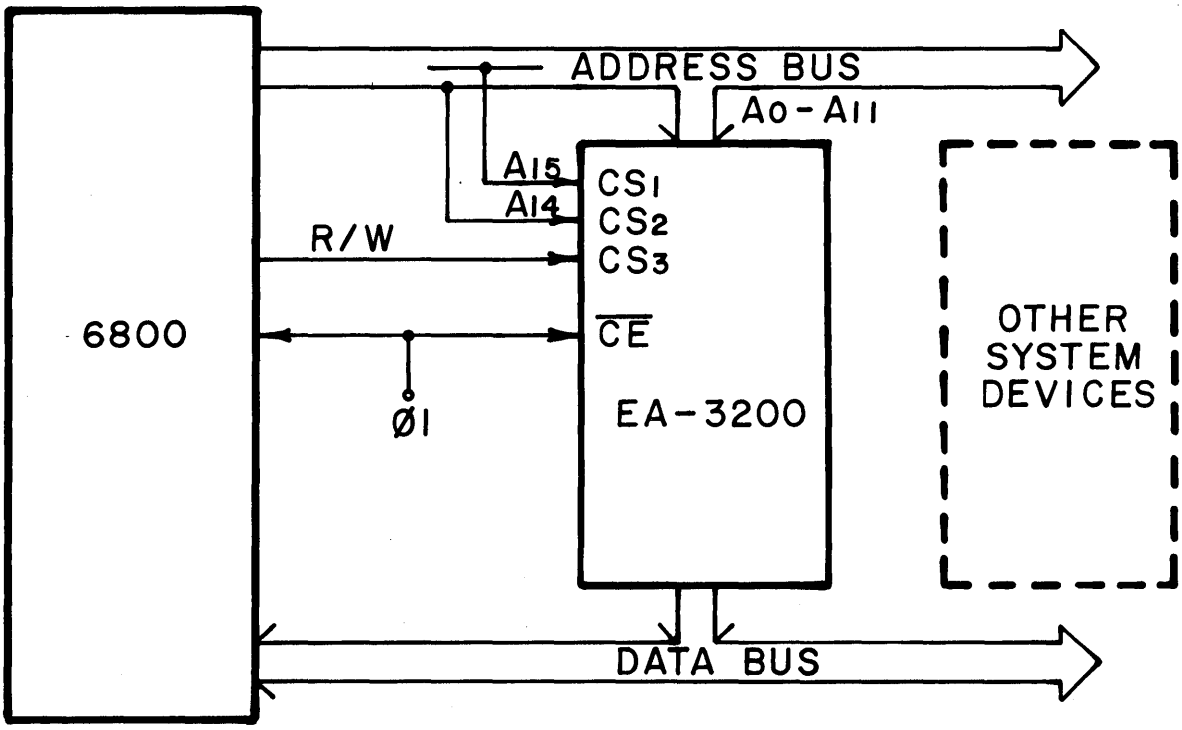


FIGURE 7 The EA 3200 in a 6800 Microprocessor System

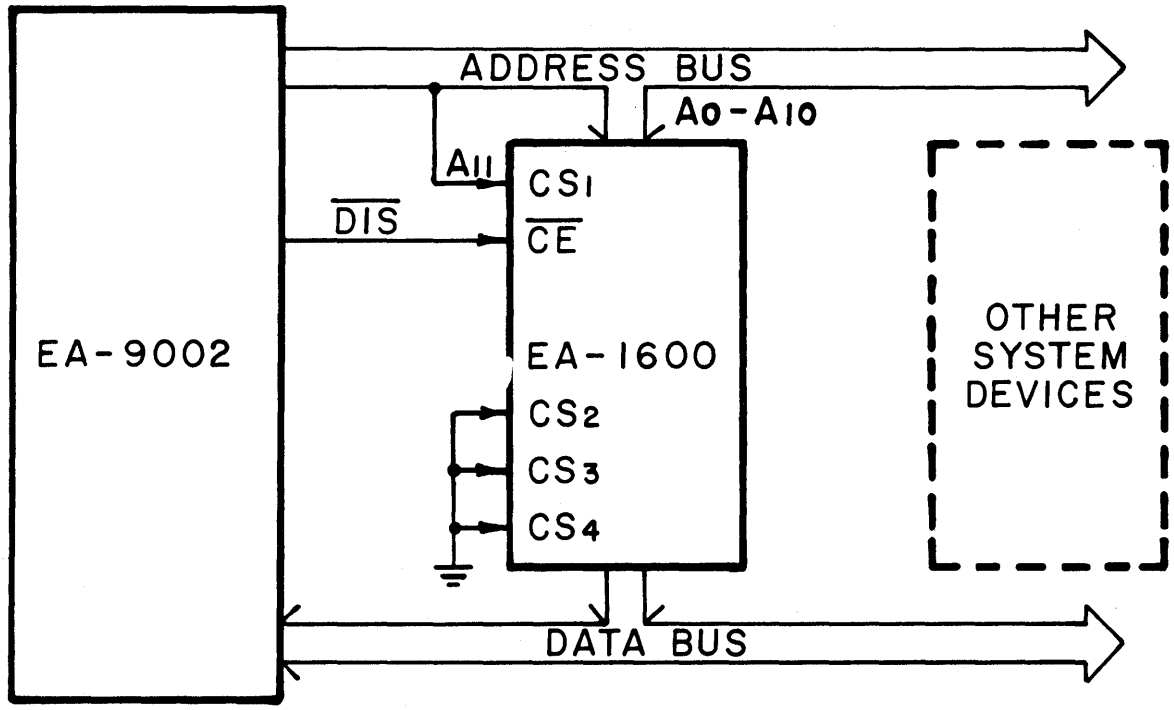


FIGURE 8 The EA 3200 in a 9002 Microprocessor System

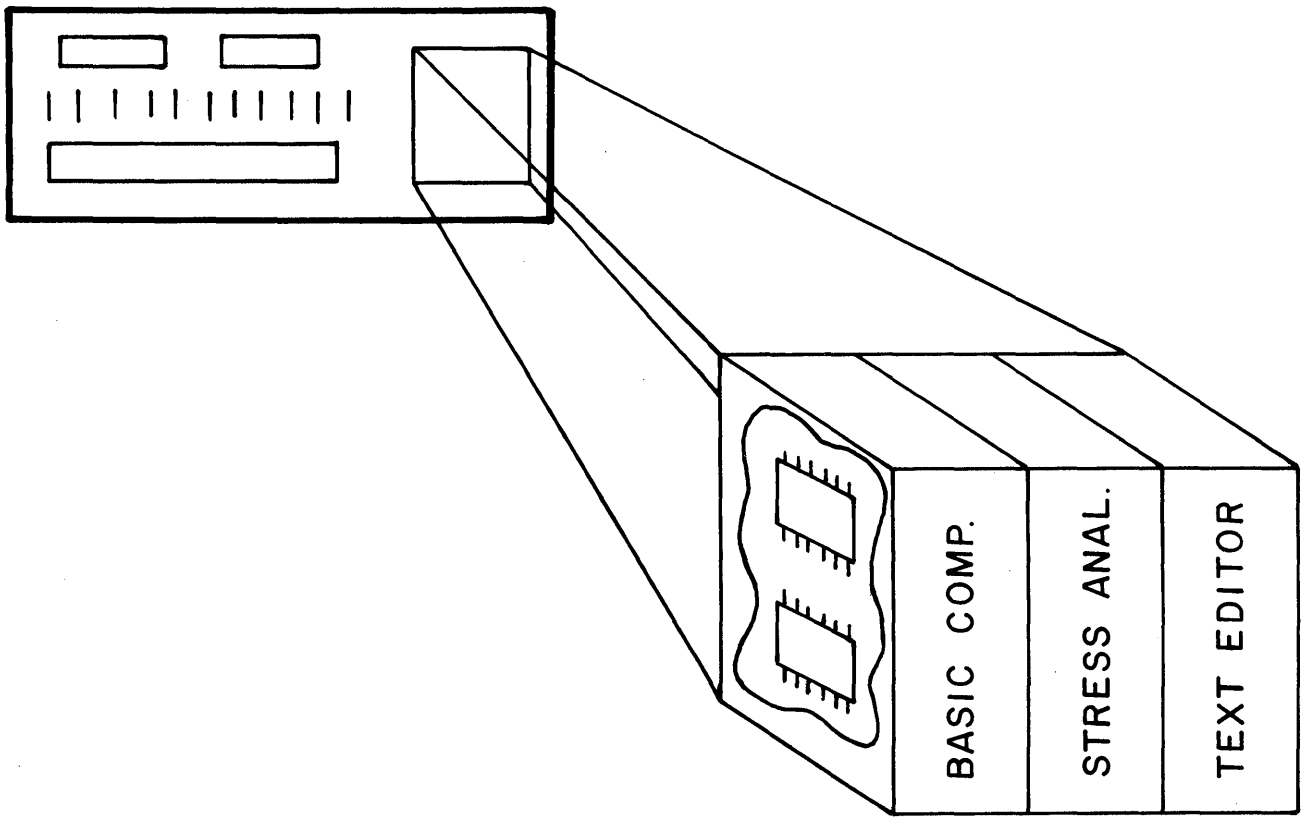


FIGURE 9 The 3200 Used for Plug-in Application Program Store

6. MICROPROCESSOR SOFTWARE DEVELOPMENT

DAVID LINDSAY
Microprocessor Design Engineer
Mostek Corporation
Carrollton, Texas

GENERAL

Microprocessors are today generating excitement in numerous fields. They are enabling performance and cost breakthroughs in many digital electronic applications, and making feasible a range of new products.

In principle, microprocessors achieve a simplification in space by expanding into the time domain. They replace a complex largely parallel circuit with a time-sequenced series of single operations (the program). With a possible sacrifice in speed, which may turn out to be unimportant or negligible, microprocessors thereby offer the user the advantages of:

1. Circuit simplicity through use of dense ICs.
2. Cost advantage of large-volume components.
3. Design flexibility achieved by program modification.

In all of this however, it should be observed that the development and design effort is largely subsumed by the program writer, who will therefore need equivalent techniques and tools to the hardware engineer's breadboard and oscilloscope. Neglect of this need could so severely hamper a project that it would not be an exaggeration for a user to select his microprocessor on the basis of the design aids available for it, rather than attempt to get reliable answers from the "black art" of benchmarking.

TECHNIQUES AND TOOLS

The primary tool of a programmer is of course the assembler, to translate from a mnemonic language (the source) to machine codes (the object). Use of a automatic aid for this is important, not merely to save the programmer's time on the translation process, but to spare him the frustration of checking for mis-translations when debugging. The microprocessor is quite capable of performing this assembly process itself when equipped with the appropriate peripherals for reading, punching, and listing, making the design system entirely self-contained.

Although a single teletype can provide all these functions, it is rather slow for extensive use. A source program can be edited most effectively if it is punched and read from cards, but depending on the peripherals available it may be preferable to leave it in memory and edit it there under the action of a text editor. The program may then be assembled immediately.

Debugging is a part of program development frequently misunderstood; it conveys an impression of hit-or-miss work, possible even carelessness or incompetence. In fact, the process deserves to be better analyzed and respected. The situation is that a programmer could be asked to write a program without need of debugging--that is, to execute correctly first time--and to have a reasonable chance of succeeding. He would however expend enormous effort routinely checking his program: emulating the microprocessor using extensive flowcharts, tabulations, and program proofs. This is largely unnecessary effort since the microprocessor itself is available for such emulation just as it was for assembly and editing. This is not of course recommending that programs be tried on the microprocessor without care or planning. It is recognizing that all thinking is a matter of conceiving, developing, examining, and modifying alternatives, and that used correctly the computer can become a vital extension to the programmer's brain in this process.

The microprocessor system used for assembling, editing, and debugging would typically be connected to the controls, displays, sensors and actuators of the user's application to develop both the program software and the interface hardware. Based on this, the user will be able to design his customized microprocessor system (the target) for the job, and be able to complete his product except for the custom ROMs to contain his program. Since there may have been significant modifications to the program and logic in changing to the target system, it would be reckless to order ROMs without first emulating them in the new system. For this purpose the user desires boards that provide pin-for-pin emulations of the ROMs, plugging into their sockets without requiring any modification to his system. And since this emulation would be pointless if it were always successful, the programmer still requires the ability to modify the program and monitor its execution, but again without requiring any modification to the target system that would render the emulation unrealistic. When this fine-tuning is accomplished, the user is ready to order ROMs and complete his product.

THE F8

Before proceeding to the description of a system designed for these requirements, a brief description of the microprocessor used is necessary. This is the Mostek F8, an 8-bit byte oriented processor, addressing 64K bytes of memory and operating with a 2MHz clock. See Fig. 1.

The F8 has a number of unusual features. Foremost among these is the complete absence of an address bus or address registers on the CPU. Since an address bus is largely occupied with sending out sequential addresses in a program, each memory chip is provided with address registers that it sequences instead; special control codes adjust

the registers for nonsequential values. This frees 16 pins on the CPU and each ROM chip, which are used to give the F8 a plethora of ports for easy I/O interfacing.

The F8 memory chips are also equipped with timers, for measuring delays without distracting the CPU from other processing, and interrupt circuitry, for automatically sending the CPU to a specified part of the program.

The F8 ROM chip is called a Programmable Storage Unit (PSU) since it contains, in addition to 1K bytes of ROM, the additional address registers, timer, and interrupt circuitry. These features require a much more elaborate emulation board than a conventional memory.

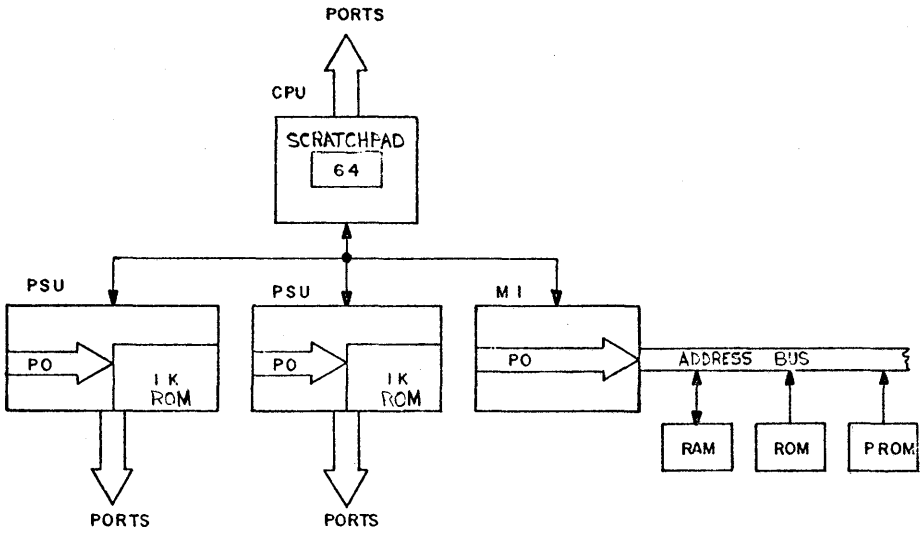


FIG 1

F8 RAM is implemented with conventional RAMs and a Memory Interface (MI) chip containing the address registers. However, a unique feature of the F8 is that its CPU contains 64 bytes of RAM, which can supply the entire RAM requirements for many applications or serve as the fast-access scratchpad in other cases. The development system must allow the programmer to display and update this scratchpad as well as the regular memory.

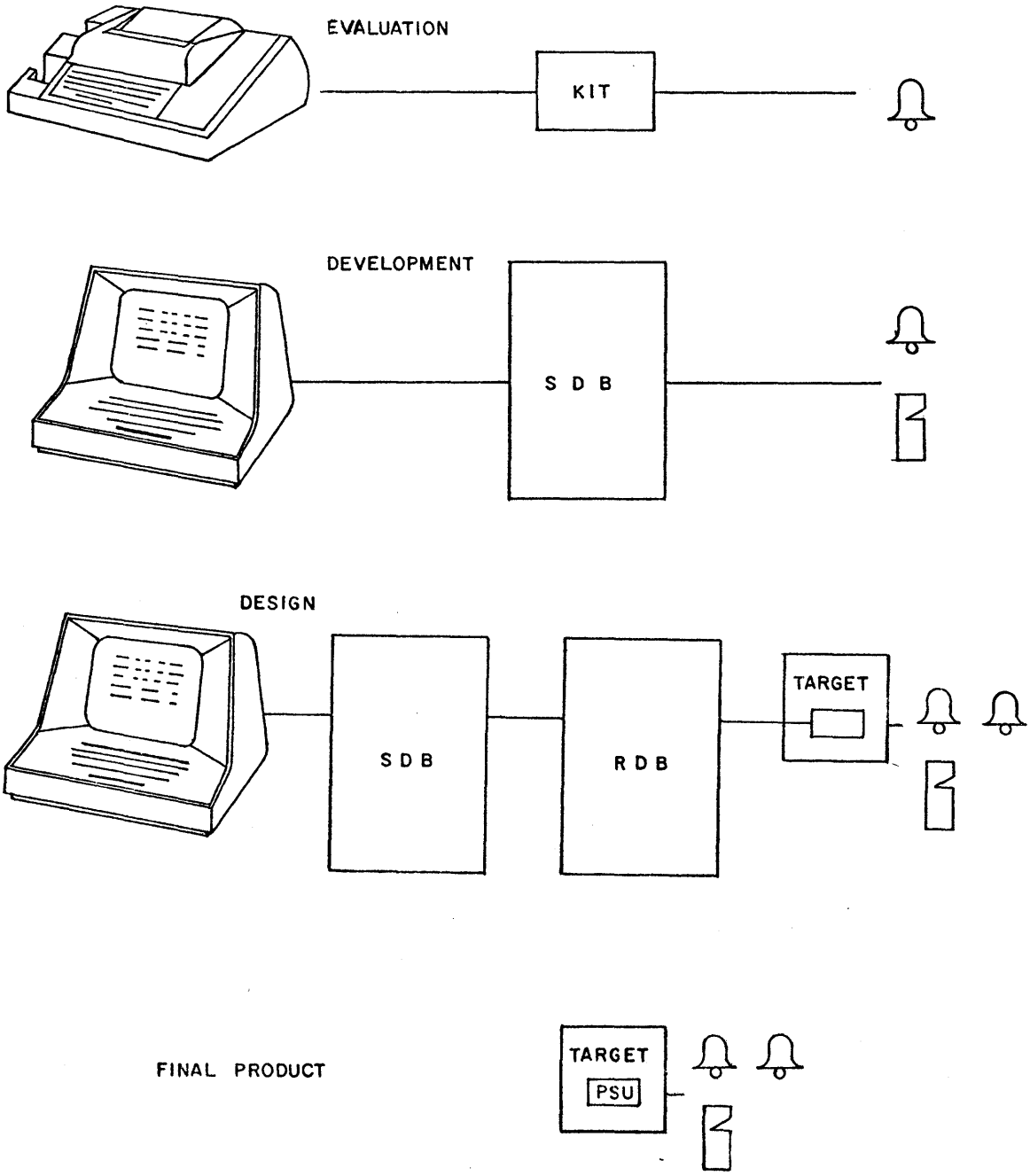
The F8 is designed with a certain parsimony in providing basic instructions to implement features with programming instead of in hardware. For example, there is no return address stack for nested subroutines; however, there are instructions to implement such a stack in about 20 bytes. There are no multiple addressing or jumping modes; however, there are operations and transfer paths available to permit rapid and flexible implementation of these and related techniques.

Some economies do pose considerable problems for the design system. For example, jumps destroy the value in the accumulator. This is no severe constraint for the application programmer, but it does demand ingenuity in the debugging system to enter and exit programs

preserving the accumulator. As another example, interrupts destroy the value of the subroutine return address, so the debugging system must not introduce any bogus opportunities for this to occur.

THE MOSTEK SYSTEM

As a user develops his application, his design needs will change. The Mostek F8 design system is intended to meet these needs at 4 identifiable stages of design, with a maximum of economy and convenience. See Fig. 2.



Evaluation

The user initially needs an inexpensive ready-to-go microprocessor system that allows him to become familiar with the instruction set, hardware features, and performance. For this a 4x6 inch self-assembled Evaluation Kit was designed, containing:

1K operating system
1K RAM
4 I/O ports
Socket for direct teletype connection
1 timer and interrupt

The operating system is described in more detail below. Essentially it allows the user to key in, execute, modify, and debug a program in machine code, following which he can dump the program onto paper tape for automatic loading thereafter.

Development

Although the Evaluation Kit has been used to fully develop application programs, the user will generally require a more powerful system. For this, the Software Development Board (SDB), an entire microcomputer on a 8x12 inch board, was developed, containing

2K operating system
4K resident assembler
8K RAM
6 I/O ports
Socket for serial communication at any of 6 baud rates
High-speed peripheral drivers
3 I/O channels
2 timers and interrupts

The operating system is an easy-to-learn enhancement of the Kit's, described in detail below. Programs can be assembled immediately on the SDB, with no need to consume time or RAM first loading the assembler. (The SDB is also available without the resident assembler, which may be loaded from tape or added in ROM at any time.) The SDB's RAM is ample enough to develop most application programs, and the user can connect the SDB into his application hardware to model his final product.

Design

With a knowledge of the hardware requirements, particularly the memory needed by the program, the user should be able to proceed directly to the design of his final target system with little intermediate breadboarding. To emulate the PSUs, the ROM Development Board (RDB) was designed, containing

1K RAM to hold program
Switches to set memory, port, and interrupt locations

40 pin plug on ribbon cable, to plug into PSU socket

The SDB is not obsoleted; the RDBs are made to mount in the same rack as the SDB, where they communicate through the backplane. The SDB operating system can then be used to load, execute, modify, and debug the program in the RDBs. Note that this involves the SDB microprocessor controlling the target microprocessor. Note also that this is achieved without any connection or modification to the target system other than the RDB plugs in the PSU sockets.

Final product

With the program in the RDBs developed to its final form, the user may order custom PSUs with maximum confidence. However, he may wish to build prototypes for field testing or other purposes using PROM, and for this the ROM Emulator Board (REB) is available. The REB is functionally identical to the RDB, but uses PROM instead of RAM and therefore operates independently of the SDB.

THE OPERATING SYSTEM

The operating systems of the Kit and SDB use 11 basic commands.

```
M s      display and update memory at s (or T s on SDB)
T s,f    tabulate memory block s,f (or M s,f on SDB)
C s,f,d  copy memory block s,f  to d
P s      display and update port s
P s,f    tabulate port block s,f (on SDB only)
L        load tape into memory
D s,f    dump tape from memory block s,f
H s,f,...compute hexadecimal expressions s,f,...
E s      execute program at s
B s      set breakpoint to exit program at s
S s      step single instruction at s in program (on SDB only)
```

The s,f,d represent operands which may be computed as hexadecimal expressions and, on the SDB, use literals and user-definable mnemonics. For error protection, an incorrect command is aborted by typing a period (the ready character), and no command is executed until a carriage return is typed.

The M command permits fast sequential modification of a block of memory as follows:

```
.M 4100
4100 67
4101 6F
4102 2B 2A    (replace 2B with 2A)
4103 48
4104 AD 66     (replace AD with 66)
4105 0E ^      (back up)
4104 66 ^
```

```
4103 48 40      (replace 48 with 40)
4104 66  .     (return to command mode)
```

On each line the address and byte are displayed in hexadecimal, and the user may optionally update the byte. If the user overshoots a line he may back up by typing ^. The update operands may be expressions also:

```
.M 41DE
41DD 90
41DE FB 418B-*=FFAD      (*=41DE)
41DF 00 LF=0046          (literal F)
41E0 00 L8=0038+80=00B8
41E1 00 .
```

The subtraction shows how a relative branch operand may be computed automatically, with * representing the value of the current address.

The SDB has 3 I/O channels, named console, object, and source, to which any suitable devices may be assigned. The channel assignment table is in RAM, and therefore updated using the M command, but 2 character mnemonics set off by colons are substituted automatically for the addresses and values. For example, to view the channel assignments, the user may type

```
.M :CI
:CI :TK          (teletype keyboard on console input channel)
:CO :TT          (teletype type head on console output channel)
:OI :TR :PR      (update to paper tape reader on object input channel)
:OO :TT          (teletype type head on object output channel)
:SI :TR          (teletype reader on source input channel)
:SO :TT          (teletype type head on source output channel)
4072 80 .        (in regular RAM again)
```

When a device is assigned to a channel as above, the driver is automatically initialized as required. The user may write his own drivers, define mnemonics for them, and then use those mnemonics to assign them to channels as above. The user may define mnemonics for any other addresses also, such as starting points of programs or subroutines.

The T command may be used to provide a compact tabulation of memory at 16 bytes per line:

```
.T 4100, 4127
4100  2B 58 1E 20  00 B7 A5 21  10 94 04 20  CA B7 44 18
4110  81 11 34 21  07 94 17 45  12 55 A5 18  21 80 C5 55
4120  90 0C A5 81  09 21 40 22
```

In this format, it may be noted that the first and second digits of relative branch operands represent corresponding vertical and horizontal displacements.

The P commands operate similarly, but inputting or outputting to ports

instead of memory. Since there are 256 possible ports, each with its specific input or output instruction, the SDB first writes a subroutine to access the one required, and then executes its subroutine.

The L and D commands load and dump object tapes on the object channel in the same format used by the assembler. Checksums are used for error detection, and the addresses of questionable blocks are typed automatically when loading.

The E command is used to execute all programs including design aid programs such as the assembler. The B command is used to set a breakpoint to exit from a program at some predetermined location for debugging putposes. To permit the programmer to see the value of all registers, including the 64 byte scratchpad in the CPU, at the instant of the breakpoint exit, the operating system dumps the registers into a 73 byte area of RAM known as the register map. Just as regular memory this map may then be displayed and updated, using mnemonic addresses in the SDB. The next E command will reload all registers from the map and then resume execution. This permits the programmer to monitor the registers in exactly the same way as RAM.

The E and B commands can thus be used together to start sections of program and then stop for inspection. For more detailed inspection of a routine, the programmer may use the S command to step one instruction at a time. As above, the registers are loaded from the map, the instruction executed, and the registers dumped so the map shows the effect of the instruction. The current address, indicating the location of the next instruction to be executed, and the accumulator are typed, and the programmer can continue stepping simply by typing carriage return.

```
.T 4100, 410A
4100 70 56 20 21  57 28 47 B6  20 3F 56
.S 4100
*4101 00          (has loaded accumulator with 0)
*4102 00          (has stored in register 6)
*4104 21          (has loaded accumulator with 21)
*4105 21.         (has stored in register 7)
.M :06
:06 00            (register 6 in map)
:07 21.           (register 7)
.S
*47B6 47.         (has called subroutine at 47B6)
```

Note that stepping (and executing) can be resumed without entering the address.

```
.B 4108           (set breakpoint at return from subroutine)
.E
!                 (user program types !)
*4108 21          (breakpoint encounted)
*410A 3F          (has loaded accumulator with 3F)
```

When a breakpoint is encountered, the address and accumulator are typed in the stepping format, and the user may continue stepping as above. The breakpoint is cleared automatically to prevent old breakpoints from cluttering up the program.

The execute, breakpoint, and step commands are implemented entirely with software, and some technical details of their operation may be of interest.

In the Kit, a 2-byte register (Q) in the 64-byte scratchpad is sacrificed. To execute, the entry address is placed in Q, all other registers are loaded from the map, and an indirect jump is performed using Q. Unlike regular jumps, this preserves the accumulator. Setting a breakpoint physically replaces the user's code with a jump to the register dumping routine, after first saving the accumulator in Q. The user's code is saved and replaced automatically when the breakpoint is encountered.

In the SDB, setting a breakpoint works similarly except that the accumulator is first saved in a reserved system port to avoid destroying Q. To execute, the program is reentered with a regular jump, but not always immediately. Because this jump will destroy the accumulator, the operating system automatically steps through the program until it reaches an instruction loading the accumulator; destroying the accumulator when jumping to such an instruction is clearly of no consequence. There are 59 possible such reentry instructions:

```
LR A,KU-QL
LR A,IS
LM
LI c
IN p
PI aa
JMP aa
LR A,r
LIS c
INS p
LR PO,Q
POP
```

(the last 2 do not load the accumulator, but perform jumps that do not destroy it). No program can accomplish any processing without using these instructions; typically, only 1.7 instructions need be stepped before a reentry point is reached, so the delay is quite unnoticeable.

Stepping thus becomes crucial to executing on the SDB. It operates without inserting any traps or otherwise modifying the user's program. Instead, the SDB writes itself a short program, similar to the way ports are accessed, in which the accumulator is restored, the required instruction is executed, and the accumulator is saved before jumping to the register dumping routine. The SDB can then reload the registers and jump to this ad hoc program. In this way

the actual instruction is executed (except for special cases of branches and jumps) but not in the program context, making program modification unnecessary. The S command can therefore be used to step through ROM programs for interpretive purposes.

Because the interrupt of the F8 destroys the subroutine return address, a certain number of instructions (including the subroutine call) are 'privileged'. They do not disable the interrupt, but postpone its acknowledgment until the next instruction has been executed, permitting an opportunity to save the return address elsewhere. The S command duplicates this characteristic by permitting interrupts only after nonprivileged instructions.

TARGET COMMUNICATION

When the user is ready to fine-tune his program in RDBs, he can apply all of the above commands to the target system simply by typing T in front of the operands. Thus:

M Ts	display and update target memory at s (or T Ts)
T Ts,Tf	tabulate target memory block s,f (or M Ts,Tf)
C Ts,Tf,Td	copy memory block s,f in SDB or target to d in SDB or target
C s,f,Td	
C Ts,Tf,d	
P Ts	display and update target port s
P Ts,Tf	tabulate target port block s,f
L T	load tape into target memory
D Ts,Tf	dump tape from target memory block s,f
E Ts	execute target program at s
B Ts	set breakpoint to exit target program at s
S Ts	step single instruction at s in target program

This is accomplished by having the target CPU execute a handshaking routine causing it to access its memory and ports as directed by the SDB. On command, it will also send its registers for dumping in the SDB's register map, reload them as directed, and begin executing at any point in its program. The user does not have to concern himself with loading this handshaking routine into his target system; it is automatically introduced into his target memory (from F400 to FBFF) when he plugs in an RDB.

The SDB is used to load the RDB memories to emulate the PSU, so it can perform any of the commands involving only RDB memory without involving the target CPU. To access other target memory and ports and to control target execution, however, the target CPU must be executing the above handshaking routine. This is accomplished by using the SDB to set a nondiagnostic breakpoint within RDB memory to intercept the target CPU; address TO is suitable since the target CPU can be directed there by operating its reset switch. The SDB will then automatically type the breakpoint address and accumulator

value, clearing the breakpoint as before. Handshaking is then established between the CPUs and the above commands can be performed with full generality.

Using this system, it is even possible for a user to develop programs in target RAM (presumably to be placed in conventional ROM or PROM) via a single RDB plugged into his sytem.

When the E command is used to send the target CPU to execute a program, handshaking is of course no longer in effect. The SDB will however be monitoring the target whenever not performing a command from the user, and will automatically report when a breakpoint is encountered and handshaking thereby reestablished.

The user can step through target programs using interrupts, which can cause the target CPU to be diverted to service the interrupt. The SDB will meanwhile wait (for 1 second) for the target to return and resume handshaking, thus permitting interrupt servicing even in a non real-time debugging context.

CONCLUSION

With this type of system, the microprocessor user can proceed to his final product in a methodical and economical manner. A single 8x12 inch board provides the user with all the hardware and programming aids for major software development. A minimum of interim breadboarding is required, and the user can check every aspect of his final target system before committing to ROM.

7. A PRACTICAL MICROPROCESSOR DESIGN EXAMPLE

DAVID C. UIMARI
Microprocessor Marketing Manager
Signetics
Sunnyvale, California

I. INTRODUCTION

With the introduction of a class of electronic components called microprocessors, the hardware implementation of physical systems, governing a wide range of applications has undergone a radical change. The objective of this paper is to provide potential microcomputer design engineers with a practical example of an actual system designed with a microprocessor and to describe the procedure by which this design was accomplished. This process is accomplished by taking the reader through the main steps of a specific electronic design problem, namely, the design of an intelligent typewriter system (ITS), using a microcomputer. This particular design example was selected because (1) the system hardware configuration is useable in a number of other applications with similar serial input/output requirements and (2) the hardware components, mounted on a PC card, are available for evaluation and demonstration.

SYSTEM DEVELOPMENT PROCEDURE

Using the microprocessor as a key system component, the system designer can significantly reduce the hardware component count and, therefore, production costs, but during the prototype development phase, he needs to carefully design the microcomputer software, and the hardware interface between the microcomputer and the "outside world".

The fundamental trade-off that must be foremost in the mind of the designer is: HOW CAN I CONFIGURE THE SYSTEM SO AS TO MINIMIZE THE COMPONENT COUNT AND HARDWARE COMPLEXITY BY PERFORMING MORE FUNCTIONS WITHIN THE MICROCOMPUTER, WITHOUT ANY SIGNIFICANT DEGRADATION IN OVERALL SYSTEM PERFORMANCE (or response)?

The sequence of procedural steps to be followed in the development of a hardware prototype system are familiar to most electronic system designers and managers. For the sake of completeness, this familiar sequence is presented in Figure 1 for a microcomputer-based prototype system. The first block requires the designer or the manager to write a detailed description of the functions the system is to perform; the next section will document the functional specification for the ITS.

On the basis of this specification, a suitable system hardware configuration must be defined to (1) meet the interface requirements between the microcomputer and the "outside world" and (2) provide adequate capability within the microcomputer to meet the functional specification. In general, for a particular microprocessor, some ingenuity is required to accomplish these requirements economically. (These new components, therefore, do not supercede the need for clever engineers.)

The next step is to design the microcomputer program. By program is meant the "customized" sequencing of logical, arithmetic and control operation of the microprocessor to meet the desired functional specification. The system designer begins by breaking down the functions required into a set of elementary procedural steps arranged in a systematic and clearly defined manner by suitable program description.

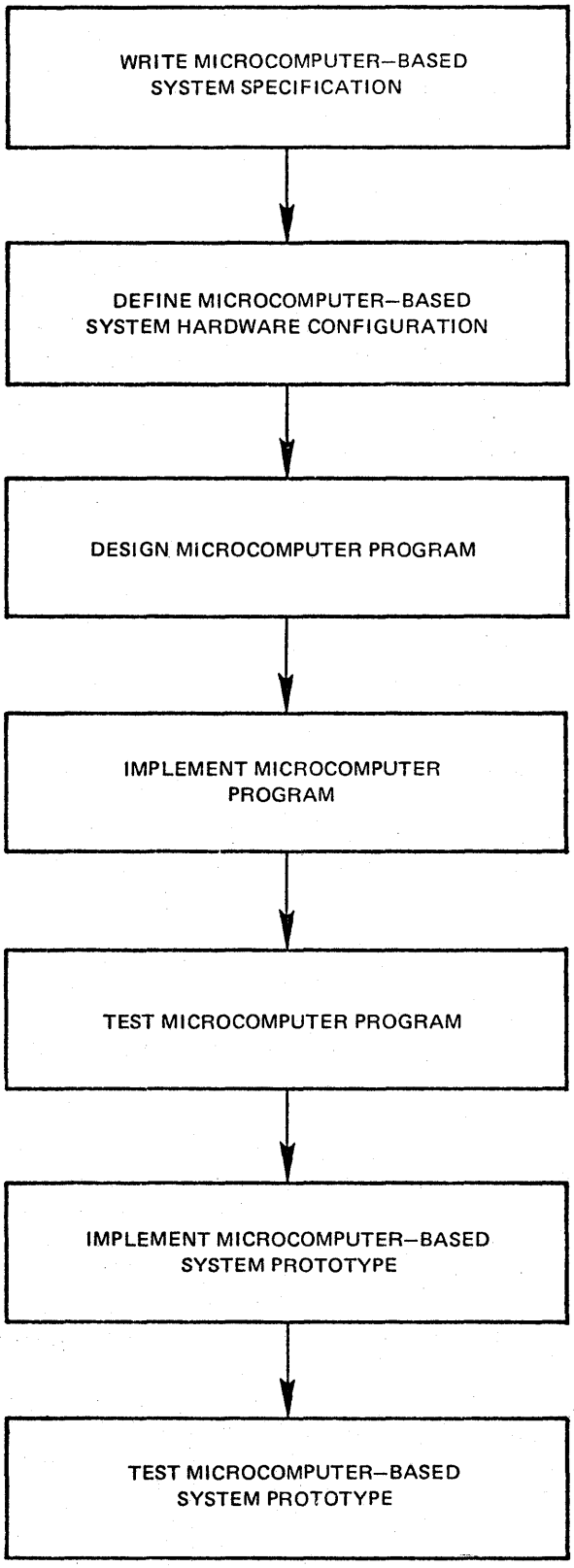


Figure 1 Prototype Development Procedure

The microcomputer program designed in the previous step is then implemented and tested in the two following blocks of Figure 1. The ease with which the program is implemented and tested, largely depends on the usage of proper structuring techniques during the program design process in the previous step. Then, a microcomputer-based hardware prototype system is implemented, incorporating the previously tested microcomputer program. Successful testing of they prototype system completes the prototype development.

THE DESIGN PROBLEM: AN INTELLIGENT TYPEWRITER SYSTEM (ITS)

The overall design problem is to implement an intelligent typewriter system (i.e. text generating system) which outputs a "previously specified" text, with certain blank spaces that can be filled in by the user, to "customize" the text (e.g. a form letter with the name, age, and social security number of the individual to whom it is to be sent). The input medium for the "perviously specified" text is to be the familiar typewriter keyboard. The output medium is to be the typewriter printing mechanism. Moreover, control characters need to be implemented into the system to allow insertion of unique characters at locations identified during text generation. Additional control characters will be required to provide an edit (i.e. erasure of the previous character entered) and system reset capability.

The above functional specification of the intelligent typewriter system (ITS) expressed in commonly used English language is reworded in more precise technical terminology later. In this section certain hardware constraints are imposed and the functional usage of the various control characters is defined. Then, the hardware and software configuration details, as outlined in Figure 2, are generated.

For the typewriter mechanism, we will employ a teletype (TTY) terminal. We will use this device because a microcomputer must always employ an input/output (I/O) device or devices. The TTY can perform all the I/O functions for our application.

Operation of the TTY is very similar to operation of a typewriter with the exception that the TTY has some additional keys. Figure 2 shows the TTY keyboard. The keyboard includes the familiar alphameric keys found on a conventional typewriter. In addition to these, there are several control keys. These are described in terms of operation of the ITS as follows:

1. The LINE FEED key advances the paper, on which the TTY is printing, by one line.
2. The TTY printing mechanism moves from left to right while printing. The RETURN key moves the printing mechanism to the left hand margin.
3. Recall that the user will be typing into the microcomputer memory. (This will include letters, numerals, line feeds, and returns.) The RUBOUT key will be used to delete from memory the last typed character or control key. Additional preceding characters can be deleted by continuing to press the RUBOUT key. RUBOUT will affect the editing function of the ITS.
4. The ITS feature for producing form letters will be achieved using the following set of controls. When the user reaches a point in the letter where unique information is to be inserted, he will simultaneously depress the CTRL key and character B. (We will refer to this combination as CONTROL + B.) This will cause the material written back from memory to stop so that the unique information may be typed in by the user. After the user has typed in the unique information, he can resume the typing from memory by depressing CONTROL + C.

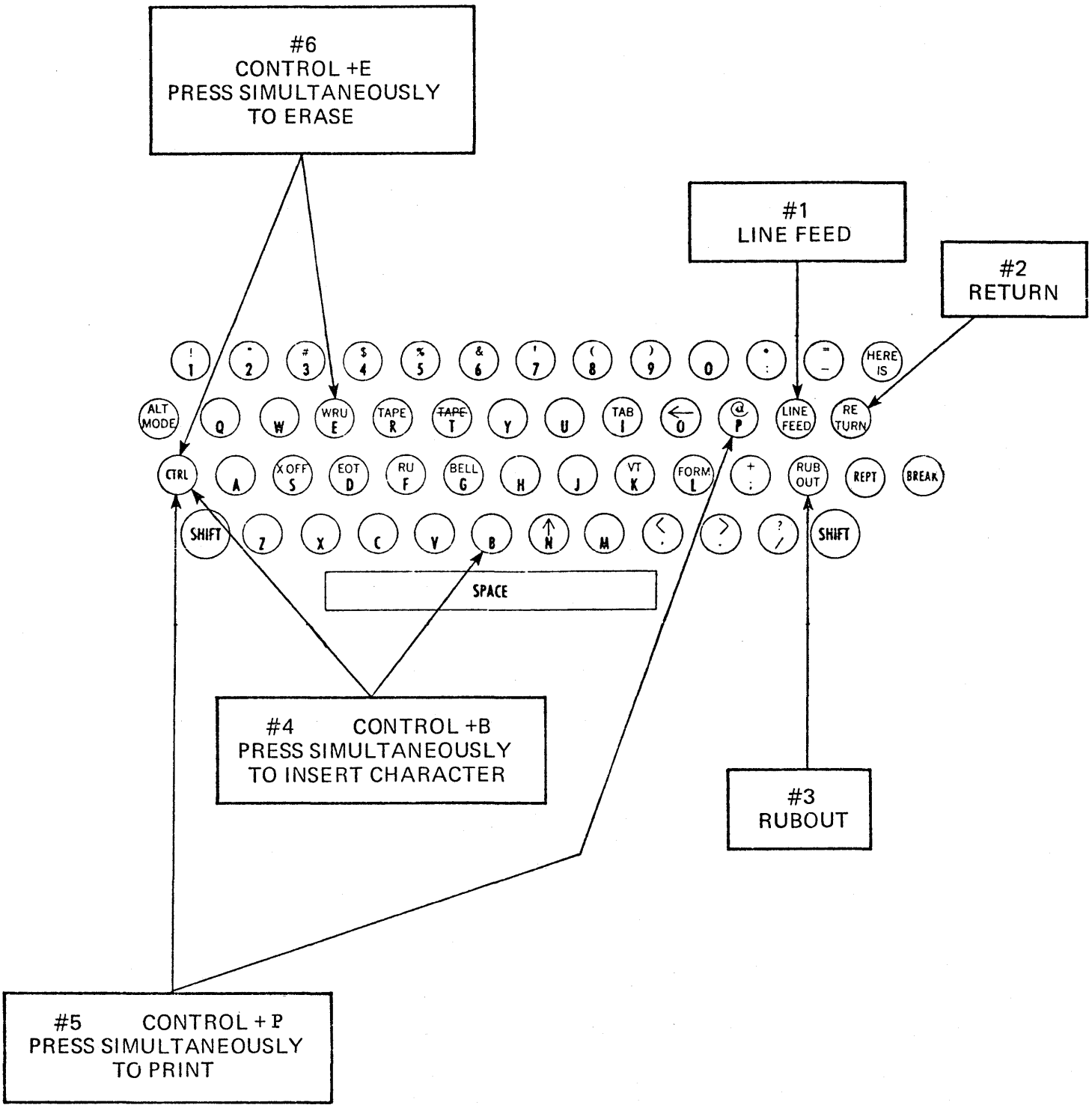


Figure 2 TTY Keyboard With ITS Commands

5. We will provide the user with a command CONTROL + P which will initiate printout from microcomputer memory.

6. Finally, the entire storage of the ITS can be erased by depressing CONTROL + E.

The TTY terminal has a bell which can be operated on command from the microcomputer. We will ring the bell when the user:

1. Attempts to store more data characters than the ITS storage will permit. Let us limit this to 250 characters.

2. Tries to read an empty memory.

3. Attempts to delete more characters than exist in storage.

4. Attempts to continue printing after the contents of memory have been printed out.

DESIGN AND IMPLEMENTATION OF THE INTELLIGENT TYPEWRITER SYSTEM (ITS)

Now, we can proceed with the design problem. We will begin by considering the interface requirements for the teletype keyboard and typing mechanism; this requirment must be met by both a conventional design using standard circuitry (i.e. LSI, MSI, SSI) as well as that using the microprocessor as a system component. Then, we will briefly consider a system level block diagram of a conventional design and make an estimate of the IC packages required. Following this, we will examine a hardware configuration using the Signetics 2650. This will be followed by the software program design and implementation details.

SYSTEM OVERVIEW

Based on the specification, we can depict the system hardware block diagram, as in Figure 3. Essentially, the system consists of a teletype (to enter and type the text), control circuitry (to implement the desired functions) and memory (to store the text).

The Teletype (TTY) is a standard device which encodes each of the keyboard character keys into a unique bit pattern which is seven bits long, together with a parity bit for error control. Similarily, when the teletype receives characters encoded in this manner, the typewriter mechanism is activated to print the appropriate symbol. This standardized serial data input/output procedure is graphically depicted in Figure 4.

Referring to Figure 3 we note that when the operator pushes a key, a unique serial bit pattern is sent to the control circuitry. The control circuitry must wait until the entire bit pattern is received and then send it over the same serial channel to the typewriter print mechanism so that the operator can visually verify that the correct character was received by the control circuitry; this process of retransmitting the received data is called echoing.

The command specification, together with the teletype serial input/output process described above, gives us adequate information concerning the user and the teletype interface. Referring to Figure 3 we note that whatever the hardware implementation of the control circuitry, at least 256 bytes of memory will be required to store the text and the corresponding commands, and then send them back to the teletype print mechanism at the request of the user.

The above description completes the discussion of the common parts of the system implementation. In the following sections, two possible hardware implementations with the associated software details will be considered. The first implementation uses conventional hardware circuitry and, thus, no software is required. The second implementation incorporates the Signetics 2650 microprocessor as a system component; a number of functions previously performed by hardware are now performed by the microprocessor program.

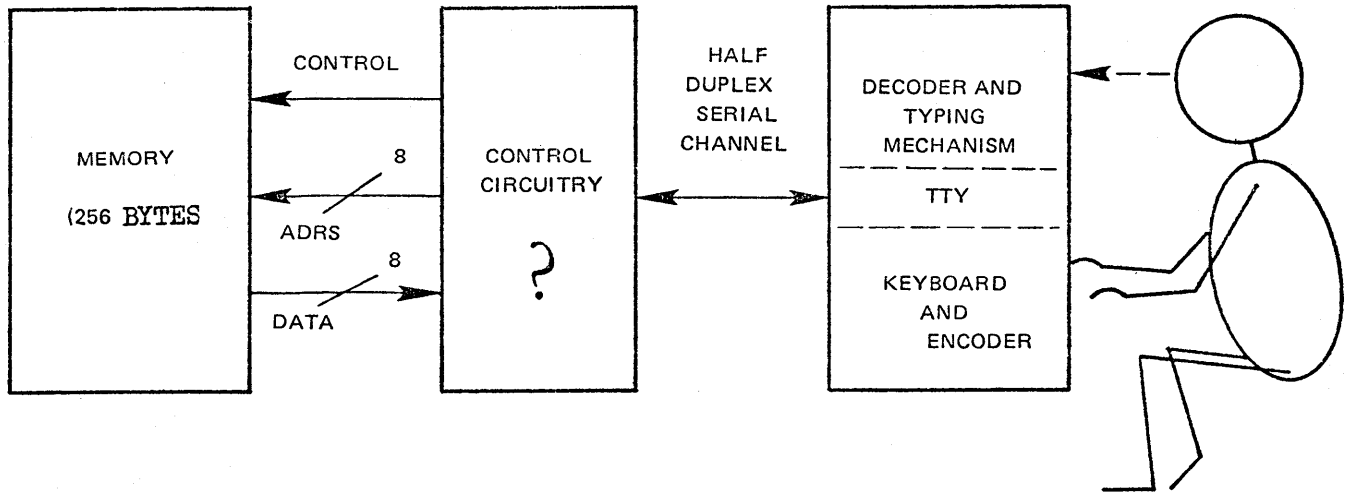


Figure 3 ITS Block Diagram

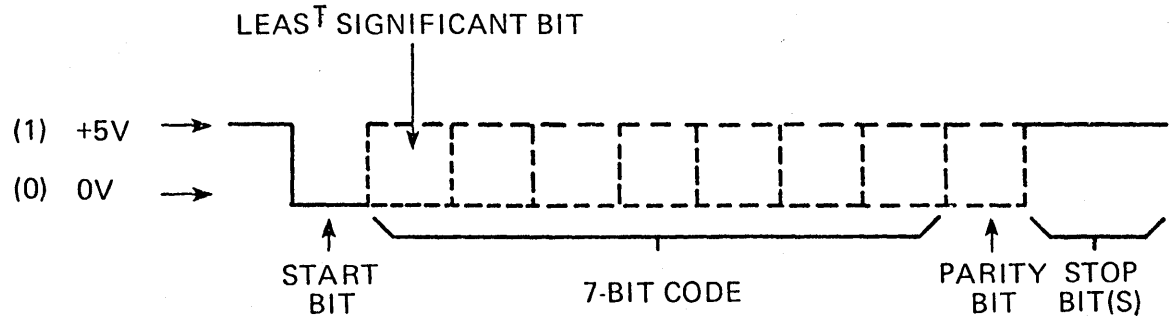


Figure 4 Teletype Serial Data I/O Transfers

Before proceeding with the actual designs, let's examine, at the outset, the main software and hardware features of each. As can be seen from table I the second design proposed is significantly superior to the first. We note that, (1) the hardware electronics parts count is reduced by a factor of 10:1, (2) support components are significantly reduced, (3) prototype development is more methodical and, therefore, less expensive, and (4) production costs are significantly reduced.

ITS RANDOM LOGIC IMPLEMENTATION

The random logic implementation of the intelligent typewriter system requires, first of all, a serial/parallel converter. This is an LSI integrated circuit which converts from the serial transmission mode (one bit of information at a time) of the teletype to the parallel mode (several bits at a time) of the memory and vice versa.

One possible serial/parallel converter that could be used is the TR1602 asynchronous receiver transmitter. The TR1602 has 40 pins of data lines, control lines, and power supply lines. Dual power supplies of +5 and -12 volts are required. Control lines are for receiving and transmitting data, error indications, clock, reset, and format control.

As noted previously, for all implementations, each memory word is required to be eight bits wide. A suitable memory component is the Signetics 2606 static RAM. Its organization is 256 words of 4 bits each. So two packages will provide the necessary 256 bytes (8 bits wide) of storage space for the text.

The largest and most complicated portion of the ITS is the control. It can be designed from TTL, SSI, and MSI integrated circuits. Figure 5 shows the hardware block diagram for the ITS using a conventional logic approach. Remember, each block contains many integrated circuit packages.

First of all, it is necessary to control the TR1602 Asynchronous Receiver Transmitter. The 37 lines of data and control are controlled by three functional blocks: 1) Receive Data Control, 2) Transmit Data Control, and 3) Miscellaneous Control, each controlling its respective function.

A clock is required to drive the TR1602 and possibly the rest of the system. The clock block performs this function.

Memory Control controls the 2606 memory. Addressing the memory, data flow control, read or write operation select, and chip enable are the functions this block provides.

The Character Storage Control block controls storage of characters received from the TTY into memory. These are the characters that will make up the printed page when the print command is issued later.

Control-Character Storage Control controls storage of control-characters received from the TTY into memory. This type of character will not be printed when printout is requested, however. Control characters control page format and provide Stop control (insertion of special user information into the letter after a stop). Control Characters are stored in memory.

The Control-Character Control is a major functional requirement of the ITS. It provides the control functions of character delete, memory erase, continue (after Stop), and printout.

Error control performs the error indication tasks of memory overfill attempt, empty print attempt, and erroneous delete attempt.

DESIGN	HARDWARE		PROTOTYPE DEVELOPMENT	PRODUCTION COST ESTIMATE (%)**
	IC PARTS COUNT	SUPPORT * COMPONENTS		
CONVENTIONAL	75	SUBSTANTIAL	SIGNIFICANT HARDWARE DEBUGGING	100%
MICROPROCESSOR BASED	6	NEGLIGIBLE	SOFTWARE DEBUGGING SOME HARDWARE DEBUGGING	10%

*Support components-PC board, connectors, cables, power supplies, cooling, packaging, etc.

**Quantities of 100 units; amortized development costs.

TABLE I SOFTWARE/HARDWARE COMPARISON OF THE DESIGNS

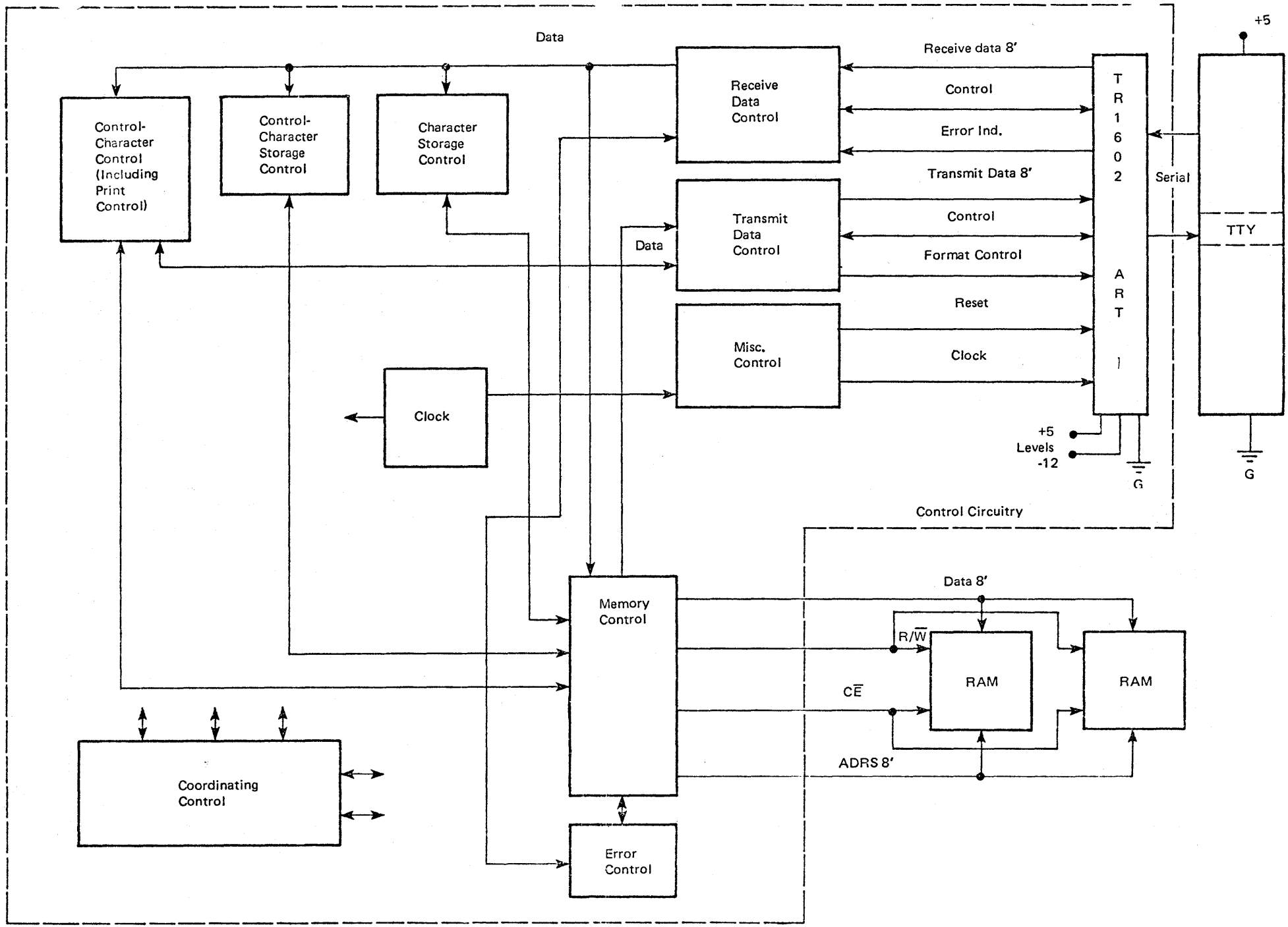


Figure 5 Block Diagram—Conventional Implementation for The Intelligent Typewriter System (ITS)

The coordinating control block is another major functional block in the ITS. It performs the coordination of all the functional blocks in the system.

In summary, the conventionally designed ITS consists of a TTY, TR1602 serial/parallel interface, a memory, and a large control section. The control section must be large and complex to handle the functions of the ITS. And it must be designed from scratch out of a large array of SSI and MSI circuits such as inverters, gates, flip-flops, multiplexors, decoders, counters, registers, etc. 75 IC pagkages are required to implement this random logic version of the ITS.

MICROPROCESSOR-BASED ITS

By designing a general purpose serial I/O interface between the Signetics 2650 microprocessor and the teletype, we can transfer the burden of designing hardware control circuitry to implementing the necessary functions as the random logic based design, with a software program within the microprocessor.

One basic design approach is to use a UART, as in the previous design, to convert from serial teletype I/O to the more convenient parallel I/O. Then, the parallel input/output data bus of the microprocessor is connected to the parallel port of the UART. The additional control circuitry required to accomplish this is presented in Figure 6. The signals lines on the left hand side of the page are the Signetics 2650 pins. These are summarized in Figure 7 and Table II. The number of IC packages to implement this version would be 18 and the length of the software program is less than 350 bytes.

The main ITS software program flow chart depicted in Figure 8, describes the process of text insertion, including the main subroutines. Referring to Figure 8, we begin by utilizing the ITS in the subroutine labeled INIT; this entails clearing the typewriter control mechanism, the keyboard buffers and the memory in which the text is stored. Then, subroutine "IN" gets a character from the keyboard buffer. Since the hardware interface is parallel, the 7-bit character pattern is directly received in register R_0 of the 2650. The line from the teletype input is high (+5V or a logical 1) when no character is being transmitted, as in Figure 4.

In this hardware configuration, the UART handles the task of determining whether or not a character is being sent. Later, we will propose a configuration where this function is performed by software.

The next operation in the basic ITS flow chart depicted by subroutine "CTRL" is the determination of the type of character just received:

1. Character for memory storage
2. TTY control character for memory storage
3. Control character for text control purposes

The sequence of operations that take place within the routine are further expanded in Figure 9. The character just received is compared by the 2650 against known values of control characters. If a match is found, like the RUBOUT control character (Figure 10), from the TTY, the control function is executed. In this example, the RUBOUT character causes delete of the last character in memory. The delete-character subroutine is called by an instruction to execute the delete task. Next, the deleted character is "echoed" to the TTY so the user can verify what he deleted.

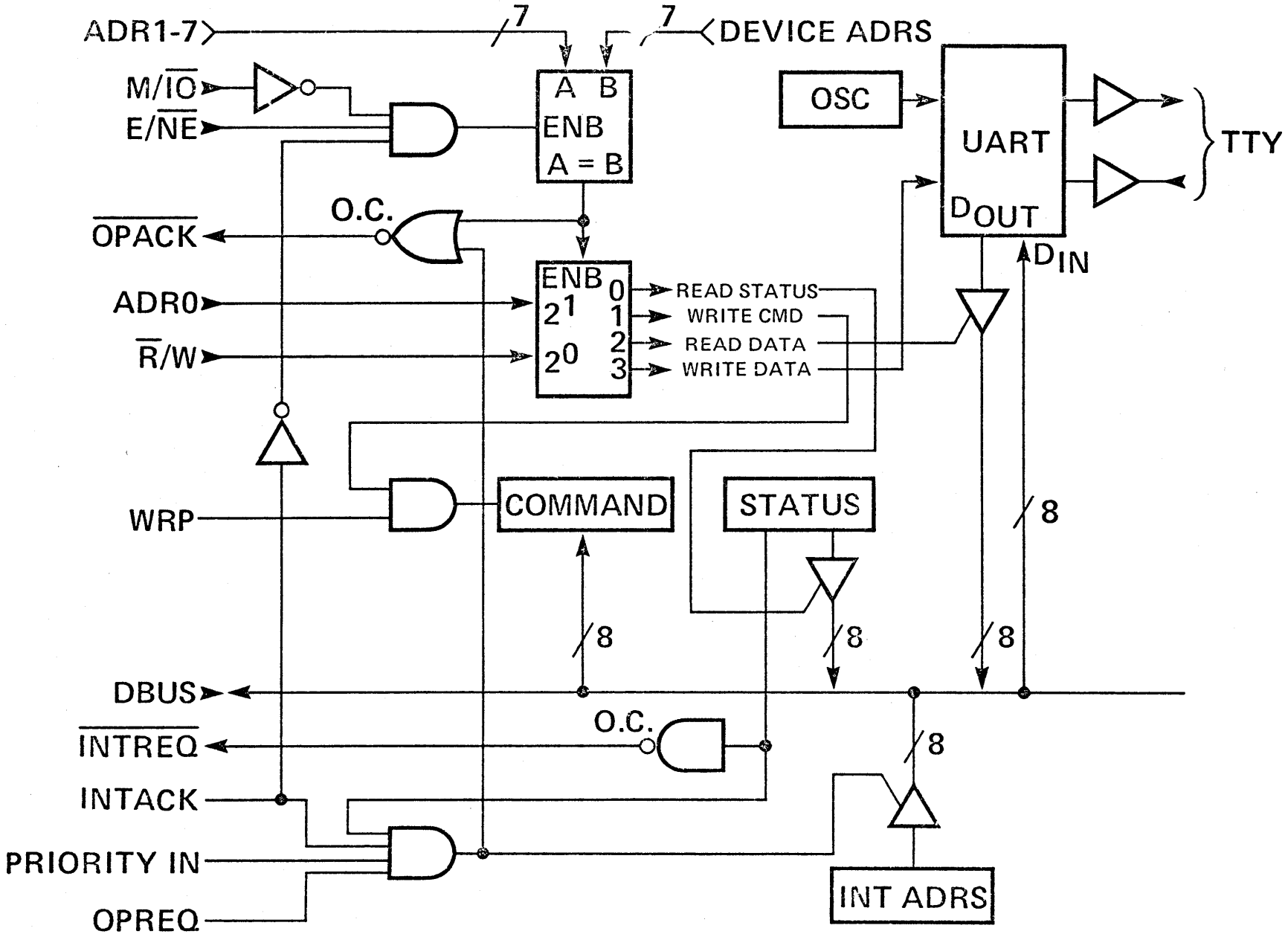


Figure 6 General Purpose Serial I/O Interface (ITS)

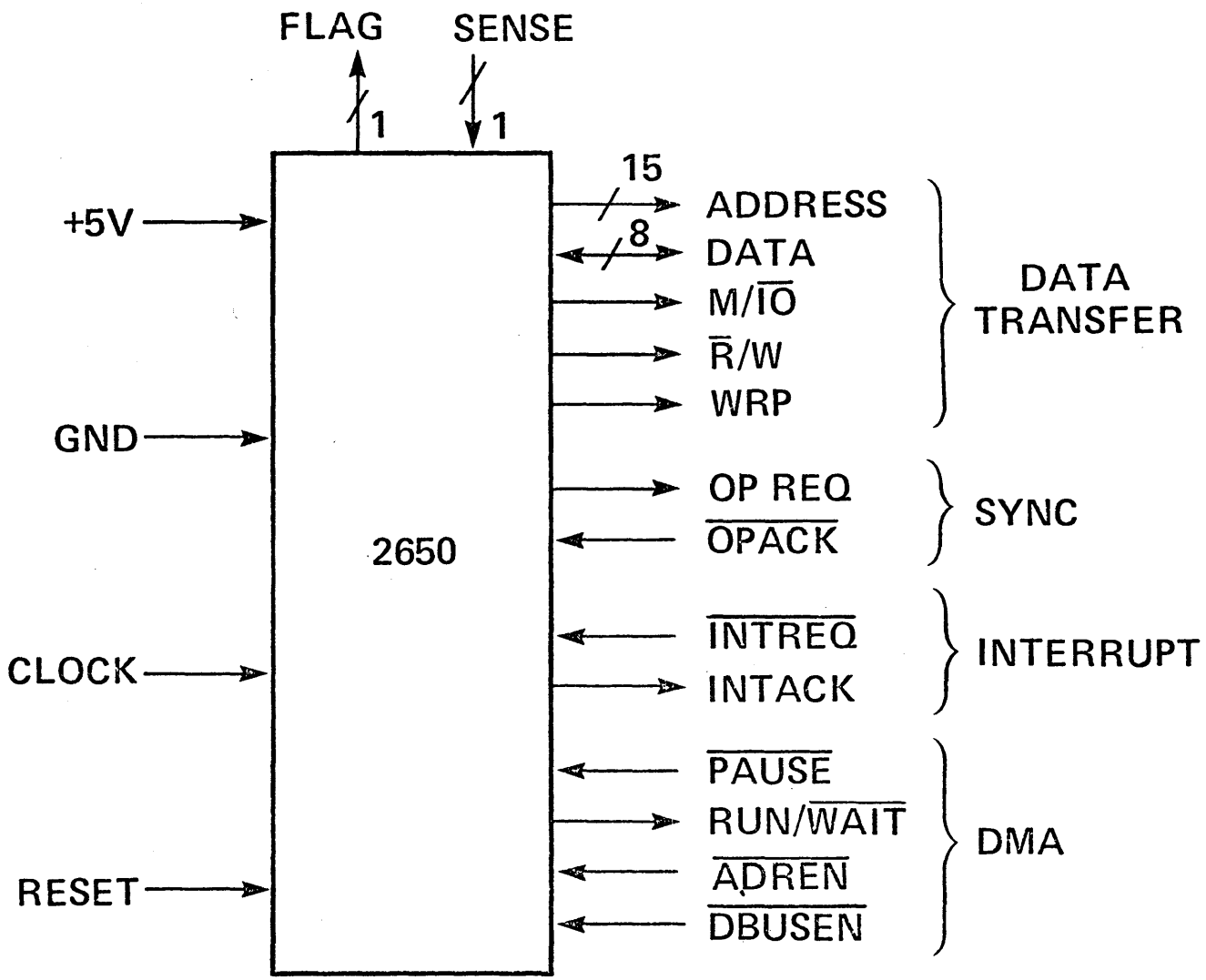


Figure 7 2650 Interface Signals

TABLE II

2650 SIGNALS PINOUT

+5V, GROUND	Power and Ground
CLOCK	Single phase, TTL level, static operation clock input
RESET	Starts processing from a known state (location zero)

Data Transfer Signals

ADDRESS	Addresses program and data memory and I/O
DATA	Bi-directional data bus for program and data memory and I/O
M/$\overline{IO}$	Specifies a memory or I/O device operation
$\overline{R}$/W	Specifies an input or output operation
SRP	A pulse during an output operation
FLAG	An output line located in the PSW. Use is programmers choice
SENSE	An input line to the PSW. Use is programmers choice

Sync Signals

OP REQ	Coordinates all external operations
$\overline{OPACK}$	Response to OP REQ from external device

Interrupt

$\overline{INTREQ}$	External interrupt
INTACK	Response to INTREQ from 2650

DMA Signals

$\overline{PAUSE}$	Request to temporarily stop operation of the 2650
RUN/$\overline{WAIT}$	Indication of the operating or temporarily stopped state of the 2650
$\overline{ADREN}$	Removes 2650 Address lines from the tri-state bus
$\overline{DBUSEN}$	Removes the 2650 Data lines from the tri-state bus

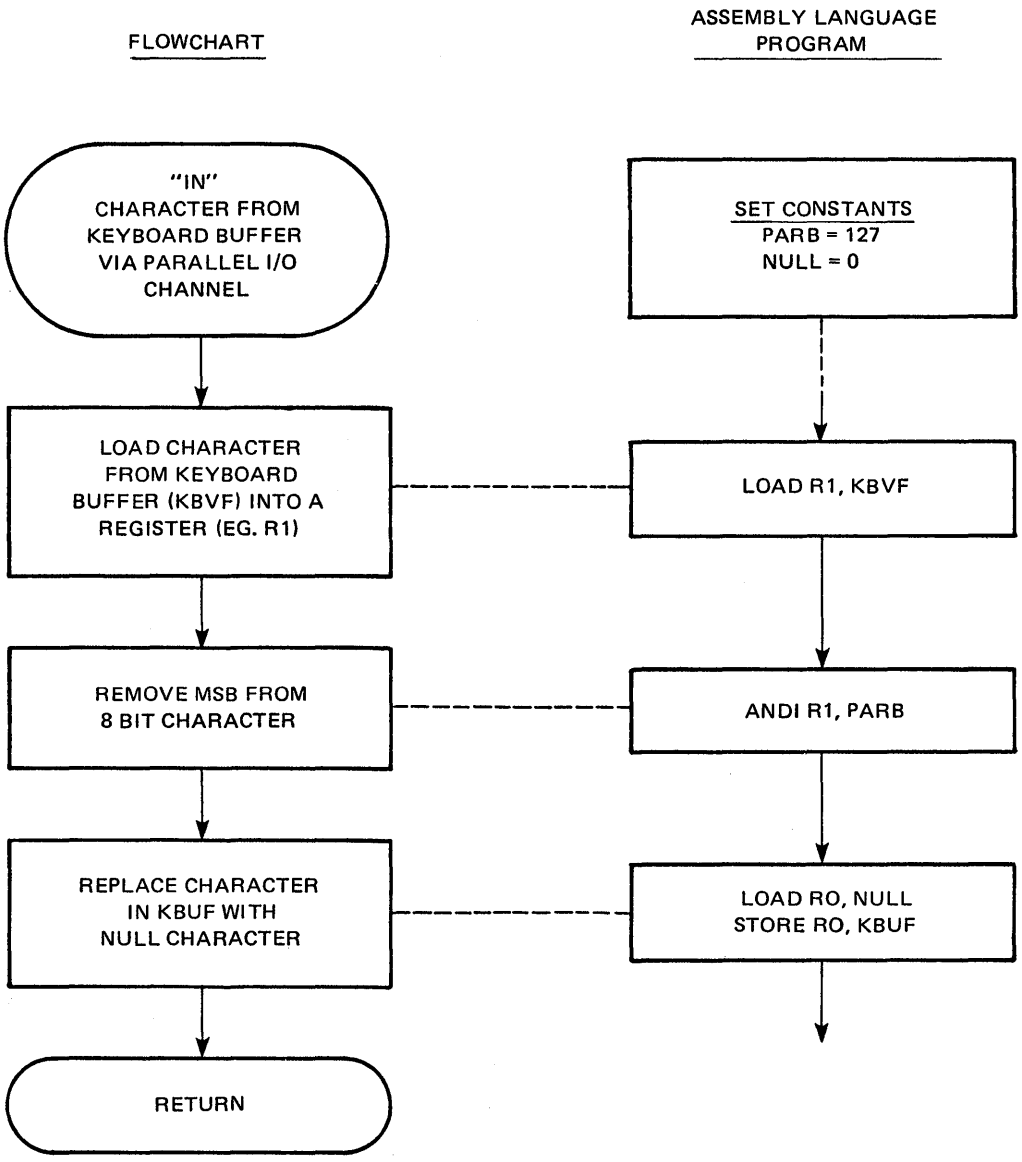


Figure 8

Input Character Routine Flowchart and Corresponding Assembly Language Instructions for a Parallel I/O Channel

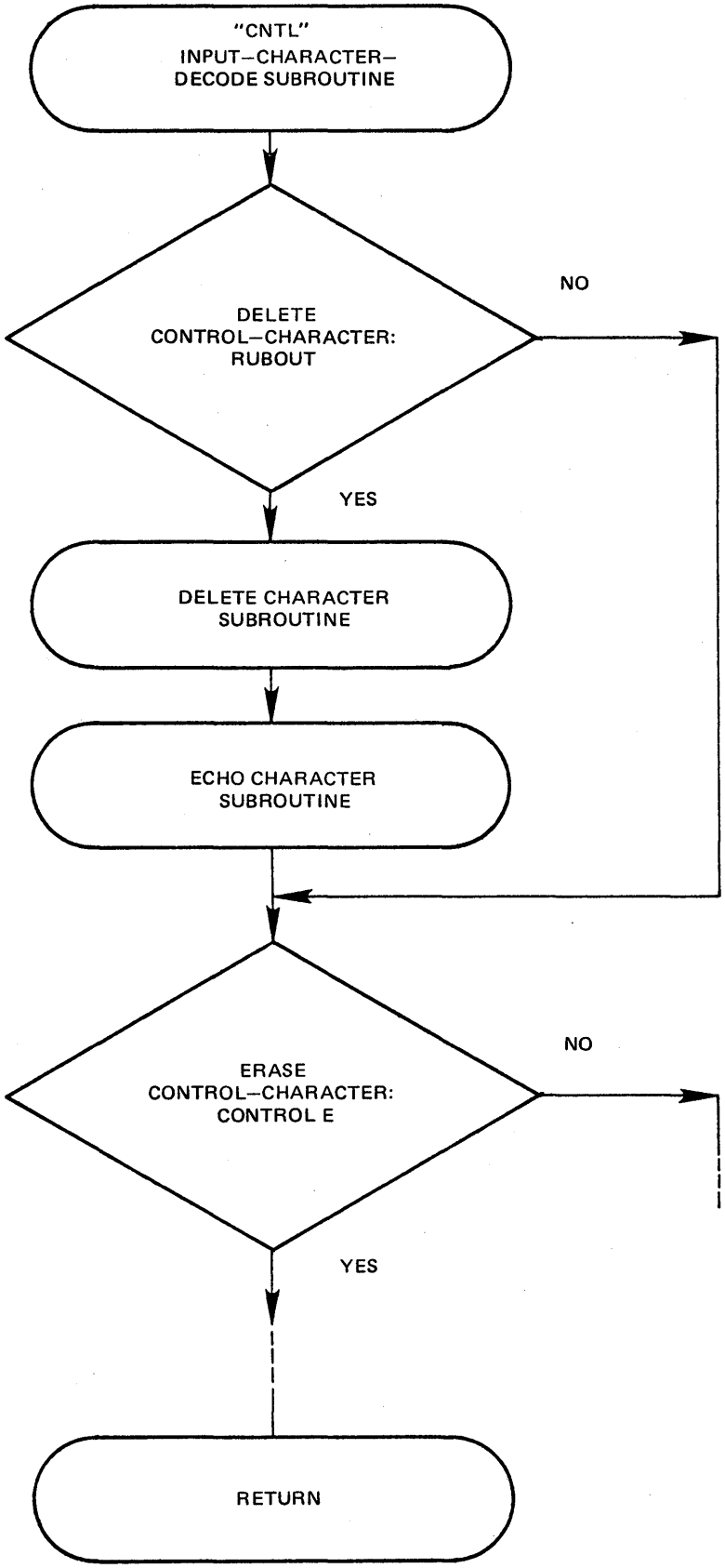


Figure 9 ITS Input Character Decode Machine (CNTL) Flowchart

Proceeding to the next level of detail, let's look at what happens inside the delete character routine, documented in Figure 10. Referring to this Figure, we note that the main operation in this routine is the replacement of the given character with a null character. In the 2650, a null character is represented by an eight-bit byte containing all zeros. This byte is readily generated by the logical function instruction called "EXCLUSIVE OR". All we have to do is "EXCLUSIVE OR" the contents of R_0 with itself. This is accomplished by the instruction:

EXCLUSIVE OR, R_0

Note that it is implicit in this instruction that the other register to be EXCLUSIVE OR'ed is R_0. We will consider a version of the echo character subroutine in the next section.

In the foregoing discussion, we began with the main ITS program of Figure 8. Then, we looked at the flow chart of a specific routine "CTRL" in Figure 9. Subsequently, we looked at a specific routine "RUBOUT" in Figure 10. Finally, we elaborated on the way in which the 2650 generated the null character by the EXCLUSIVE OR operation. This process of sequentially proceeding to the next level of detail until the task to be performed can be described by the microprocessor instructions themselves is called top-down design. Starting with a system specification, the job of the microprocessor-based system designer is to plan the functioning of the entire system by this logical top-down programming process. Thus, the emphasis in developing a good design in a timely manner is to design well-structured, easy to debug/modify/understand programs.

Going back to Figure 8, we see that the next task, after performing the functions in routine "CTRL" is to check the editor status. If the editor is not in the print mode, then, it implies that we are inputting the text; consequently, we add a character to the text buffer memory in routine "SAVE". Of course, if the character was a control operation as described in the last paragraphs, it is not stored in memory. But, if it is one of the following, it is stored in memory:

1. Character for Memory Storage
2. TTY Control Character for Memory Storage
   (like typewriter carriage, return, line feed, or advance paper and stop)

After ensuring that there was, indeed, room left to store this new character, we sent the character back to the teletype printing mechanism (ECHO), so that the user can verify what he typed in. This whole process is repeated in an endless loop until an appropriate command is decoded to indicate the completion of the text insertion task.

MICROPROCESSOR-BASED ITS USING SERIAL I/O

We noted previously that the teletype was a serial I/O device. In the previous microprocessor-based design, a UART was used to convert the serial I/O teletype channel to a parallel channel so that the characters could be input to the 2650 via the parallel data bus. But, for an application involving a relatively low speed device such as a teletype, there is not real need to use the high speed parallel data transfer paths of the 2650.

Referring to Figure 7 and Table II, note that the "sense" bit, in the 16-bit program status word (PSW), is located in the most significant bit location i.e. bit 7 of the upper half of the PSW designated as PSU; and bit 6 is the flag bit in the PSU. These bits are directly accessable on the 2650 pins. These two pins, namely the sense and the flag pins, can be used to implement a serial I/O channel in the following manner.

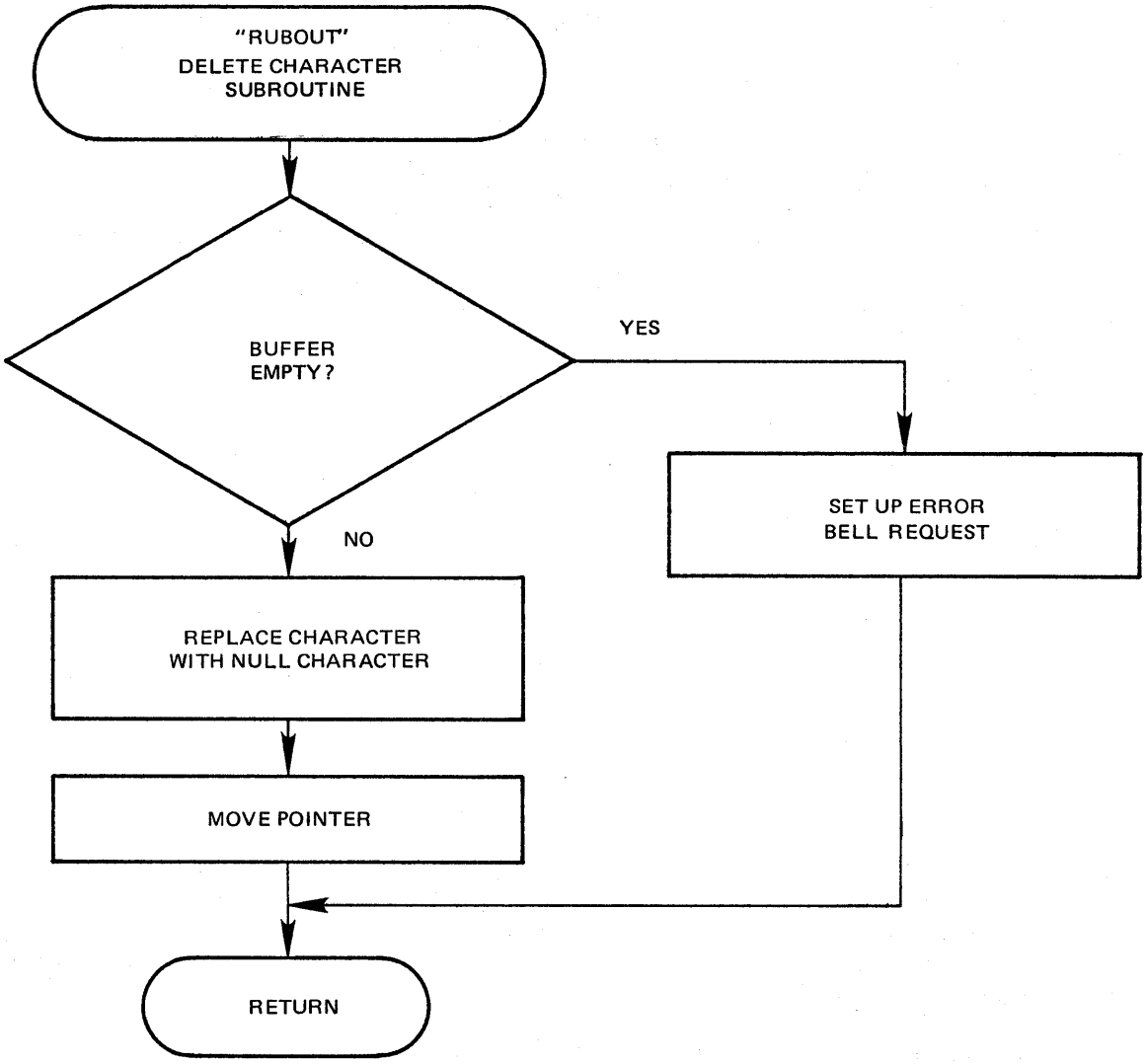


Figure 10 Delete Character Subroutine Flowchart

For inputting TTL compatible serial input data, we can use the sense line. The sense bit is normally a 1 (+5V) between data transfers. The line drops to zero volts (0) to indicate a start bit. Then 8 bits are serially transferred. After this, the line goes back to a 1 (+5V) for one or two stop times, depending on the data transfer rate. This line can be sampled inside the 2650, under software control, by executing a "STORE PSU" instruction which stores the contents of the PSU into R_0 and sets the condition code bit (CC) of the PSW. For outputting TTL compatible serial data, we can use the flag line. To transmit a start bit back to the teletype, we set the flag bit of the PSU to a 0; to transmit a stop bit, we set the flag to a 1. Moreover, to transmit data bits, the flag bit is set the same as the corresponding data bit. This process is accomplished under software control by executing the "SET PSU" instruction.

Thus, we realize that, in the case of this dedicated microprocessor application, namely ITS, there is really no need for the generalized serial I/O interface. Instead, we can directly use the sense/flag pins on the 2650 for serial I/O. The resulting hardware configuration for this dedicated ITS application is shown in Figure 11.

Three control signals from the Signetics 2650 control the ITS memory, not including the address bus. OPREQ is a coordinating signal indicating that an external operation is taking place. OPACK is grounded and unused since the 2606 and 2608 respond in less than 1 μsec to a 2650 request. R/W selects a read or write operation of the 2606 RAM memory, and WRP provides a timing pulse for the same. The 10th address bit, ADR10, acts as a chip select. It places the 2608 in address space 0 to 1023, and the 2606 in the address space 1024 to 2047. ADR10 and ADR0-ADR9 select one location in those address spaces. Notice that we have a total of 6 IC packages and only one +5V supply drawing about 500 milliamps! The hardware for this system is available from Signetics on a 2-inch by 3-inch printed circuit card!!

Now let us look at the software program. Functionally, the software program becomes more simple! We no longer have to generate the UART control signals. The only significantly new software program is one that converts the serial input from the sense line to parallel byte format for further processing and the logic required to set the flag line to echo or print the proper character on the teletype printer. We will look at this program in more detail in the following.

Referring to Figure 11, we note that keyboard processing is done in subroutine "IN". Let us discuss the detailed flow chart and the corresponding program for this subroutine, using the 2650 instruction set.

The flow chart for this conversion is shown in Figure 12. The first job is to continually sample the sense line until a start bit is detected. Then, we introduce a delay of half the bit time to test the sense line again to ensure that it was not a noise spike. After ensuring that it was, indeed, the start bit, we, then, introduce a delay of one bit time to test the sense line for the first bit of the 7 bit character. This process is repeated until all 7 bits are received and put into the proper parallel byte format.

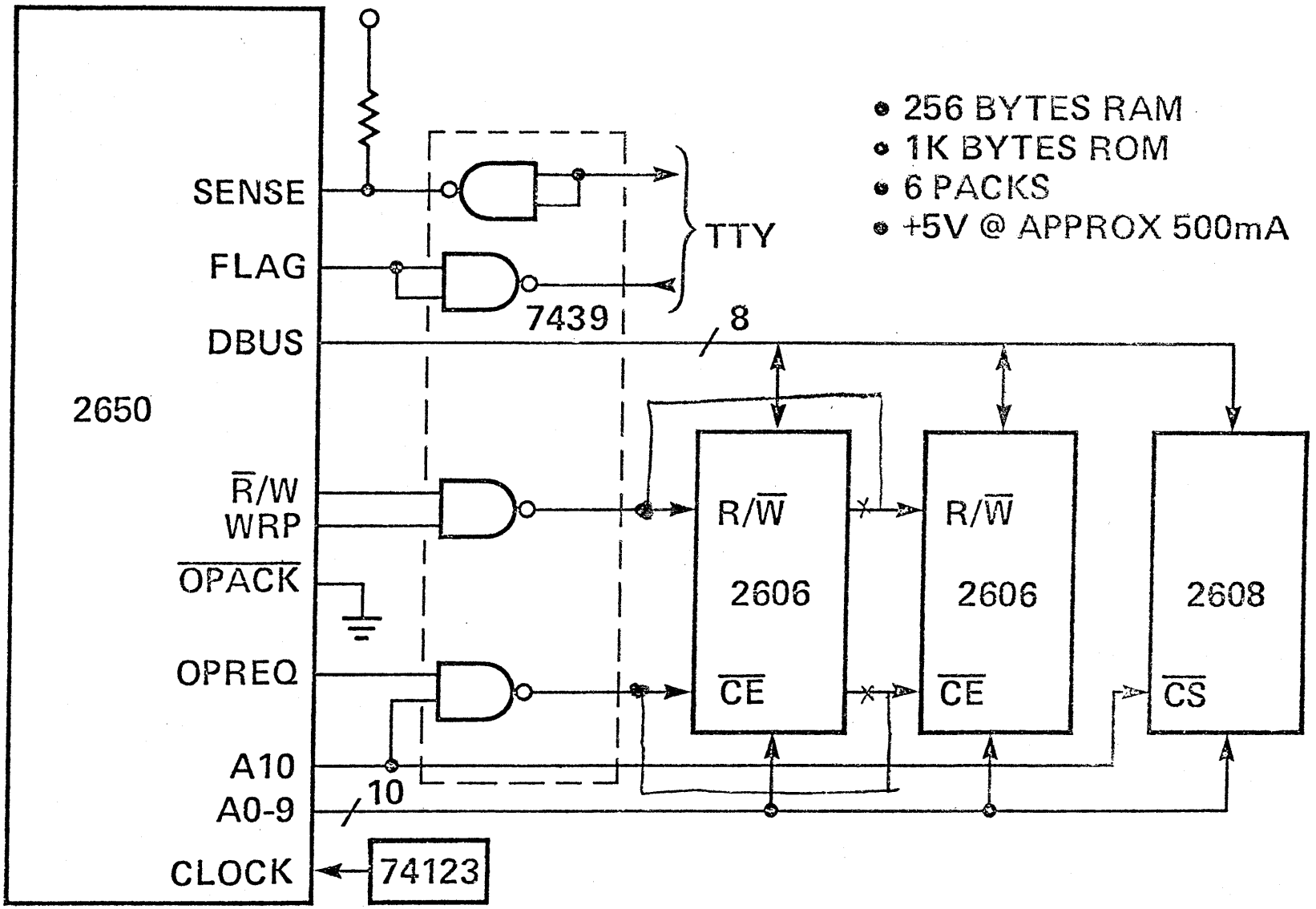


Fig. 11: Dedicated Serial I/O ITS

Fig 12 : Serial/parallel keyboard processing subroutine 'IN'.

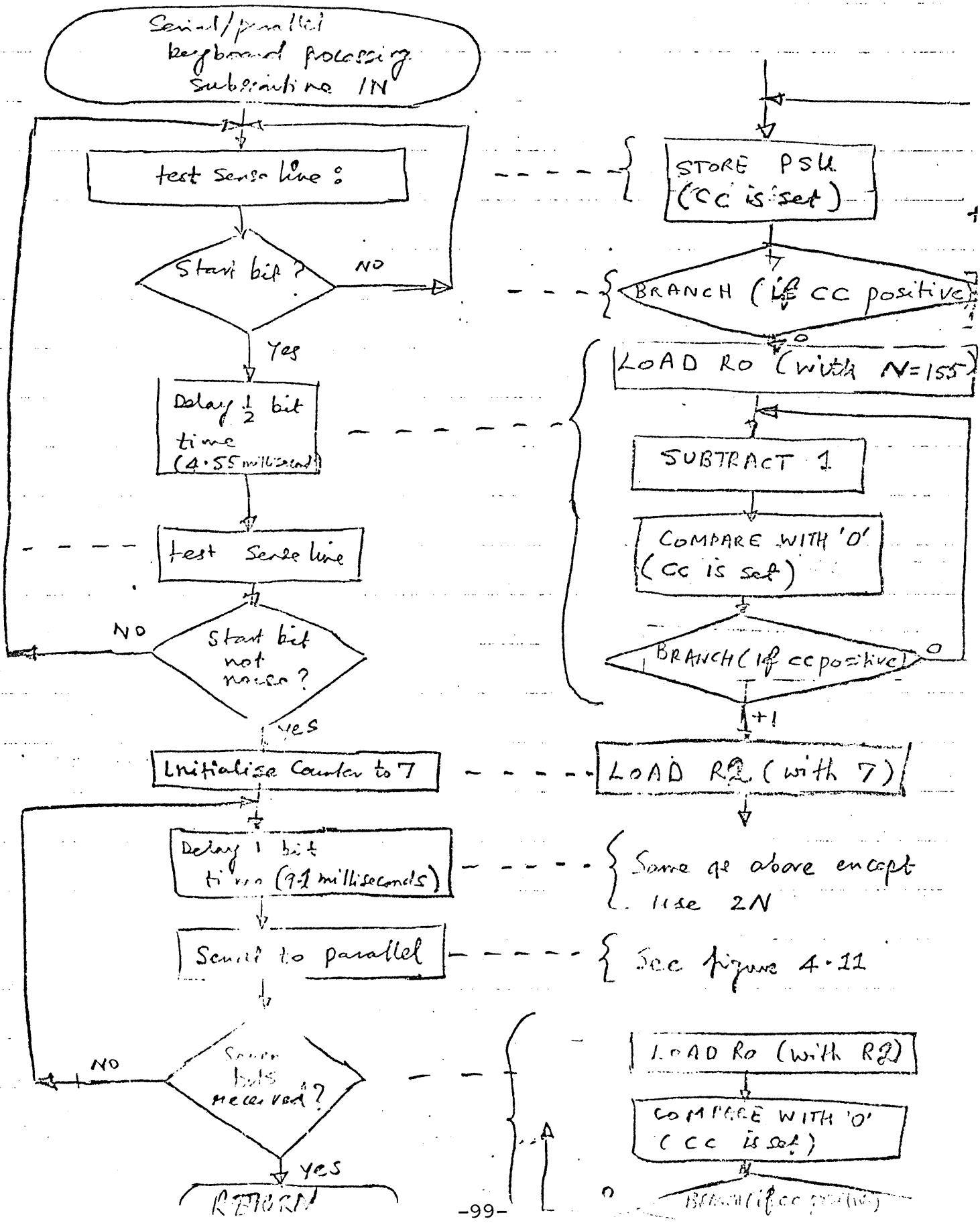
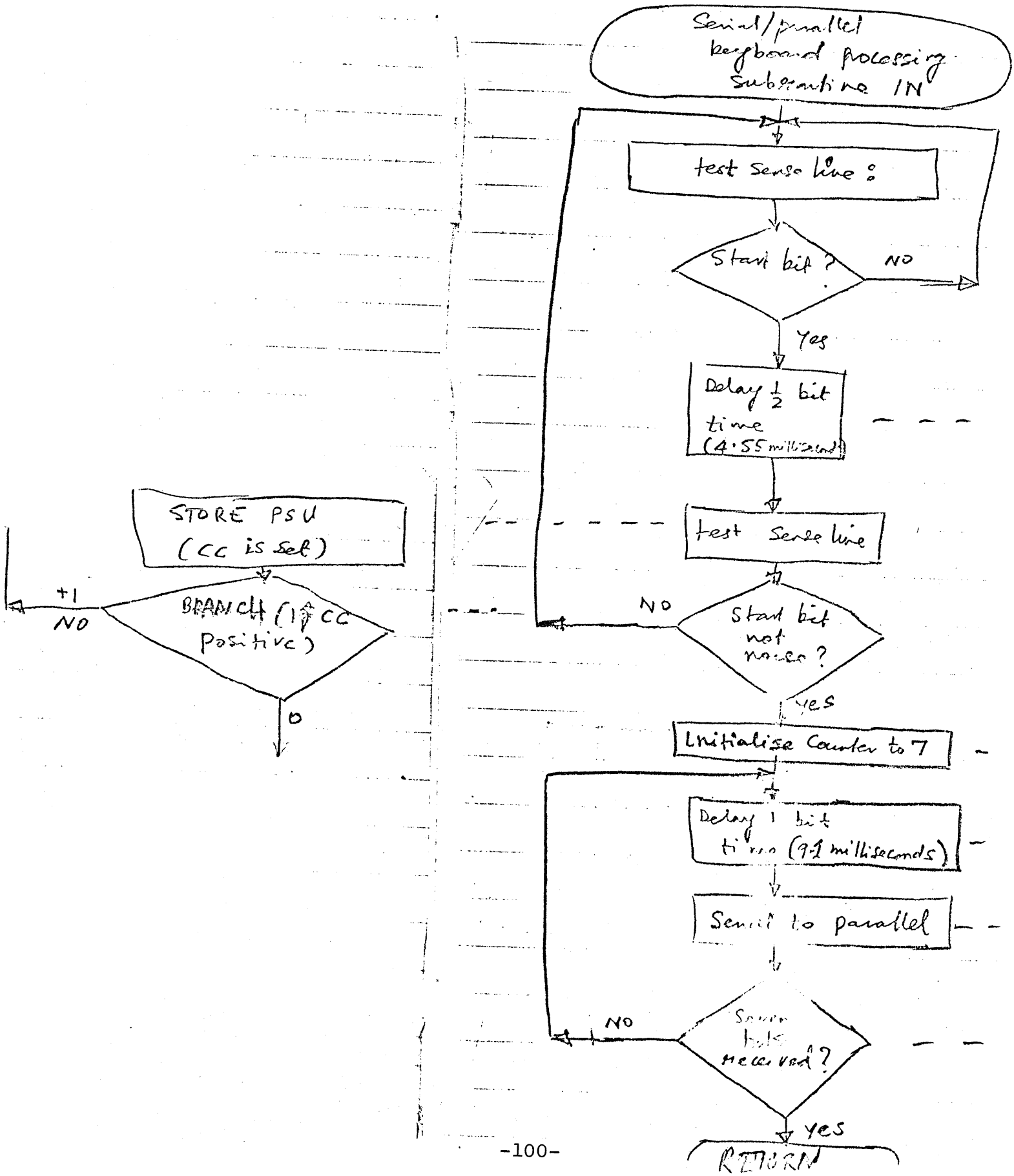


Fig : Serial/parallel keyboard



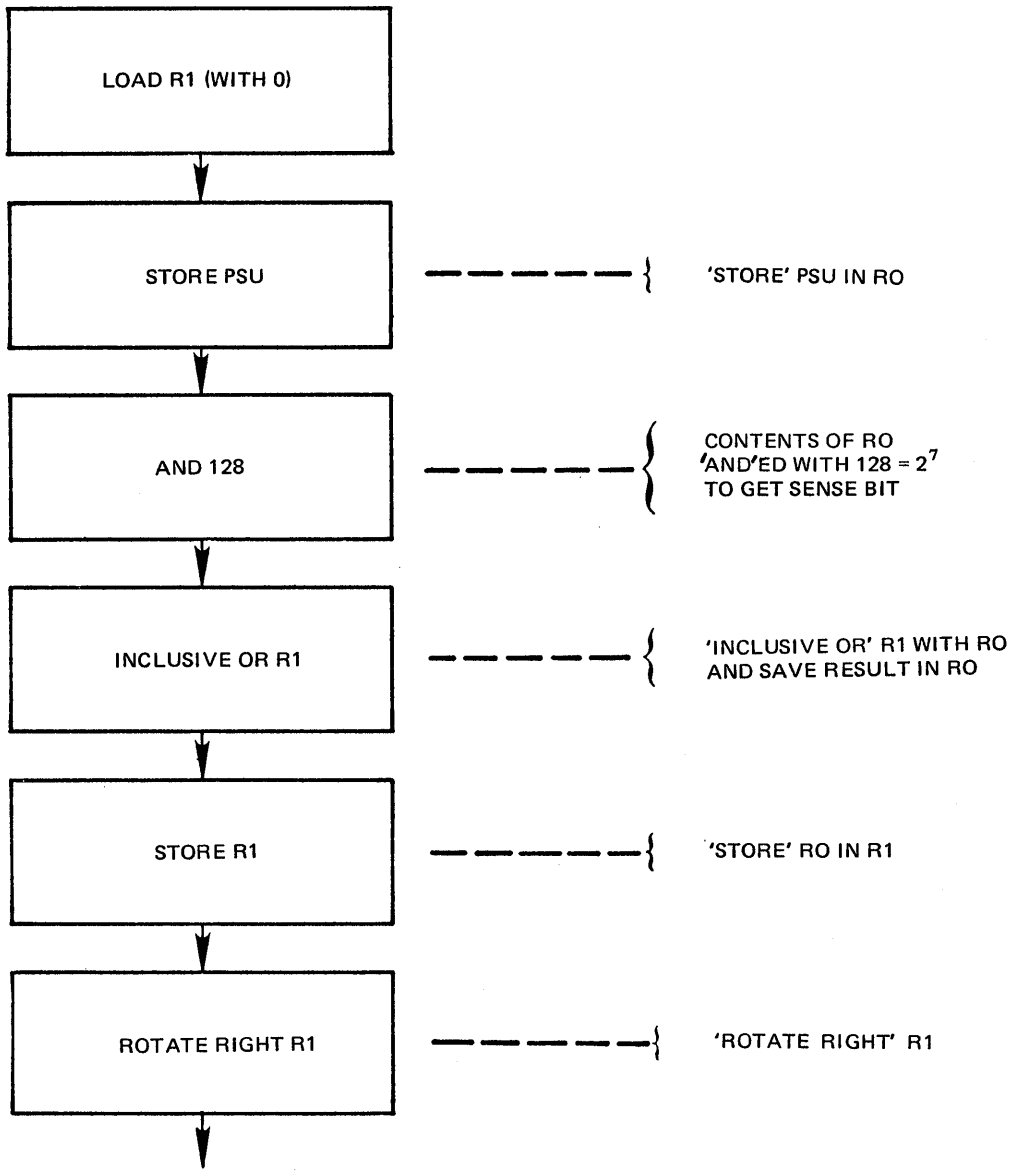


Figure 13 Serial to Parallel Conversion Flowchart

To delay for any timed operation is a simple matter in the Signetics 2650 Microprocessor. Any register, like R0 is loaded with a number. The register is decremented by one each time through a program loop (a loop is a sequence of instructions which transfers execution from finish to start and is usually executed more than once). When the register is tested (each time through the loop) and found equal to zero, the timed delay is complete. The timing is provided by three things:

1. 2650 input clock frequency 1 MHz in the case of the ITS. The 2650 clock frequency is variable up to a maximum of 1.25 MHz.
2. Instruction execution time. The time to execute an instruction is a fixed value which depends on the type of instruction and the clock frequency. The total of the execution times of every instruction in the loop gives the loop delay time.
3. Number loaded into the register being used in the program loop. This is the number of times the loop is executed, and, therefore, the number of loop delay times.

Example

Clock Frequency = 1.25 MHz

Loop contains instructions A, B, and C

Instruction execution times	A = 4.8 μsec
	B = 4.8 μsec
	C = 7.2 μsec
Loop execution time	= 16.8 μs
Number of times through loop	= 100
Total delay time	= 1.680 Msec

Once a valid start bit has been detected, a delay of one bit time (~9.1 msec) is made until the middle of the first data bit. The middle of the first data bit was reached in the following manner: the leading edge of the Start bit was detected because the 2650 program was continuously looking for it in a tight loop; the program loop is very fast compared to the frequency of the sense signal (several microseconds compared to 9.1 milliseconds); so when the start bit was detected, it can be assumed the leading edge was detected and not the middle; the middle of the start bit was located due to the 1/2 bit time delay during the noise check; finally, the middle of the first data bit was detected due to the one bit time delay from the middle of the start bit.

The first data bit is sampled on the sense line as "1" or "0" (high or low), and saved. When 7 bits have been received in this manner (a count is kept in R2), an entire character has been received.

The serial to parallel conversion for each character is accomplished by transferring a data bit from the sense bit into R0 with the "STORE PSU" instruction. The data bit alone is left in R0 after execution of the "AND" instruction. The last data bit sampled is assembled together with the data bits previously received in R0 by the "INCLUSIVE OR" instruction. The "STORE" instruction puts the contents of R0 into R1. Finally, the "ROTATE RIGHT" instruction gets the contents of R1 ready for the next bit of the character.

The correspondance between the verbal flow charts of Figure 12 and 13 and a set of basic instructions developed has been shown.

SUMMARY

We are now in a position to discuss the main reasons why microcomputers have a significant advantage over random logic:

1. Reduces system complexity
2. Ease of development
3. Flexibility (ease of system function modification)
4. Reliability
5. Ease of support
6. Lower Cost

Comparing the two implementations, we can see that the system complexity is significantly reduced. Since the hardware complexity is reduced in terms of parts count, it is much easier to lay out the printed circuit boards; cross talk, and other interference problems are reduced; connections, cabling, cooling and packaging requirements are reduced. Most significantly, the 2650 required only one +5V supply. Other reasons for the ease of development are that a software programs are usually much easier to understand than an equally complex piece of hardware. (Debugging software is much more systematic and, therefore, usually less time consuming than hardware troubleshooting.) For example, problems such as electronic circuit malfunction, interfacing, timing pulse alignment, radio frequency interferance are practically eliminated. Debugging the 2650 is particularly easy because its internal circuitry is static rather than dynamic; consequently, the clock can be stopped to look at its pins without losing data or status. The microcomputer-based system is more flexible and easier to support because of the fact that software can be readily modified and is readily documentable. Reliability is greatly enhanced, again due to reduced parts count.

All the above factors can finally be translated into cost savings to the manufacturer. Software development is a one-time cost that can be spread across the production run. Field support is easier with fewer spares required in stock. Finally, the product can be continually upgraded without altering the hardware packaging leading to market competitiveness in terms of the introduction of newer products.

8.

THE 6710 MICROPROGRAM CONTROL UNIT

JOHN BIRKNER
Senior Applications Engineer
Monolithic Memories
Sunnyvale, California

5710/6710 Logic Symbols (Fig. 1)

The logic symbols of the 5710/6710 MCU can be drawn in the active High (Positive Logic) or active Low (Negative Logic) representations. Data signals on Flag inputs, shift I/O lines and input Field N_{1-8} do not alter their function. The instruction fields however, on change of active state reorder the operation tables. The signal CRAC, which signifies address 511 or 510 in the active High case, now signifies address 0 or address 1 in the active Low case. The MCU also steps through a Control Memory backwards in the active Low logic representation. The active High representation is a little easier to understand and it is suggested that programming take place with this in mind. If data inputs are in the active Low logic polarity it is then a simple matter to invert the result of the information using the I_7 control for all conditional branches.

MCU BLOCK DIAGRAM (Fig. 2)

CRAR Logic

The CRAR is split up into two sections, an eight bit section which can remain unchanged, be incremented, loaded from a subroutine temporary storage register latch and loaded from an 8 bit external field N_{1-8}. The one bit section is the least significant bit of the Address Word and is directly driven from the output of the Flag Status Logic. The output of the Flag Status Logic therefore defines whether the next address is even or odd depending upon the condition of the selected flag signal. This allows two way branching at every clock period.

The eight bit section of the register passes through a pass/increment unit and back through a 3 way multiplexer. The output of the pass/increment unit can also be stored in a temporary storage register where it may be returned to the CRAR at the end of a microsubroutine. The pass/increment also provides an end of page signal CRAC indicating that the CRAR is at address 510 or 511. Incrementation occurs in the pass/incrementor when the MCU is at an odd address ($CRAR_0$=1) or a Conditional Branch has been selected.

The output of the CRAR passes through buffers to output pins. The eight bit section buffers have three state outputs so that when the Output Enable ($\overline{OE}$) is High external signals may drive the Control Memory Address lines.

Control Counter Logic

The Control Counter Logic is 5 bits wide with the logic very similiar to the CRAR logic. The logic includes a register,

pass/decrement unit, temporary subroutine register and a 3-way input multiplexer which can select information from the external field bits N_{1-5}, the temporary storage register, or the pass/decrement unit. The counter logic is used during the two conditional jump instructions. Each time the MCU encounters a conditional jump instruction the control counter register is tested for zero and decremented. If the register was zero then the MCU instead of performing a Jump instruction continues on to the next address pair. The next time the conditional jump instruction occurs the procedure is repeated but now the value of the register is one less. The control counter register can be loaded from the N_{1-5} inputs on receipt of a Continue Load Control Counter instruction.

Flag Status Logic

The Flag Status Logic consists of a loadable Shift Left/Shift Right register, an eleven way multiplexer, an exclusive OR gate and a small amount of control logic. The shift register can be loaded with four flags C,N,V,Z with separate loading selection for C and the group NVZ. The register can be shifted one place Left with N going to C,V,to N, etc., and a logic zero being pulled into Z. The C register bit is placed on the bidirectional input/output line Q_0 and would most likely in a system enter the least significant bit of the Q register in the 5701/6701. A Right shift causes C to be loaded with Q_0 this time acting as an input, and what is in C going to N, etc.

The eleven way multiplexer can be used via the select and control lines I_{3-6} to select one of the following signals: Stored C,N, V,Z, Present C,N,V,Z, logic 0, Q_0, and Q_3. All of these signals can be inverted by having I_7=1, enabling branching on C,N, Etc. The output of the Flag Status logic Ex Or gate is the input to the least significant CRAR register bit. The Flag Status logic also provides a signal to the pass/increment unit to indicate that a conditional branch is present and the unit should increment.

Shift Control Logic

The Shift Control logic provides the connections for a minimum set of useful 5701/6701 shifting options. Two control inputs I_8 and I_9 are used to select the desired option. I_8 indicates which bidirectional buffers are actively sending information and I_9 selects various signals to apply to the shift lines of the 5701/6701. Provision is made in the shifting logic to provide correct sign information during right arithmetic shifts by an exclusive OR of the N and V signals.

Control Logic

The Control Logic uses the instruction control inputs I_{0-2}, the test zero output of the Control Counter Logic, and the signal which indicates that the Control Counter temporary subroutine storage register contains the same value as the five least significant bits of the

CRAR register. These signals are then encoded into the control signals necessary to implement the MCU instructions.

Two flip flops are included in the control logic. The first remembers that the MCU is in a microsubroutine and when a return is encountered it is obeyed if the flip flop is set, and if the flip flop is clear the return is ignored. The second flip flop indicates that a Preprogrammed Return Subroutine is in progress called by instruction 101, and the MCU should automatically return when equivalence of the CRAR and CC subroutine latch is achieved. Both these flip flops are automatically cleared during power on and may be reset by under microprogram control by loading the Control Counter with N_6=1.

Generalized Control System Using 5710/6710 (Fig. 3)

The 5710/6710 MCU, although designed to work efficiently with the 5701/6701 Microcontroller, is a powerful controller in its own right. The device could be used in any application which requires sophisticated control of a complex system. The diagram shows system inputs entering a PLA and being translated into condition variables which can be processed by the MCU. The MCU is driven by a program stored in a control memory and the system outputs also derived from the control memory.

Typical System Block Diagram (Fig. 4)

A typical system using the MCU would require one 5710/6710 device and a number of 5701/6701 Microcontrollers defined by the word length of the machine. The system memory would store both instructions and data. Instructions would be loaded into an instruction register and then the starting address of the microprogram code, defining the operation of the instruction, would be sent to the MCU. The MCU would then step through the microprogram under control of the information stored in the microprogram memory and the information coming from the Flag Status conditions of the 5701/6701s. The MCU would also send the programmed connections for shifting operations in the 5701/6701.

System Instruction Fields (Fig. 5)

The 5710/6710 MCU requires up to an 18 bit microprogram word. This field is split into several groups I_{0-2} which define the MCU operation, I_{3-7} which control the Flag Status logic and choose the branch condition, I_{8-9} which control the shifting connections in a 5701/6701 based system, and an external input field N_{1-8} which provides both the jump address and Control Counter value and Subroutine flip flop clear. This 18 bit word can be shortened, first by addressing a smaller memory, second by not using the shifting capability, and third by not using the full branch condition and operation power of the MCU.

The 5701/6701 Microcontroller microprogram word is also up to 18 bits. Eight bits are required for the two port memory addresses A and B. The instruction is split up into two fields I_{0-2}, used for defining the destination of results and the shifting option, and I_{3-7} which define the operands and operation to be performed on the

data. The remaining bits are the Output Enable and Carry In to the arithmetic unit.

5710/6710-5701/6701 Interconnections (Fig. 6)

Interconnections between the two Computer Logic products consist of the Flag signals from the 5701/6701 OVR, C_{n+4}, F=L and the DO_4 output, which are connected to the V,C,Z and N inputs of the 5710/6710 respectively. The other interconnections are the shift connections with SQLO/SQRI of the 5701/6701 going to the Q_3 input of the 5710/6710 etc. The two logic circuits would generally operate synchronously so that the clock inputs of both devices are connected together.

MCU Control Options (Fig. 7)

The MCU has eight instructions controlled by the field I_{0-2}. All of these instructions act only on the eight most significant bits of the CRAR register with the least significant bit of the register under control of the Flag Status logic. The instructions apply to an instruction pair.

The first operation is Continue on to the next instruction pair. If, however, the action is an unconditional branch the next instruction pair is only the target address if the MCU is at an odd address. This enables the MCU to step unconditionally through addresses in the Control Memory by changing the state of I_7. The second instruction is also a Continue but in parallel with the action. The Control Counter is loaded with the value on the external inputs N_{1-5} and if N_6=1 the two control flip flops used during subroutines are cleared.

The next two instructions are Conditional Jumps and cause a Jump to the address specified on the inputs N_{1-8} if the Control Counter is not equal to zero. If the Control Counter is zero the Jump is ignored and the MCU Continues on to the next instruction pair. Again the Continue action changes slightly if an unconditional branch is specified. During these Conditional Jump instructions the Control Counter is always decremented by one each time the instruction is obeyed.

The Conditional Subroutine Jump if the Control Counter is not zero decrements the counter by one and stores the result in its subroutine storage register. In parallel with this operation the present value of the CRAR register is incremented by one if a conditional branch was being performed or the MCU was at an odd address. The Return instruction replaces both the Control Counter and the CRAR register so that the MCU can continue through the main program.

Instruction 101 is a special Subroutine Jump instruction where the return is preprogrammed into the Control Counter. Prior to using this instruction the Control Counter is loaded with a value which is equal to the five least significant bits of the CRAR register. The MCU on obeying this instruction stores the CRAR return address and the Control Counter and then starts the subroutine. The MCU automatically returns when the CRAR reaches the value stored in the

Control Counter subroutine storage register. This instruction allows pieces of code to be used as subroutines without writing subroutines and facilitates the addition and debugging of programs.

The last two instructions are Unconditional Jumps with the Jump address being provided by the field N_{1-8}.

If the MCU encounters a Return instruction and is not in a subroutine then the MCU ignores the return and continues through the program. This feature enables traps to be placed after Return instructions so that programs can be debugged and system errors do not necessarily cause complete system failure.

Flag Status Logic (Fig. 8)

The Flag Status Logic consists of an eleven way multiplexer, 4 bit shift left/shift right register, an exclusive OR gate and some control logic. The output of the multiplexer passes through the exclusive OR gate under control of the I_7 control input and goes directly to the least significant bit of the CRAR register. The logic also sends a signal to the incrementor of the CRAR to indicate that the branch is conditional.

The register which holds Flag Status conditions can be shifted left and right for storage of flags during interrupts.

Flag Status Control Options (Fig. 9)

The Flag Status Control allows one of eleven choices of signal to be used as a branch signal. These are logic 0 for unconditional branching, Q_0, Q_3 and both the present and previous stored value of C,N,V and Z. Selection of the signal is made by a four bit field I_{3-6} and each signal can be inverted by the control signal on I_7. There are six codes which result in an unconditional branch. These codes also cause other actions in the Flag Status logic and allow flag signals to be stored and shifted into and out of the shift register.

Shift Operations (Fig. 10)

The MCU provides a minimum set of shift connections to the 5701/6701. These connections provide both an arithmetic shift left (multiply by 2) and an arithmetic shift right (multiply by ½). Arithmetic shift left repeats the sign bit so as to keep the correct sign of the shifted number. The input to the most significant bit of the 5701/6701 word is the equation $N \oplus V$ which also satifies the situation where two numbers are added or subtracted and then shifted right. This is required during multiplication type routines.

The remaining shift options are rotate left and rotate right with the 5701/6701 register forming an endless loop. Rotates do not include the Q register in the loop.

All shifts are assumed to be double and shift the Register and Q, but the 5701/6701 has separate control over the register and Q so that the decision of whether the shift is single or double length depends on the destination code of the 5701/6701. During a single

length arithmetic shift the connections result in the most significant bit of Q being shifted into the least significant bit of the register. Q must therefore be preset to the desired value or a single length shift left should be derived by adding the register to itself.

I_8 controls the activity of the bidirectional shift connections of the MCU and in the majority of systems can be tied to the I_0 instruction input of the 5701/6701, thereby saving a bit in the Control Memory word. I_9 defines the inputs to the multiplexer which places onto F_0 and F_3 the desired logic functions to realize the various shift operations.

MCU Next Address Table (Fig. 11)

The key to understanding the operation of the MCU and the relative ease of microprogramming using the device is to split the Control Memory into Address Pairs. Each Address Pair has an even and odd member. The member of an Address Pair to which the MCU points is always defined by the output of the Flag Status Logic. If the output of the Flag Status Logic is a 1 at a certain time period then in the next time period the MCU will address an odd member of an Address Pair; if it was a 0 then an even member would be addressed in the next clock period. This even and odd separation allows combinatorial two way branching based upon a flag condition at every microstep with both target addresses specified implicitly. This assists complex porgramming since tags do not have to be attached to jump addresses as in conditional jump branching.

At each conditional branch the MCU always has two target addresses and during a Continue instruction these two addresses are the next Address Pair in sequence. In order to save address space, if an unconditional Continue is specified the MCU only moves to the next Address Pair if the MCU is already at an odd address. This enables a string of unconditional steps to move through consecutive Control Memory Addresses by calling for an unconditional step and programming I_7 to point to the next even or odd address as required.

During jump instructions the target address is specified exactly by the external input field N_{1-8} and the value of the Flag Status Logic output.

Return instructions are a little more complex. Which Address Pair is stored in the Subroutine Temporary storage register depends upon the MCU state at the entry point of the subroutine. If the entry was conditional or at an odd address then the address was incremented prior to storage, and the MCU returns to the Address Pair after the entry point. If the entry was unconditional and at an even address the MCU returns to the same Address Pair which called the subroutine. These characteristics can be used to advantage to program a wide variety of subroutine entry and return variations.

Operation - Add the Contents of R to the Contents of R+1 if N $\oplus$ V is True

This microprogram example shows how the ability to branch on a Flag input, and also its inverse, allows for a very flexible system and saves code. The exclusive OR operation takes 4 words of Control Memory and 2 microsteps.

Operation - Add Contents of R to the Contents of R+1 if NV + VC + NVCZ is True (Fig. 12 - 15)

This is an example of a 16 way branch required by examining 4 variables using the Flag Status inputs. The 16 possible conditions are shown in the Flow Chart. Certain branches need not go to the complete extreme since the result is known earlier. These branches are included in dotted lines and can be removed. The resulting Flow Chart can then be microcoded using the MCU.

The first action is to branch dependent upon the value of N, without specifying as yet the target addresses. Underneath this pair of words a branch on the value of V should occur, and underneath that a branch on the value of C, and then Z. Now the Flow Chart can be followed: if N = 0 then branch to test V. If V = 0 down this branch then a jump is required since the equation is False. If down this branch, however, V = 1 then a branch on C is required. If C down this branch is a 0 then a branch to Z is required, otherwise the equation is True and the addition is instructed to take place.

The programmer thus proceeds through the chart and parallels the chart in the code as closely as possible. Since all target addresses are implicit the translation from chart to code can be performed quickly and easily.

Operation - 16 Bit 2s Complement Multiply (Fig. 16)

This multiplication microprogram performs 2s complement multiplication of two 16 bit signed numbers held in two 5701/6701 registers.

Initially, the multiplier is in register R+1 and the multiplicand is in register R. The first action of the 6701 is to move the multiplier into the Q register, and the MCU steps on to the next address. The 6701 then performs a send zero to R+1 so as to clear it ready for receipt of the partial product and shifts the register formed with zero a Q one place to the right arithmetically. As the register shift is set up the least significant bit of Q, the least significant multiplier bit, is sent out on the SQRO/SQLI pin and is available by the MCU as a conditional flag input. The MCU during this period loads the control counter with the value 14 and the next address is determined by the value of Q_0, the multiplier bit.

In the next step, which is the multiply iteration, the 6701 either adds the multiplicand to the partial product and shifts the result and the Q register right one position or performs just the shift action. The choice is determined by the value of the Q register least significant bit, the multiplier bit under examination. In order to repeat the iteration 15 times the MCU performs a 4 way conditional branch on Q and the value of the control counter. The MCU forces the system to move between A+1,0 and A+1,1 each time performing a double length shift and adding in the multiplicand to the partial product if the previous Q_0 was a 1, and decrementing the control counter. After 15 iterations the control counter reaches zero and the MCU moves the system to the next address pair where the most significant multiplier bit action is handled. Since in 2s complement multiplication the most significant multiplier bit carries a

negative weight a 1 signifies a subtraction of the multiplicant is required. The last step moves Q to R, placing the lease significant part of the product in register R.

The microprogram uses only 7 words of control memory and takes 19 steps. Unsigned multiplication would only require 5 words of memory since the last iteration can be combined into the 4 way branch action.

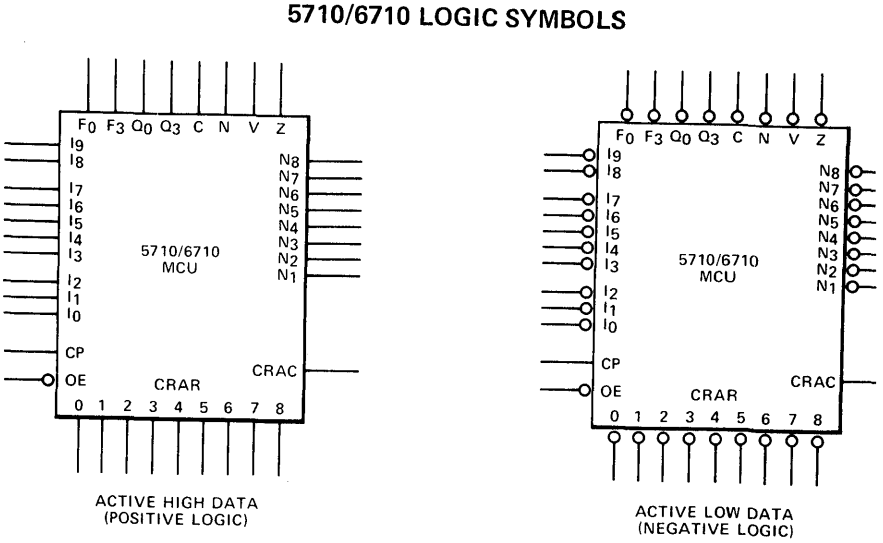


Fig. 1

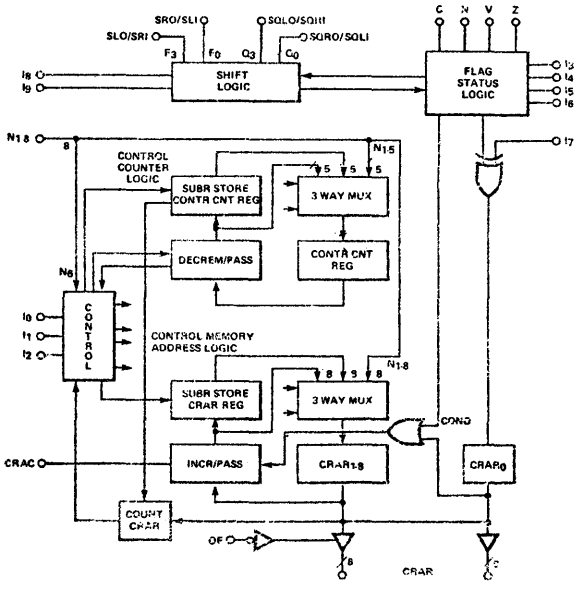


Fig. 2

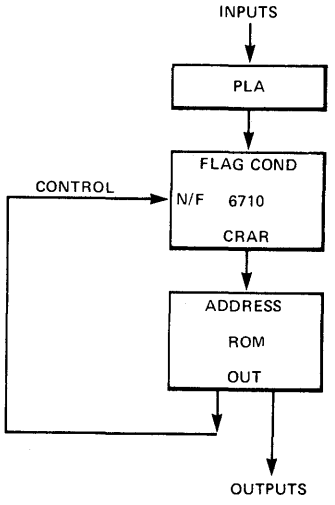


Fig. 3

TYPICAL SYSTEM BLOCK DIAGRAM

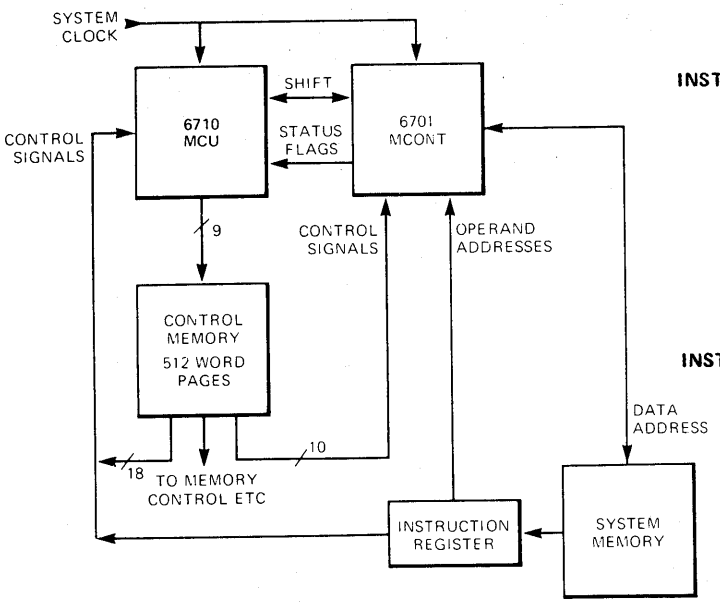


Fig. 4

SYSTEM INSTRUCTION FIELDS

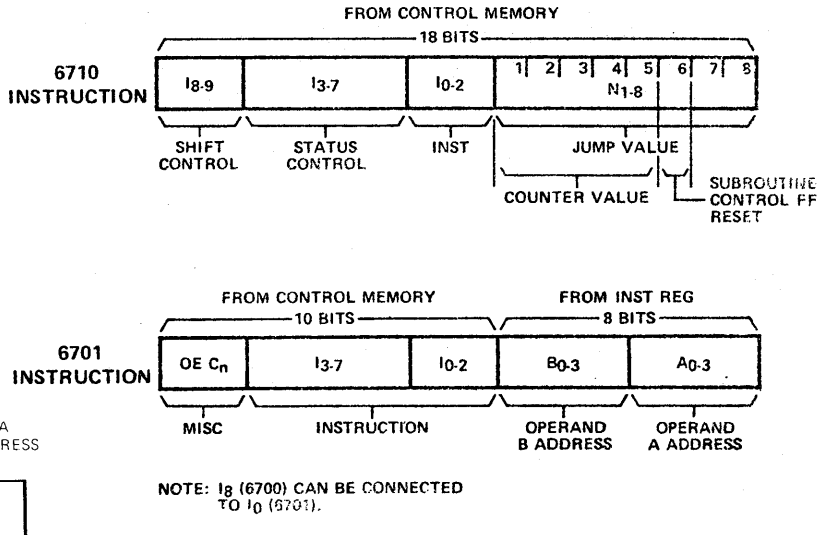


Fig. 5

5710/6710 – 5701/6701 INTERCONNECTIONS

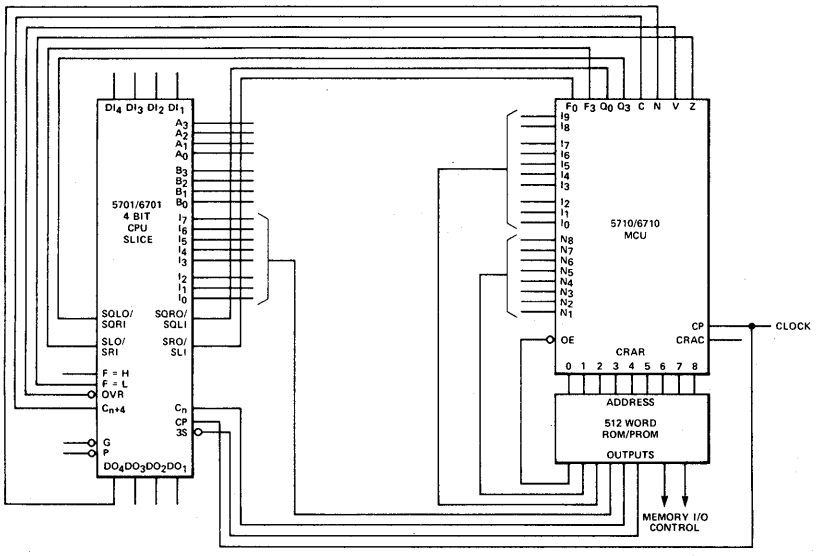


Fig. 6

MCU CONTROL OPTIONS

CONTROL CODE			ADDRESS FIELD DESTINATION	CONTROL ACTION
I_2	I_1	I_0		
0	0	0	NONE	CONTINUE TO NEXT μINSTRUCTION
0	0	1	CONTROL COUNTER	CONTINUE TO NEXT μINSTRUCTION
0	1	0	NONE/CRAR (COND. JUMP)	JUMP TO NEXT μINSTRUCTION IF CONTROL COUNTER ≠ 0, DECREMENT CONTROL COUNTER
0	1	1	NONE/CRAR (COND. SUBR. JUMP)	SUBROUTINE JUMP TO NEXT μINSTRUCTION IF CONTROL COUNTER ≠ 0, DECREMENT CONTROL COUNTER
1*	0	0	NONE	RETURN FROM SUBROUTINE
1**	0	1	CRAR (JUMP SUBROUTINE)	JUMP TO NEXT μINSTRUCTION RETURN FROM SUBROUTINE WHEN SUBROUTINE CONTROL COUNTER LATCH = $CRAR_{0-4}$
1	1	0	CRAR (JUMP)	JUMP TO NEXT μINSTRUCTION
1	1	1	CRAR (JUMP SUBROUTINE)	SUBROUTINE JUMP TO NEXT μINSTRUCTION

* THE MACHINE WILL ONLY RETURN TO THE CALLING PROGRAM IF IT ENTERED A SUBROUTINE VIA A SUBROUTINE JUMP INSTRUCTION OTHERWISE IT WILL CONTINUE TO THE NEXT μINSTRUCTION IN SEQUENCE.

** THIS OPERATION ALLOWS THE CONTROLLER TO BRANCH TO A SECTION OF CODE, PERFORM THE OPERATIONS OUTLINED BY THE CODE AND RETURN AFTER A PREPROGRAMMED CROM ADDRESS HAS BEEN REACHED OR IF A RETURN IS ENCOUNTERED.

Fig. 7

FLAG STATUS LOGIC

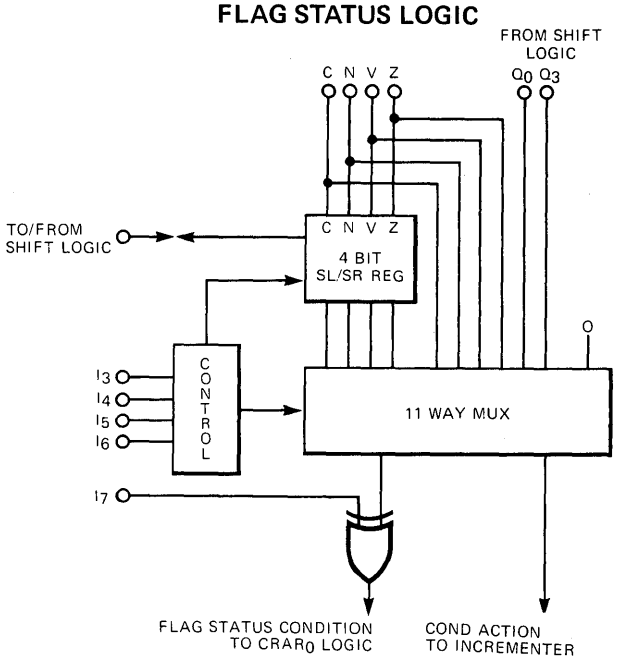


Fig. 8

MCU FLAG STATUS CONTROL OPTIONS

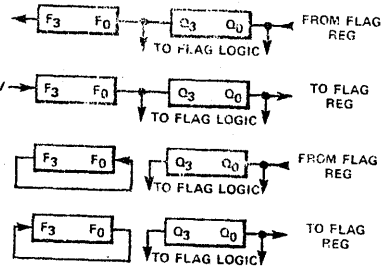
	I_7	0	0	0	0	1	1	1	1
	I_6	0	0	1	1	0	0	1	1
I_4	I_5	0	1	0	1	0	1	0	1
0	0	'0'	'0' SHIFT LEFT	SC	C	'1'	'1' SHIFT LEFT	$S\bar{C}$	$\bar{C}$
0	1	'0' STORE C	'0' SHIFT RIGHT	SN	N	'1' STORE C	'1' SHIFT RIGHT	$S\bar{N}$	$\bar{N}$
1	0	'0' STORE N, V, Z	Q_3	SV	V	'1' STORE N, V, Z	Q_3	$S\bar{V}$	$\bar{V}$
1	1	'0' STORE C, N, V, Z	Q_0	SZ	Z	'1' STORE C, N, V, Z	Q_0	$S\bar{Z}$	$\bar{Z}$

SC = STORED CARRY ETC.

Fig. 9

SHIFT OPERATIONS

SHIFT CODE		OPERATION
I_9	I_8	
0	0	ARITHMETIC LEFT
0	1	ARITHMETIC RIGHT
1	0	LOGIC END AROUND LEFT
1	1	LOGIC END AROUND RIGHT



CONTROL CODE		SHIFTING OPERATION	SLO/SRI (F_3)	SRO/SLI (F_0)	SQLO/SQRI (Q_3)	SQRO/SQLI (Q_0)
I_9	I_8					
0	0	ARITH. LEFT SHIFT	–	Q_3	–	FLAG (SC)
0	1	ARITH. RIGHT SHIFT	N⊕V	–	F_0	–
1	0	ROTATE LEFT	–	F_3	–	FLAG (SC)
1	1	ROTATE RIGHT	F_0	–	–	–

– HIGH IMPEDANCE
SC CARRY FLIP FLOP

Fig. 10

ADDRESSES IN PAIRS EVEN A,0 ETC
ODD A,1 ETC

INCREMENT OCCURS IF PRESENT ADDRESS ODD OR CONDITION

INSTRUCTION	FLAG STATUS	PRESENT ADDRESS	NEXT ADDRESS	COMMENTS
CONTINUE	UNC '0'	A,0	A,0	LOCKED LOOP
	UNC '1'	A,0	A,1	NEXT CONS ADDRESS
	UNC '0'	A,1	A+1,0	NEXT CONS ADDRESS
	UNC '1'	A,1	A+1,1	SKIP OVER A+1,0
	COND	A,0	A+1,COND	SKIP OVER A,1
	COND	A,1	A+1,COND	NEXT ADDRESS PAIR
JUMP	UNC '0'	A,X	J,0	JUMP TO J,0
	UNC '1'	A,X	J,1	JUMP TO J,1
	COND	A,X	J,COND	JUMP TO ADDRESS PAIR
RETURN	UNC '0'	A,X	R,0 R+1,0	RETURN ADDRESS DEPENDS ON STATE OF INCR CONDITION AT SUBROUTINE ENTRY
	UNC '1'	A,X	R,1 R+1,1	"
	COND	A,X	R,COND R+1,COND	"

X = '0' OR '1'

MCU NEXT ADDRESS TABLE

Fig. 11

CROM ADDRESS	CONDITION	MCU ACTION	6701 ACTION
A-1,1		CONTINUE TO A,N	FINISH PREVIOUS OPERATION
A,0	N = 0	CONTINUE TO A+1,V	
A,1	N = 1	CONTINUE TO A+1,$\overline{V}$	
A+1,0	V = 0	JUMP TO A+2,0	
A+1,1	V = 1	CONTINUE TO A+2,0	ADD R TO R+1

4 WORDS OF CONTROL ROM – 2 STEPS

NOTE AN N VARIABLE EXCLUSIVE OR TAKES ONLY 2N WORDS OF CONTROL ROM AND N STEPS. EVERY 2 VARIABLE FUNCTION CAN BE PERFORMED WITH JUST 4 WORDS AND 2 STEPS.

OPERATION

Add contents of R to the contents of R+1 if N⊕V is true.

Fig. 12

EXAMPLE: IF $N\overline{V} + VC + \overline{N}V\overline{C}Z$ IS TRUE (T) ADD C(R) TO C(R+1), IF FALSE (F) DO NOTHING.

STRAIGHTFORWARD SOLUTION
EVERY CONDITION INSIDE DOTTED AREA CAN BE REMOVED.

6 BLOCKS TO GIVE 12 WORDS CROM.
MAXIMUM OF 4 STEPS.

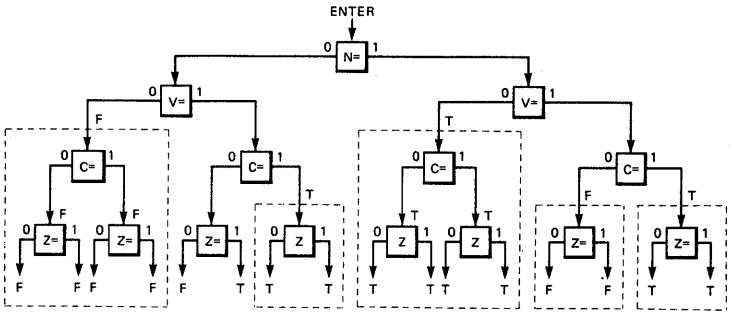


Fig. 13

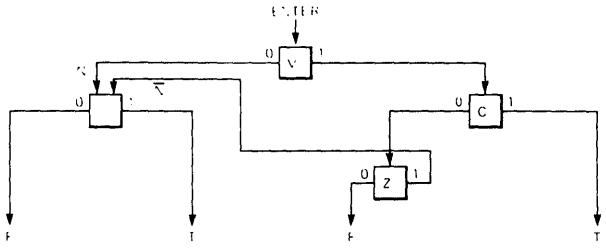
OPERATION – ADD CONTENTS OF R TO THE CONTENTS OF R+1 IF $N\overline{V} + VC + \overline{N}V\overline{C}Z$ IS TRUE.

CROM ADDRESS	CONDITION	MCU ACTION	6701 ACTION
A-1,1		CONTINUE TO A,N	FINISH PREVIOUS OPERATION
A,0	N = 0	CONTINUE TO A+1,V	
A,1	N = 1	JUMP TO A+4,V	
A+1,0	V = 0	JUMP TO A+6,0	
A+1,1	V = 1	CONTINUE TO A+2,C	
A+2,0	C = 0	CONTINUE TO A+3,Z	
A+2,1	C = 1	JUMP TO A+6,0	ADD R TO R+1
A+3,0	Z = 0	JUMP TO A+6,0	
A+3,1	Z = 1	JUMP TO A+6,0	ADD R TO R+1
A+4,0	V = 0	JUMP TO A+6,0	ADD R TO R+1
A+4,1	V = 1	CONTINUE TO A+5,C	
A+5,0	C = 0	JUMP TO A+6,0	
A+5,1	C = 1	CONTINUE TO A+6,0	ADD R TO R+1
A+6,0			

12 WORDS OF CONTROL ROM – ONLY 4 STEPS MAXIMUM

Fig. 14

A MINIMUM SOLUTION



CROM ADDRESS	CONDITION	MCU ACTION	6701 ACTION
A-1,1		CONTINUE TO A,V	FINISH PREVIOUS OPERATION
A,0	V 0	CONTINUE TO A+1,N	
A,1	V 1	JUMP TO A+2,C	
A+1,0	N 0	JUMP TO A+4,0	
A+1,1	N 1	JUMP TO A+4,0	ADD R TO R+1
A+2,0	C 0	CONTINUE TO A+3,Z	
A+2,1	C 1	JUMP TO A+4,0	ADD R TO R+1
A+3,0	Z 0	JUMP TO A+4,0	
A+3,1	Z 1	JUMP TO A+1,N	
A+4,0			

8 WORDS OF CONTROL ROM ONLY 4 STEPS MAXIMUM

Fig. 15

OPERATION – 16 BIT 2s COMPLEMENT MULTIPLY

MULTIPLY THE CONTENTS OF R BY THE CONTENTS OF R+1 TO FORM A DOUBLE LENGTH PRODUCT WITH THE MOST SIGNIFICANT HALF IN R+1 AND THE LEAST SIGNIFICANT IN R.

CROM ADDRESS	CONDITION	MCU ACTION	6701 ACTION
A-1,1		CONTINUE TO A,0	FINISH PREVIOUS OPERATION
A,0		CONTINUE TO A,1	R+1 TO Q
A,1		SET CONTR. CNTR. TO 14 CONTINUE TO A+1,Q_0	CLEAR R+1, SHIFT ARITH RIGHT R+1,Q
A+1,0	$Q_0 = 0$	JUMP TO A+1,Q_0 IF CC≠0 DECREMENT CC.	SHIFT ARITH RIGHT R+1,Q
A+1,1	$Q_0 = 1$	JUMP TO A+1,Q_0 IF CC≠0 DECREMENT CC.	ADD R TO R+1, SHIFT ARITH RIGHT RESULT,Q
A+2,0	$Q_0 = 0$	JUMP TO A+3,0	SHIFT ARITH RIGHT R+1,Q
A+2,1	$Q_0 = 1$	JUMP TO A+3,0	SUBTRACT R FROM R+1 SHIFT ARITH RIGHT RESULT,Q
A+3,0		CONTINUE TO A+3,1	Q TO R

7 WORDS OF CONTROL ROM – 19 STEPS

UNSIGNED MULTIPLY WOULD TAKE ONLY 5 WORDS OF CONTROL ROM AND 19 STEPS.

Fig. 16

9. USE OF THE MOSTEK F8 MICROCOMPUTER AS A SOFTWARE UART

R. L. BALDRIDGE AND D. LINDSAY
APPLICATIONS ENGINEERS
MOSTEK CORPORATION
CARROLLTON, TEXAS

The Mostek F8 I.C. family permits the design of a complete microcomputer system using only two chips, the MK3850 Central Processing Unit (CPU) and the MK3851 Program Storage Unit (PSU). This paper describes an actual application of these chips in the development of a full duplex software UART for Teletype I/O. This software UART forms an integral part of Designer's Debugging Tool I (DDT-1), a software package developed as a powerful design aid for the Mostek F8 Survival Kit. A complete listing of DDT-1, as well as a schematic of the F8 Survival Kit are given in the Appendix.

A block diagram of the MK3850 CPU is shown in Figure 1. The outstanding features of this chip are:

- N-channel Isoplanar MOS Technology
- 64 bytes of scratchpad RAM
- On-chip clock and power-on-reset circuitry
- Two 8-bit bidirectional I/O ports
- Vectored interrupt capability
- 2 MH_Z clock rate (2μs cycle time)
- Over 70 instructions (most 1 cycle)
- Low power dissipation, less than 330 mW
- Available in a low cost plastic package

A programming model for the MK3850 is shown as Fig. 2.

The MK3851 PSU is shown in Fig. 3. Its main features are

- 1024 bytes of ROM
- On-chip programmable timer
- Two 8-bit bidirectional I/O ports
- Vector interrupt address
- 16-bit program counter with stack
- 2 μs cycle time
- low power dissipation, less than 275 mW
- available in low cost plastic package

These two chips together therefore form a complete microcomputer with 1K of memory, 32 bits of I/O, a programmable clock, vectored interrupt capability, and 64 bytes of scratchpad memory. The only external circuitry required to make the system work is a 2 MH_Z crystal, two 10pf capacitors, and a +5V, +12V power supply.

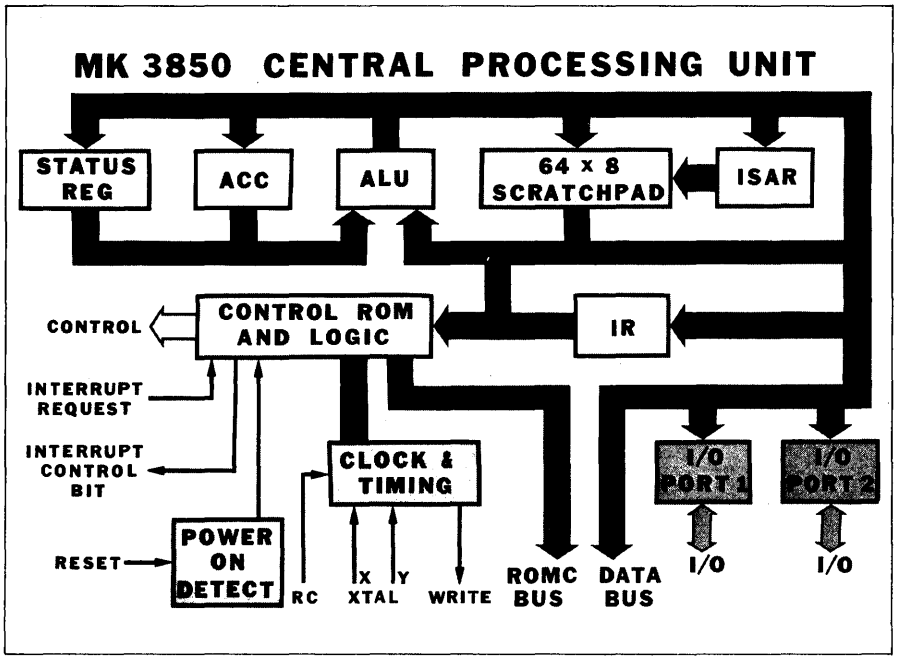


Fig. 1 MK3850 CPU

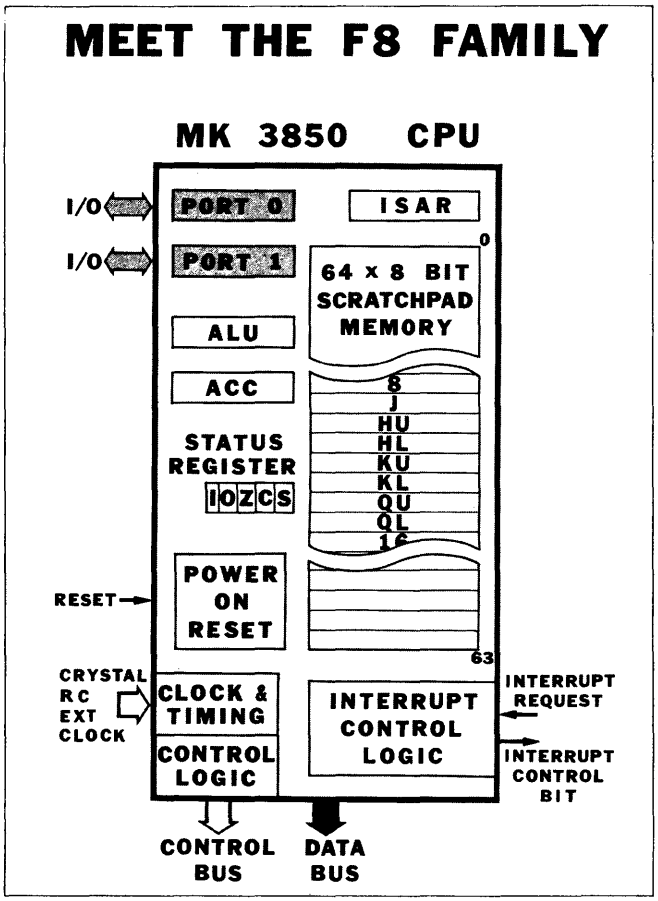


Fig. 2 MK3850 Programming Model

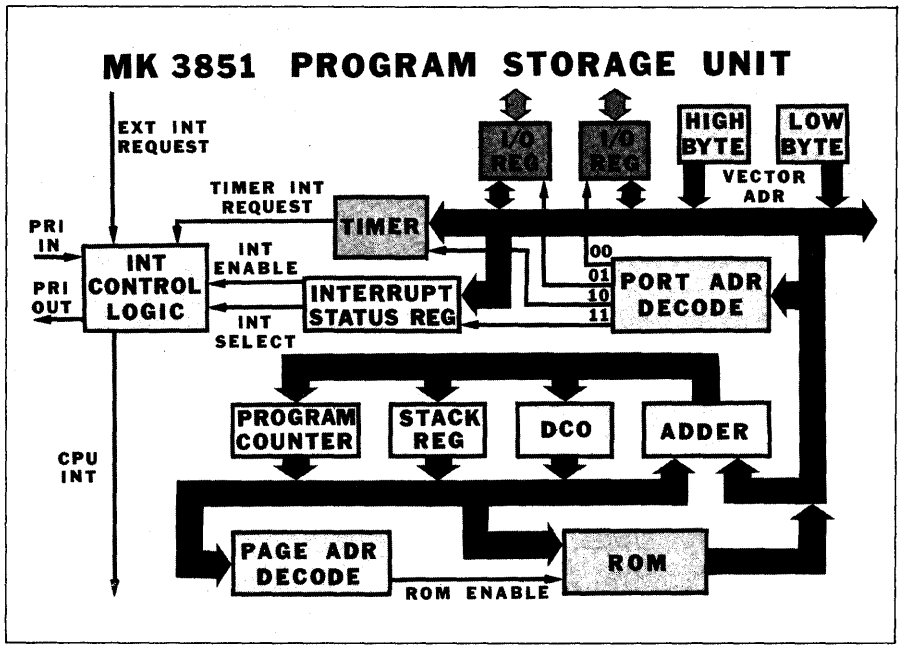


Fig. 3 Program Storage Unit (PSU)

The next section describes the use of these chips as a software UART and also presents some design considerations for general applications.

THE SOFTWARE UART

This design example illustrates some of the outstanding features of the Mostek F8 2-chip microcomputer, in particular the use of the programmable clock, the bidirectional ports, the vectored interrupt, and the scratchpad memory.

Problem: Design a software UART for full duplex teletype I/O.

Systems Analysis

I/O Requirements:

Data width - 1 bit in, 1 bit out
I/O rates - 110 baud in and out
Peak I/O rates - 300 baud
Timing required - send bit time, receive sample bit time

Data storage requirements:

OB - output buffer
OC - Output counter
IB - input buffer
IC - input counter
AS - accumulator save register
J - status save register

The data storage requirements are thus well within the 64 byte scratchpad capability.

Design

Fig. 4 shows the timing for 110 baud asynchronous ASCII with 1 start bit and 2 stop bits. The sample bit times and send bit times are shown relative to the programmable clock which divides each bit interval into 8 timing slots. When sending an ASCII character the output counter is initialized to '57' (HEX) when the start bit is sent, thereby establishing bit times every 8 counts, or when the last 3 bits of the counter are all ones. When the counter counts down to 'FF' (HEX) the output is finished. Similarly, on input when a start bit is detected the input counter is set to '49" (HEX) and sample bit times also occur every 8 counts of the clock.

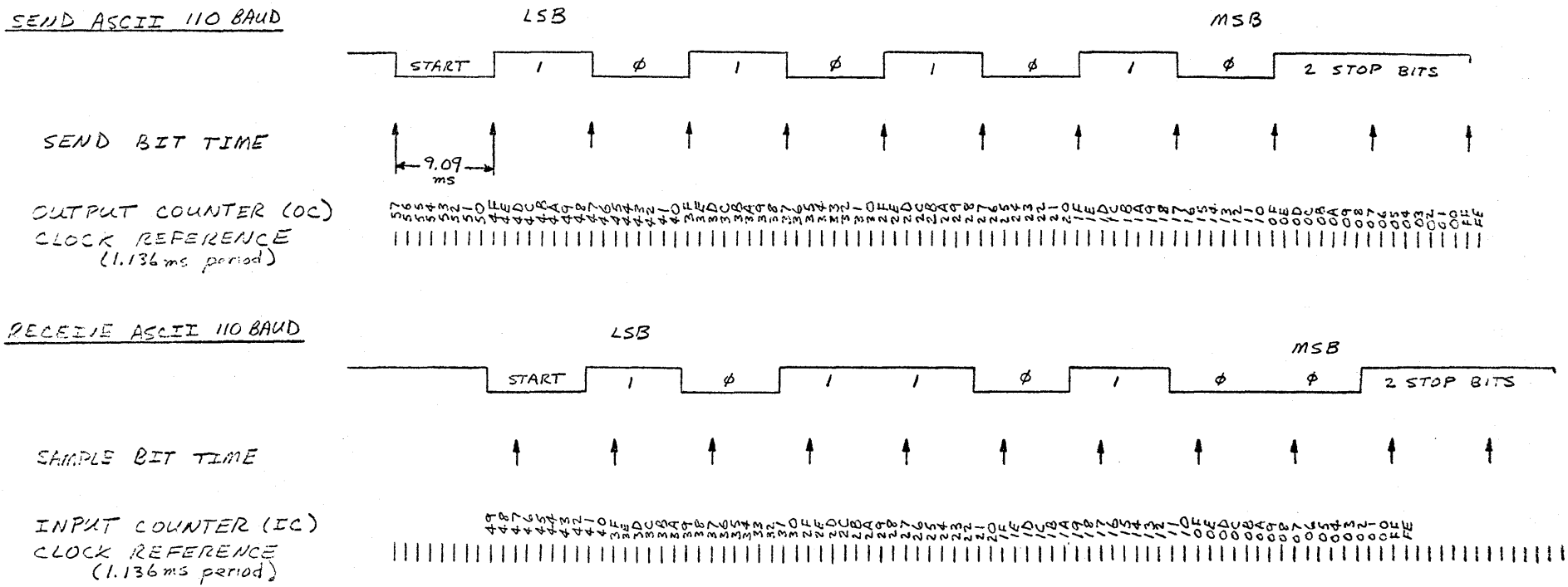


FIGURE 4 - TIMING FOR TELETYPE I/O

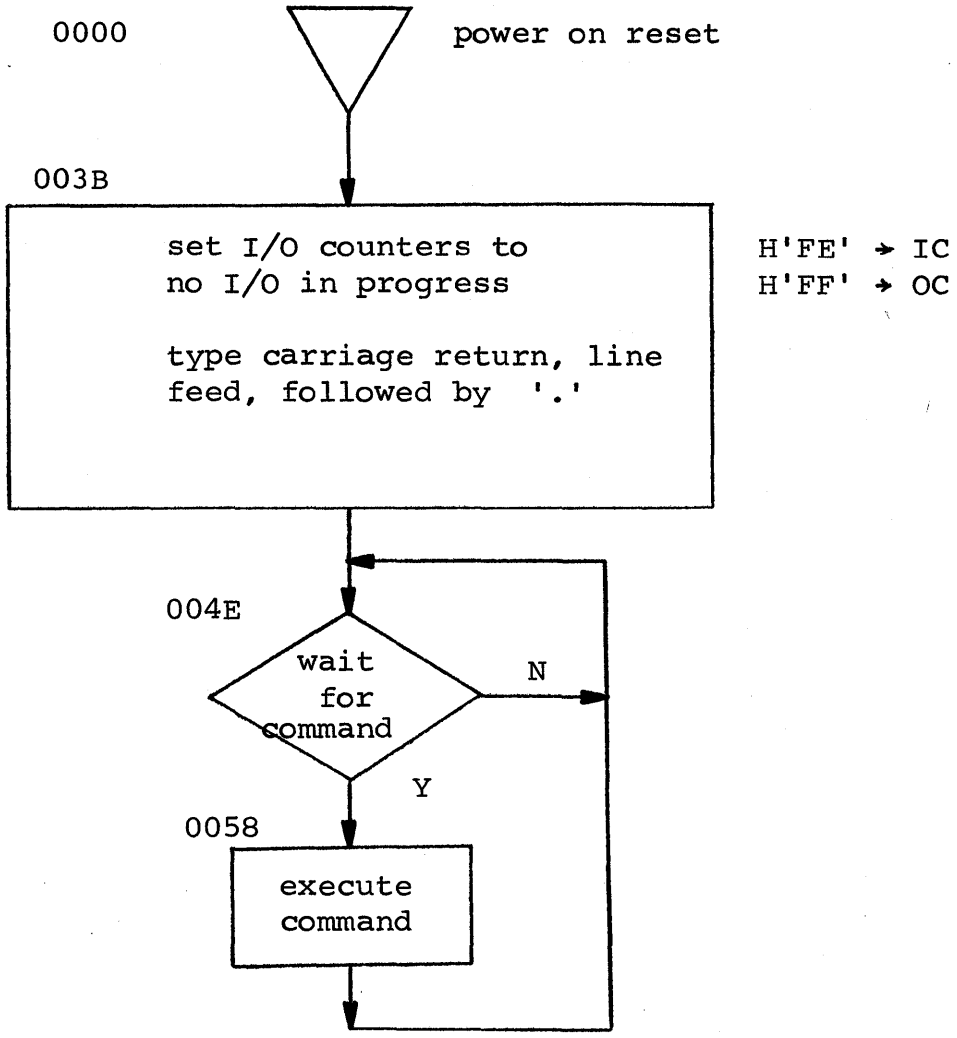
The flowcharts for accomplishing this task are shown in Fig. 5 and are keyed to the DDT-1 listing in the Appendix.

SOFTWARE UART DESCRIPTION

Fig. 5 (a) shows the power on initialization required for operation of the software UART. The input count and output count registers are initialized for "No input in progress", and "No output in progress" respectively.

To output a character a call is made to subroutine TC shown in Fig. 5(b). After saving the return address, a check is made to determine whether an output is already in progress by examining the output count register (OC) for a positive value. When any existing output has finished the character to be sent is moved from the appropriate scratchpad register into the output buffer (OB). The output count is initialized to Hex '57' and a call is made to the timer interrupt routine (TI) to output the start bit, start the timer, and enable interrupts. (Refer to the state of the output counter on the I/O timing diagram of Fig. 4) To input a character a call is made to subroutine KC (Key-in a character) shown in Fig. 5 (c). This routine saves the return address and waits for the input counter to be counted down to Hex 'FF' by timer interrupts. When the input finishes the input counter is set to Hex 'FE' - "ready for another input" and the assembled ASCII character is moved from the input buffer (IB) to the user's scratchpad register.

The timer and keyboard interrupts are handled by routine TI shown in Figs. 5 (d), (e) and (f). This routine is responsible for loading the proper count into the programmable clock, detecting the start bit of an input, and sending the start bit of an output. It also determines the I/O bit times, or when to send the next bit on output or sample the next bit on input. On output it shifts each bit of the input buffer out to the TTY port at the bit times shown on Fig. 4. On input it samples and shifts each bit into the input buffer at the bit times also shown in Fig. 4. I/O ceases when the appropriate counter counts (IC or OC) down to a negative value (Hex 'FF'). In this way simultaneous input and output may take place on two different bit positions of the same bidirectional I/O port. The schematic for the Mostek F8 Survival Kit in the Appendix shows Port 5 being used for this purpose.



DDT-1 commands

B - Breakpoint address
C - Copy memory arrays
D - Dump memory to tape
E - Execute at specified address
L - Load memory from tape
M - Memory content display and modify
P - Display ports and modify
T - Type memory content array
H - Hexadecimal arithmetic

Fig. 5 (a) - Power on reset

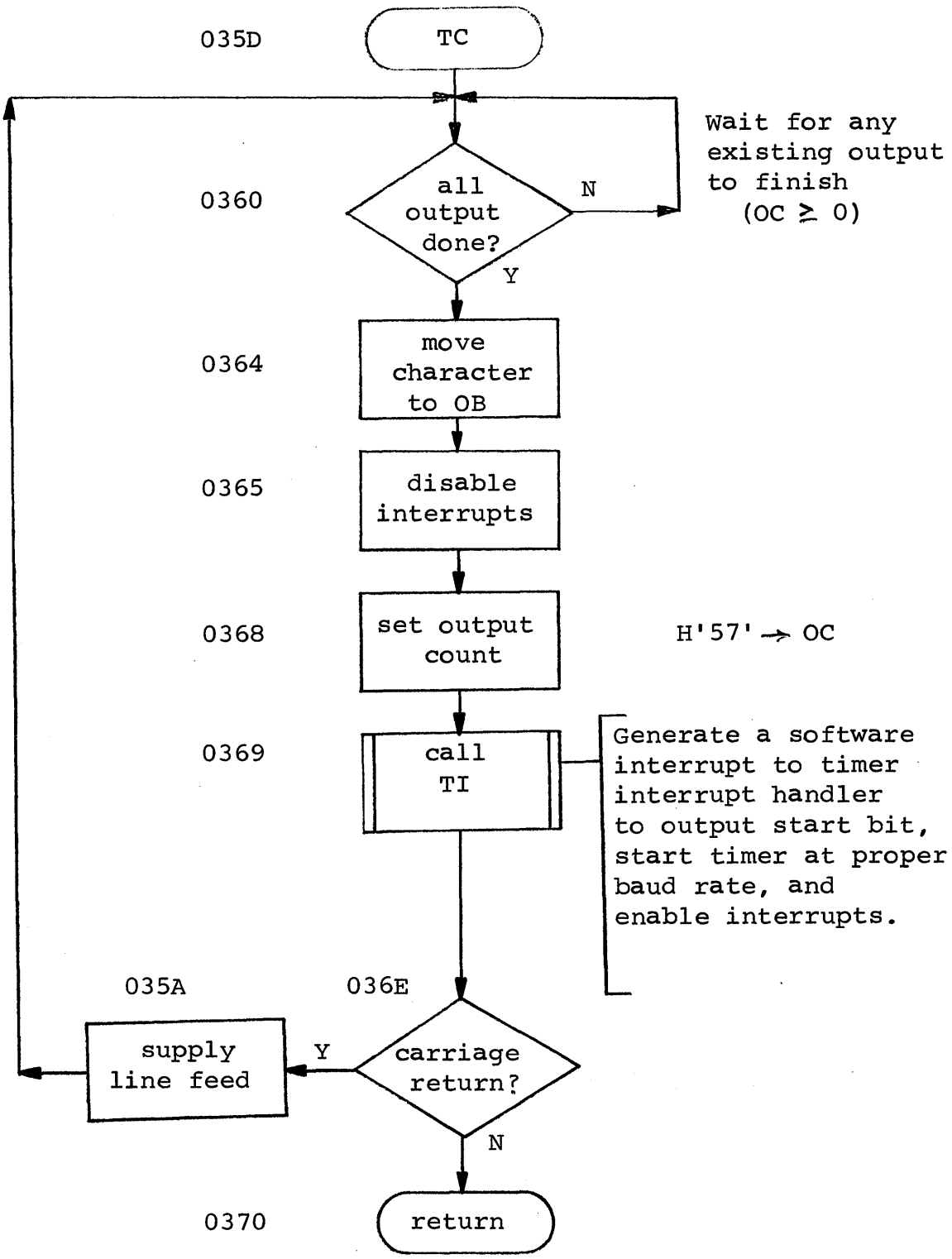


Fig. 5 (b) - Type A character

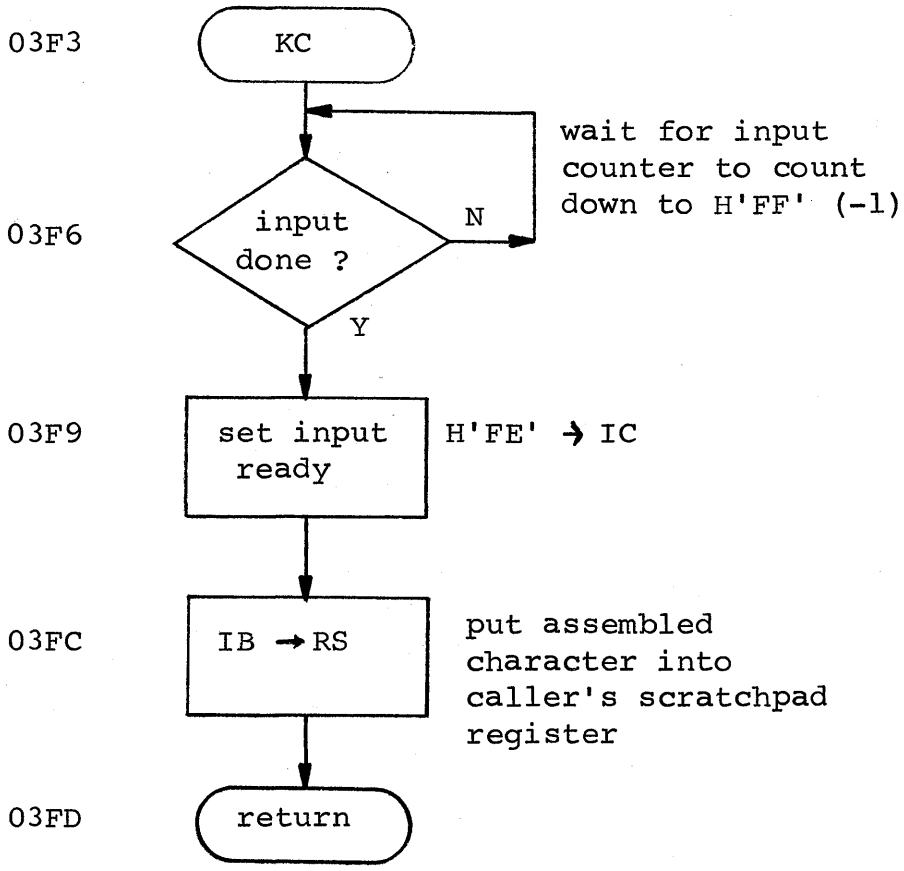


Fig. 5 (c) - Input A Character

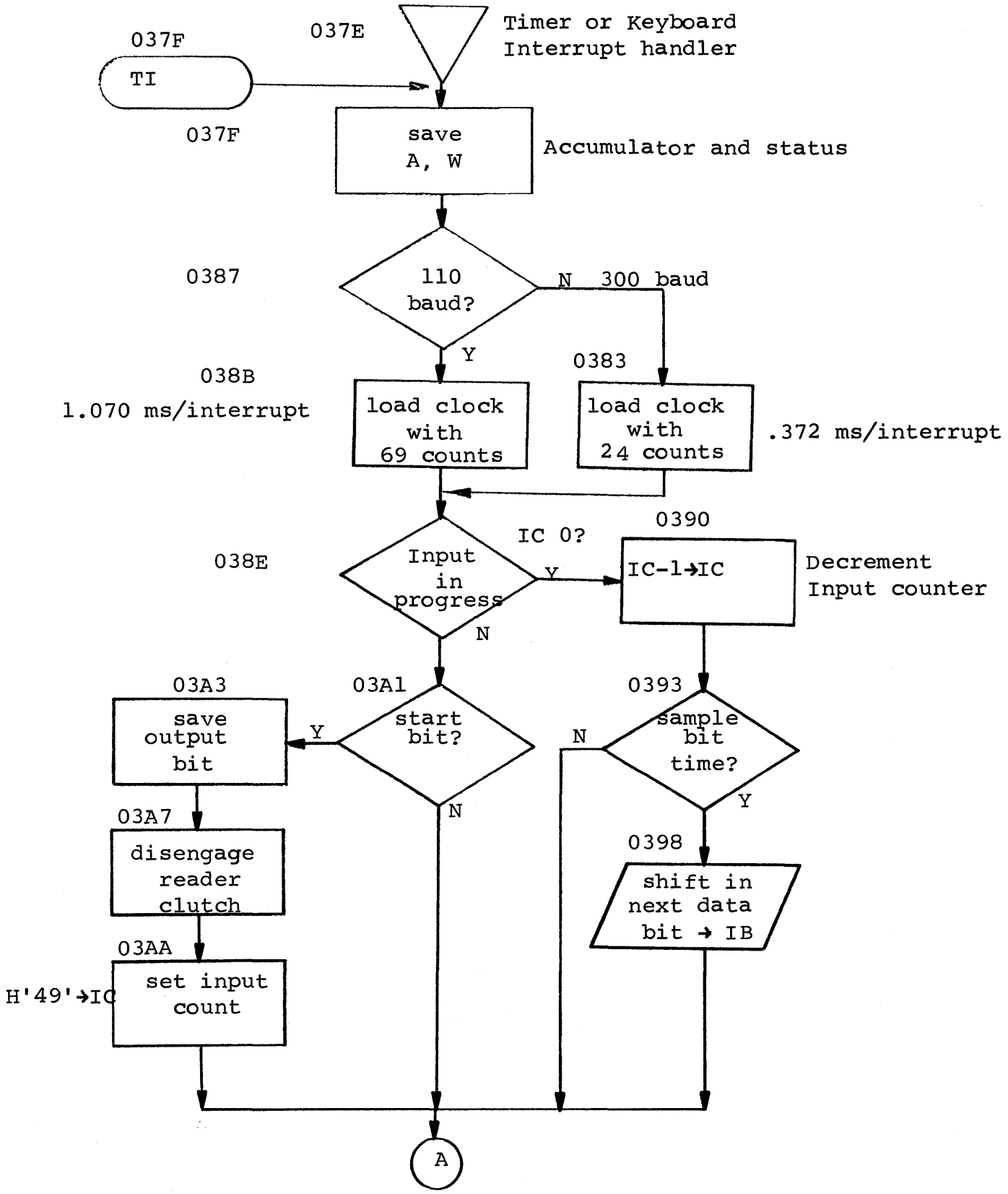


Fig. 5 (d) - Timer interrupt handler

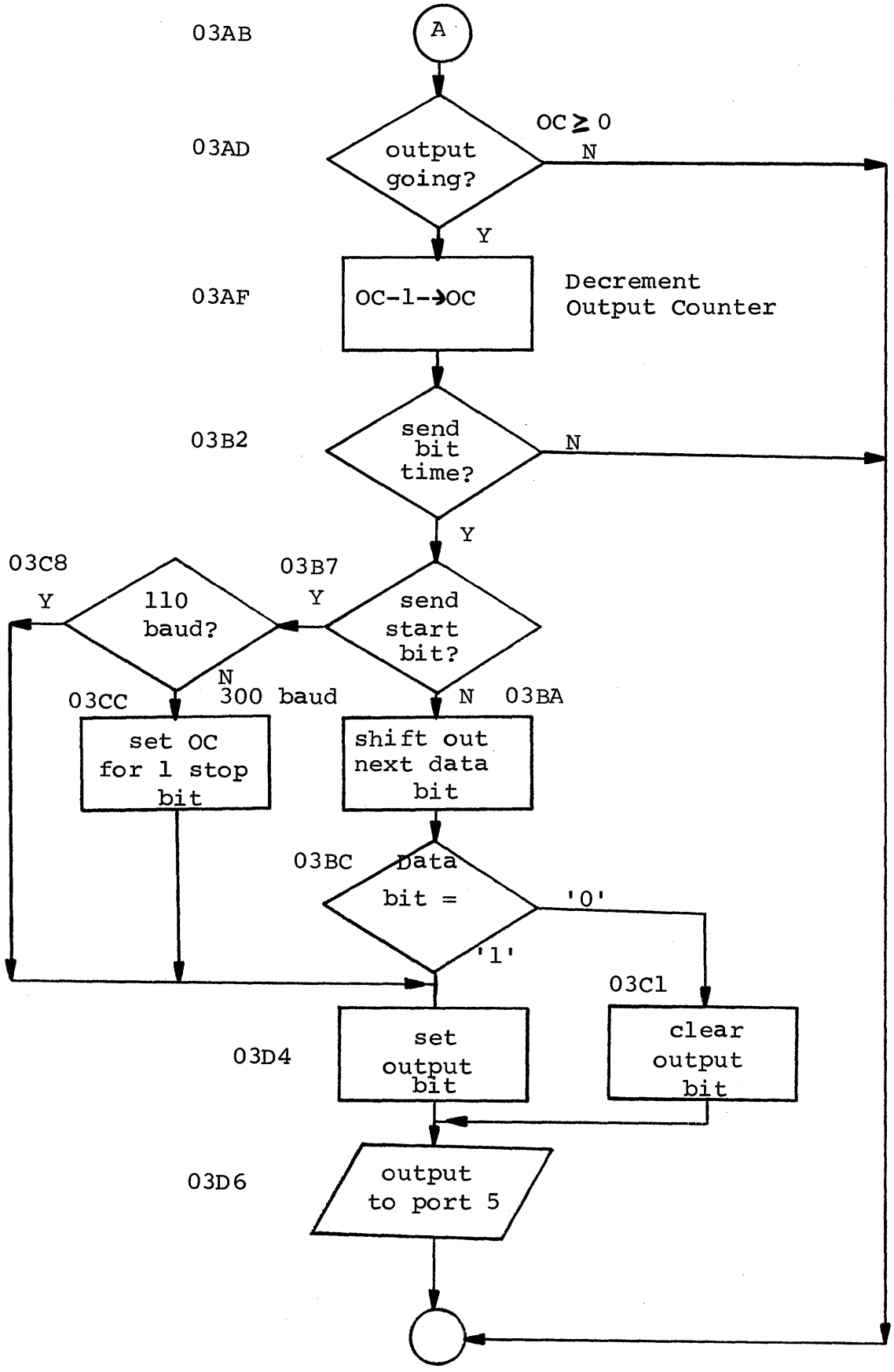


Fig. 5 (e) Timer Interrupt Handler Cont.

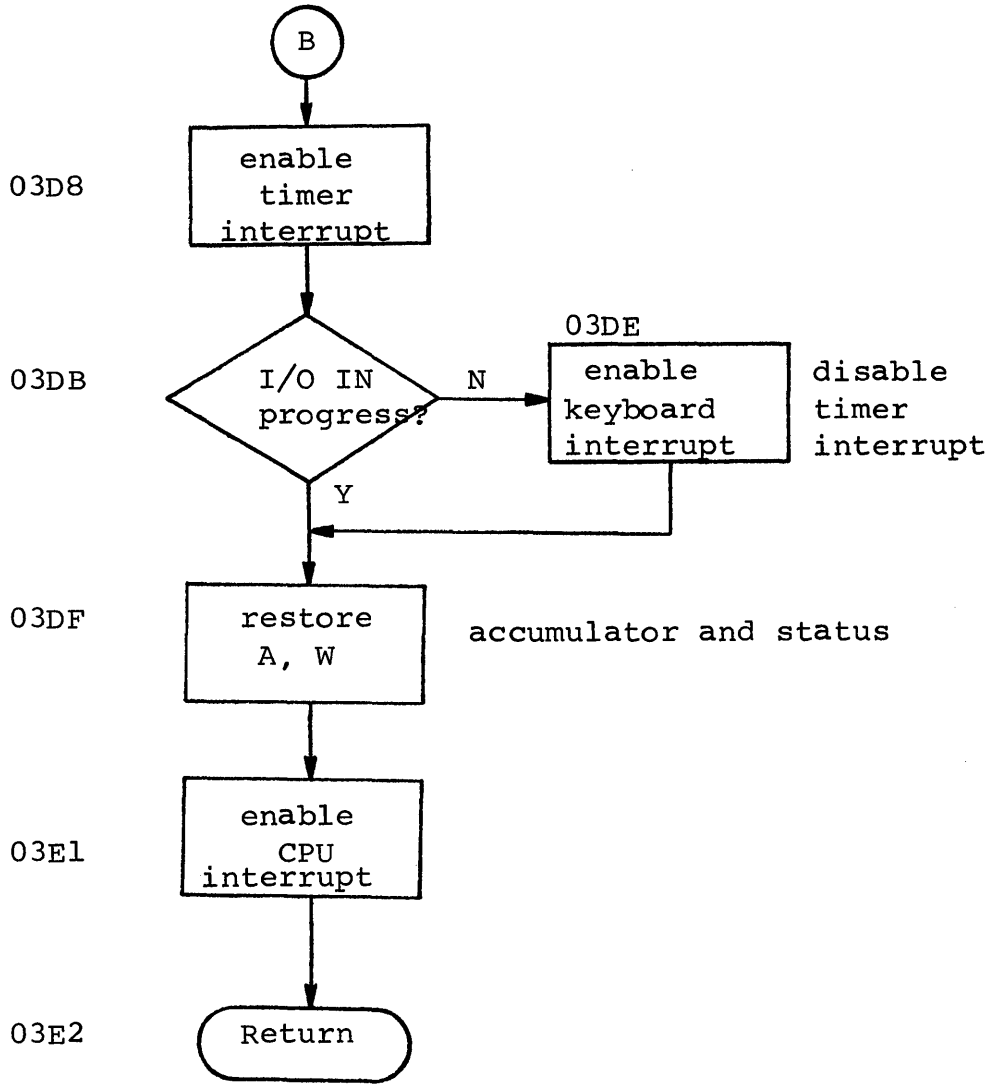


Fig. 5 (f) - Exit from interrupt handler

How To Calculate The Clock Count

In order to obtain the proper clock constant to provide for system timing is is necessary to take into account:

1. Clock countdown time before interrupt.

2. Interrupt latency time - or the time required for the CPU to finish its current instruction before acknowledging the interrupt.

3. Interrupt cycle time - the time required for the CPU to acknowledge the interrupt and transfer control to the interrupt service routine.

4. Program overhead - the time spent to save registers, status,

Fig. 6 depicts the timing for a single clock cycle, showing the hardware and software times involved.

These times may be computed as follows for the Mostek F8 software UART:

Interrupt latency time

= Time required to finish a 2 cycle instruction

$$= 1 \text{ instruction} \times 2 \frac{\text{cycles}}{\text{instruction}} \times 2 \frac{\mu s}{\text{cycle}}$$

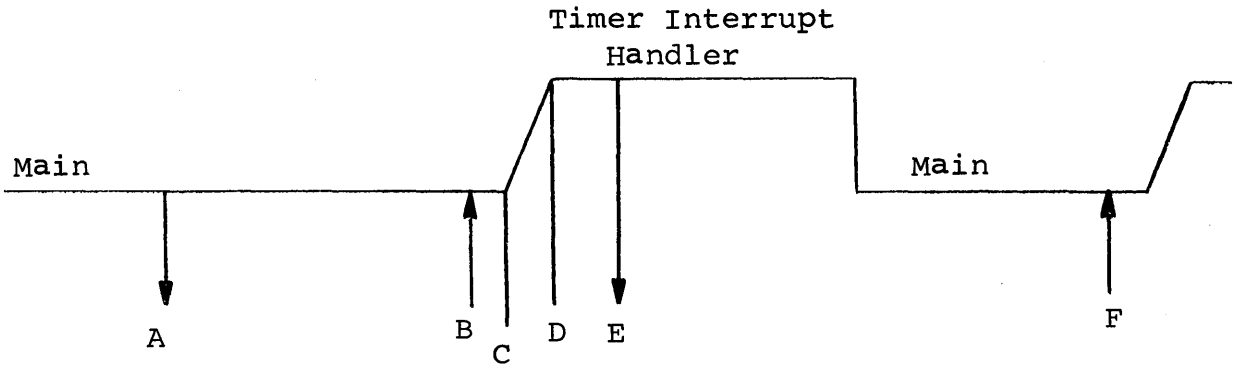
$$= 4 \mu s$$

Note: The interrupt latency time may vary depending on the point at which the interrupt occurs during an instruction, how many cycles the instruction requires, and whether the instruction is privileged (i.e. the next instruction must be executed before acknowledging the interrupt). For the Mostek F8 this can be a variation of 0 to 25 μs. However, many of the instructions are single cycle, hence the above figure of 2 cycles is intended to be a representative average.

Interrupt Cycle Time

Upon recognizing an interrupt, the Mostek F8 goes through a 5 cycle sequence in order to obtain the vectored interrupt address. This gives

$$5 \text{ cycles} \times 2 \frac{\mu s}{\text{cycle}} = 10 \mu s$$



EVENTS

A - Program loads clock counter

B - Clock requests CPU interrupt

C - CPU acknowledges interrupt

D - Interrupt service routine begins

E - Interrupt service routine reloads clock counter to begin next cycle

F - Clock interrupts again etc.

TIME INTERVALS (explained in text)

T_{AB} - Clock countdown time (Hardware)

T_{BC} - Interrupt latency time (Hardware)

T_{CD} - Interrupt cycle time (Hardware)

T_{DE} - Program overhead (Software)

Fig. 6 - Timing for one clock cycle

Program Overhead

Referring to the DDT-1 program listing in the Appendix at location 037E(Hex) it is seen that 10 instructions are executed before the clock counter is loaded again. The cycles required for these instructions may be added

Instruction	Cycles
NOP	1
LR	1
LR	1
LI	2.5
OUTS	4
INS	4
NI	2.5
BNZ	3.5
LI	2.5
OUTS	4
	26.0 Cycles

and the program overhead time computed as:

$$26 \text{ cycles} \times 2 \frac{\mu s}{\text{cycle}} = 52 \mu s$$

Hence the clock countdown time required to obtain the proper timing for the Teletype I/O becomes:

$$CDT = \frac{1 \text{ I/O BIT PERIOD}}{8 \text{ clock periods}} - (\text{Interrupt Latency Time} + \text{interrupt cycle time} + \text{program overhead})$$

For the software UART example this gives:

$$\text{Clock countdown time} = \frac{9.09ms}{8} - (4\mu s + 10\mu s + 52\mu s)$$

$$= 1.136ms - .066ms = 1.070 \text{ ms}$$

The clock countdown time is a function of the system clock on the Mostek F8 as follows:

CDT = (System clock period)x(clock count)x(31)

The clock count for the example may therefore be computed as:

$$\text{Clock counts} = \frac{1.070 \text{ ms}}{500 \text{ ms} \times 31} = 69$$

Hence to establish the proper timing a count of 69 is loaded into the programmable clock of the PSU.

OTHER APPLICATIONS

The Mostek F8 I.C. family may be designed into a wide variety of applications. For example, a motor controller may be implemented with 2 chips as shown in Fig. 7. Consistent timing may be obtained by using the on-chip programmable clock as a reference. An optical tachometer generates a symmetrical pulse train related to motor speed, the microcomputer computes the speed and compares it to a desired reference, then generates an error signal to speed up or slow down the motor as required. Another example is the cash register system shown in Fig. 8. Here the Mostek F8 scans a keyboard, controls a printer, and also drives a display. These functions may be multiplexed through the bidirectional I/O ports by using the programmable timers and the external interrupts provided on the PSUs. This example also demonstrates how the basic 2-chip system may be expanded upward in complexity as the application grows using other I.C. chips in the Mostek F8 family. Fig. 8 shows an additional PSU added to the F8 bus to control the printer; Fig. 9 shows additional I/O ports added with the Mostek MK 3861 peripheral I/O (PIO) chip; Fig. 10 shows how to add additional RAM memory by using the MK 3853 Static Memory Interface (SMI); and finally, Fig. 11 shows how to add erasable programmable ROM into the F8 system.

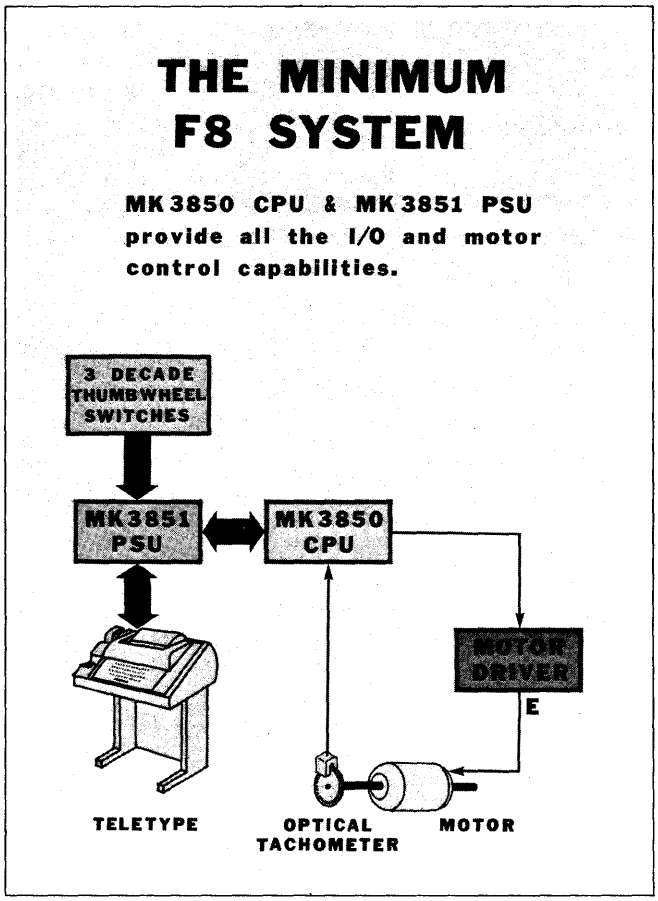


Fig. 7 - Mostek F8 as a motor controller

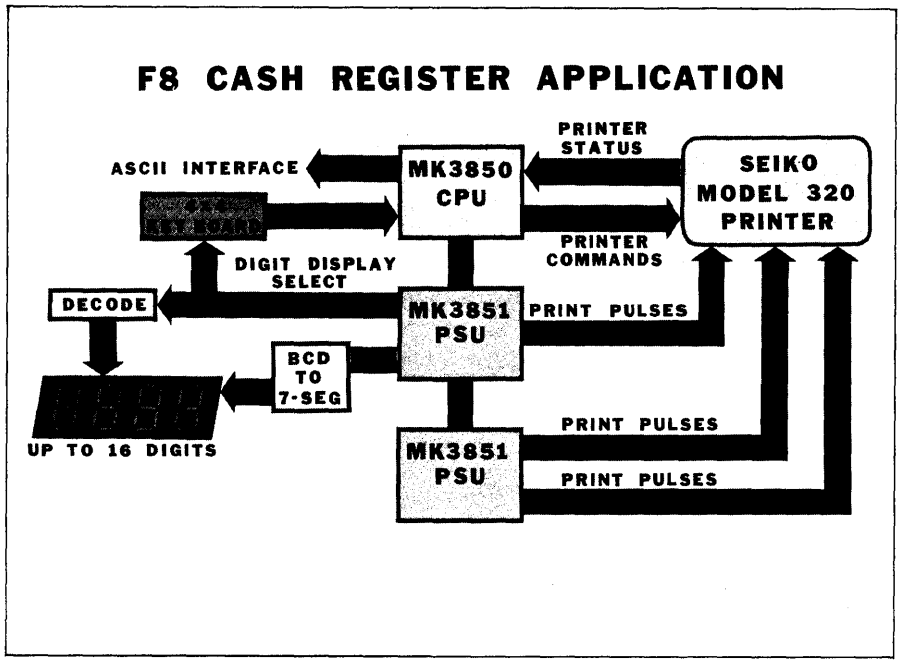


Fig. 8 - Mostek F8 application - adding additional PSUs

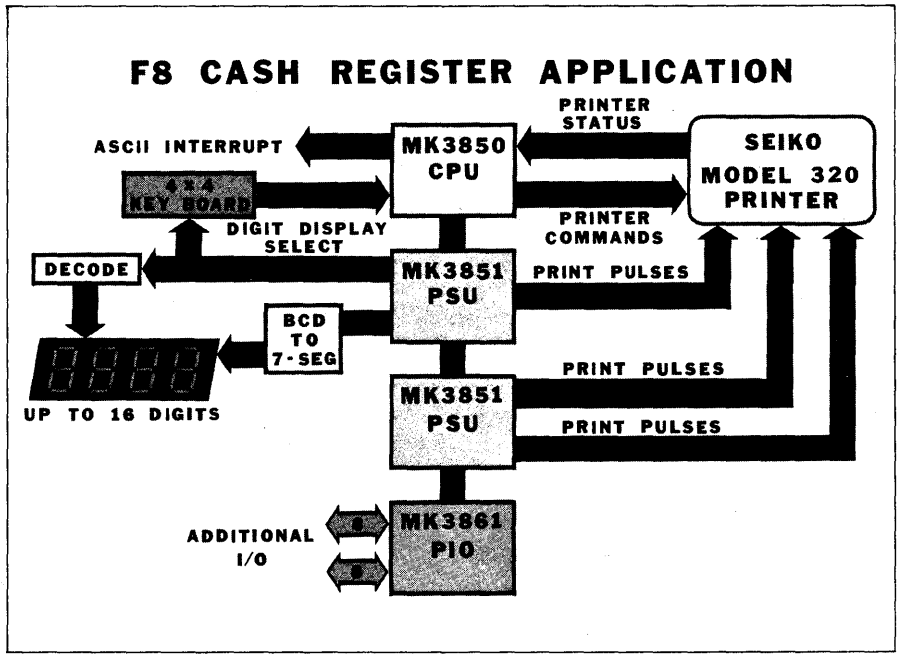


Fig. 9 - Adding more I/O

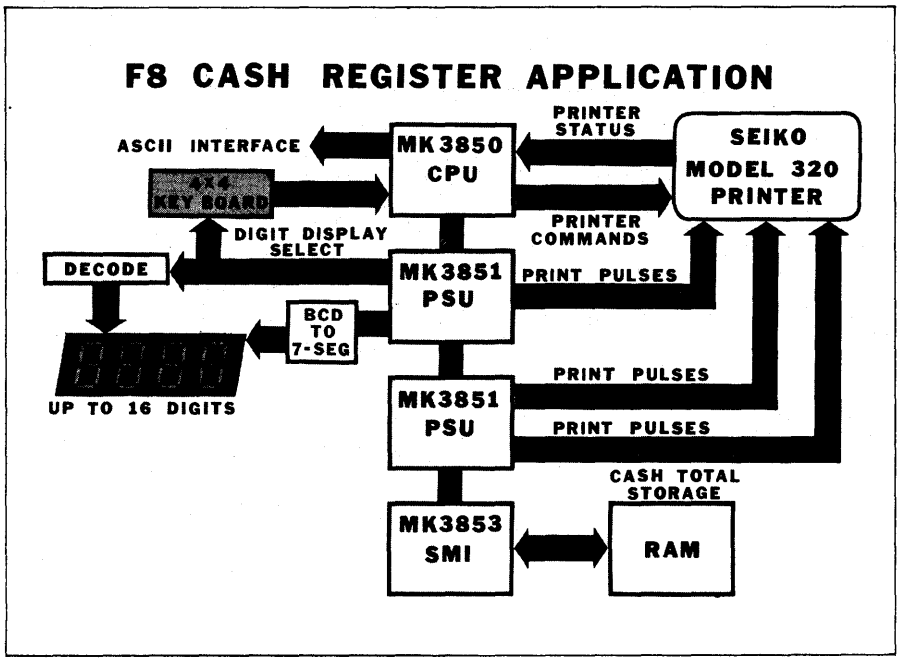


Fig. 10 - Adding more RAM

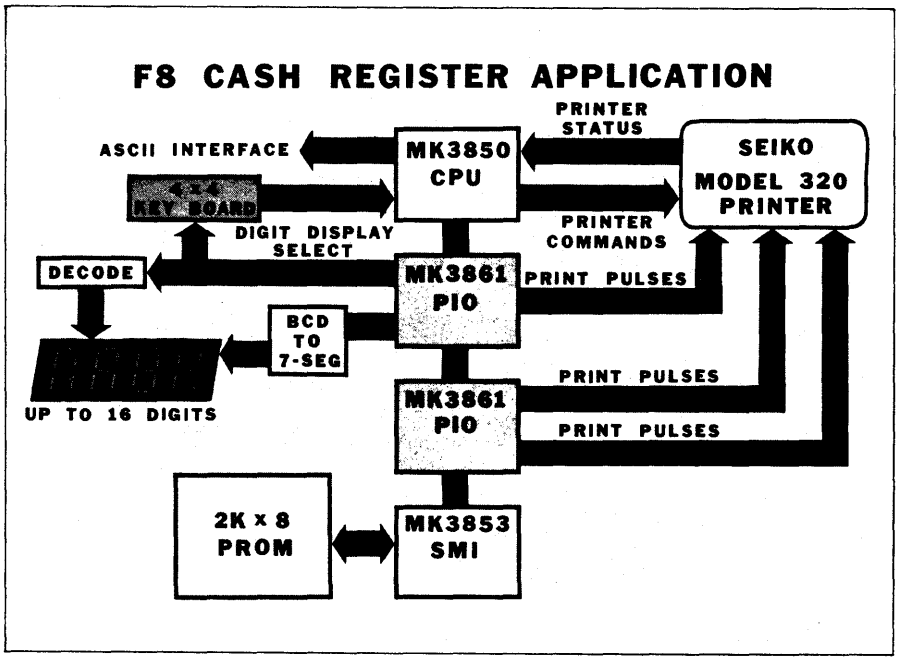


Fig. 11 - Adding PROM

```
F8X V01                              PAGE    1

                *
                * ALL SUBROUTINES USE SCRATCHPAD REGISTERS C,D (K REGISTER) TO SAVE PC1,
                * PERMITTING INTERRUPTS AT ANY TIME.  INTERRUPTS USE REGISTERS 8,9
                * (AS,J) TO SAVE ACC,W, AND REGISTERS 4,5,6,7 (IC,IB,OC,OB) TO PERFORM
                * FULL DUPLEX I/O.
                XU      EQU     H'0'    X REGISTERS
                XL      EQU     H'1'
                YU      EQU     H'2'    Y REGISTERS
                YL      EQU     H'3'
                IC      EQU     H'4'    INPUT COUNT REGISTER
                IB      EQU     H'5'    INPUT BYTE REGISTER
                OC      EQU     H'6'    OUTPUT COUNT REGISTER
                OB      EQU     H'7'    OUTPUT BYTE REGISTER
                AS      EQU     H'8'    ACC SAVE REGISTER (FOR INTERRUPT ONLY!)
                J       EQU     H'9'    J REGISTER (FOR INTERRUPT ONLY!)
                HU      EQU     H'A'    HU REGISTER
                HL      EQU     H'B'    HL REGISTER
                PT      EQU     H'5'    PORT FOR TELETYPE I/O
                PI      EQU     H'6'    PORT FOR INTERRUPT CONTROL
                PC      EQU     H'7'    PORT FOR TIMER COUNT
                TO      EQU     H'40'   TELETYPE OUTPUT CODE
                TR      EQU     H'20'   TELETYPE READER CLUTCH CODE
                TB      EQU     H'12'   TELETYPE BAUD RATE
                *
                * POWER ON

0000
                        ORG H'0000'
0000 90 3A              BR      IN
                *
                * REENTRY JUMP OFF
0002 16         E2      LM
0003 90 AD              DC      H'90AD' BRANCH TO REENTRY AT FFB1
                *
                * BREAKPOINT ENTRY
0005 03         BE      LR      A,QL    RESTORE ACC AFTER JMP
0006 0F                 LR      Q,DC
0007 2A FE BE           DCI     H'FFBE'
000A 17                 ST              STORE ACC
000B 0A                 LR      A,IS
000C 17                 ST              STORE ISAR
000D 60                 LISU    0
000E 68                 LISL    0
000F 4D                 LR      A,I
0010 17                 ST              STORE R0
0011 49                 LR      A,J     J IN ACC
0012 1E                 LR      J,W     W IN J
0013 F9                 XS      J       J/W IN ACC
0014 50                 LR      0,A     J/W IN R0
0015 F9                 XS      J       J BACK IN ACC
0016 59                 LR      J,A     RESTORE J
0017 F0                 XS      0
0018 50                 LR      0,A     SAVE W IN R0
0019 4C         BE0     LR      A,S
001A 17                 ST
```

```
001B 0A             LR    A,IS
001C 24 C1          AI    H'C1'
001E 0B             LR    IS,A
001F 94 F9          BNZ   BE0   STORE REST OF SCRATCHPAD
0021 2A FF BB       DCI   H'FFBB'
0024 08             LR    K,P
0025 00             LR    A,KU
0026 17             ST
0027 01             LR    A,KL
0028 17             ST          STORE PC1
0029 40             LR    A,0
002A 17             ST          STORE W
002B 2A FF B5       DCI   H'FFB5'
002E 16             LM
002F 06             LR    QU,A
0030 16             LM
0031 07             LR    QL,A  SET Q TO OLD BREAKPOINT
0032 70             LIS   H'0'
0033 50             LR    XU,A
0034 5C             LR    S,A   SET INCREMENT
0035 73             LIS   H'3'
0036 51             LR    XL,A  SET COUNT
0037 11             LR    H,DC  SET H TO START OF OLD CODE
0038 28 02 CA       PI    CHQ   COPY BACK INTO PROGRAM
                    *
                    * INITIALIZE
003B 20 20    IN    LI    TR
003D B5             OUTS  PT    DISENGAGE READER CLUTCH
003E 20 FF          LI    H'FF'
0040 54             LR    IC,A
0041 34             DS    IC    SET INPUT COUNT READY
0042 56             LR    OC,A  SET OUTPUT COUNT COMPLETE
0043 67       IN0   LISU  7
0044 6F             LISL  7
0045 28 03 50       PI    TC1   TYPE CARRIAGE RETURN, LINE FEED
                    *
                    * FETCH COMMAND
0048 20 2E    FC    LI    C'.'
004A 5C             LR    S,A
004B 28 03 5D       PI    TC    TYPE .
004E 28 03 F3       PI    KC    KEY IN CHARACTER
0051 28 03 5D       PI    TC    ECHO CHARACTER
0054 6E             LISL  6
0055 28 03 54       PI    TC2   TYPE SPACE
                    *
                    * DECODE EXPRESSION
0058 70       DE    LIS   0
0059 06             LR    QU,A
005A 07             LR    QL,A  CLEAR Q
005B 6E       DE7   LISL  6
005C 5E             LR    D,A   SET SIGN (ISAR=3D)
005D 70             LIS   0
005E 5A             LR    HU,A
005F 5B             LR    HL,A  CLEAR H
```

```
0060 28 03 F3 DE11   PI    KC     KEY IN CHARACTER
0063 28 03 5D        PI    TC     ECHO CHARACTER
0066 4C              LR    A,S
0067 25 2E           CI    C'.'
0069 84 D9           BZ    IN0    IF ., GO START NEW LINE
006B 25 2F           CI    H'2F'
006D 82 0D           BC    DE0    IF BEFORE 0, IS END OF CONSTANT, GO ADD
006F 25 39           CI    C'9'
0071 82 3C           BC    DE1    IF 0-9, GO CONVERT
0073 25 40           CI    H'40'
0075 82 05           BC    DE0    IF :-AT, GO ADD
0077 25 46           CI    C'F'
0079 82 36           BC    DE2    IF A-F, GO CONVERT
007B 25 2A     DE0   CI    C'*'
007D 84 42           BZ    DE3    IF *, GO PUT DC INTO H
007F 25 52           CI    C'R'
0081 84 41           BZ    DE4    IF R, GO PUT FFC0 INTO H
0083 6E              LISL  6
0084 4E              LR    A,D    (ISAR=3D)
0085 25 2D           CI    C'-'
0087 94 06           BNZ   DE5    IF - SET,...
0089 28 03 71        PI    SU     SUBTRACT H FROM Q
008C 90 04           BR    DE6
008E 28 03 E3 DE5    PI    AD     ADD H TO Q
0091 4C        DE6   LR    A,S
0092 25 0A           CI    H'A'
0094 84 46           BZ    E      IF CARRIAGE RETURN, LINE FEED, GO EXECUTE COMMAND
0096 25 2B           CI    C'+'
0098 84 C2           BZ    DE7    IF +,=,
009A 25 2D           CI    C'-'
009C 84 BE           BZ    DE7    OR -, GO START NEW H
009E 25 3D           CI    C'='
00A0 84 2F           BZ    DE8    IF =, GO TYPE Q
00A2 25 5E           CI    H'5E'
00A4 84 30           BZ    DE9    IF CARAT, GO DECREMENT DC
00A6 11              LR    H,DC
00A7 4A              LR    A,HU
00A8 52              LR    YU,A
00A9 4B              LR    A,HL
00AA 53              LR    YL,A   TRANSFER DC TO Y...
00AB 0F              LR    DC,Q   AND Q TO DC
00AC 90 AB           BR    DE
00AE 24 07     DE1   AI    H'07'  CONVERT 0-9
00B0 24 C9     DE2   AI    H'C9'  CONVERT A-F
00B2 5C              LR    S,A
00B3 4A              LR    A,HU
00B4 15              SL    4
00B5 5A              LR    HU,A
00B6 4B              LR    A,HL
00B7 14              SR    4
00B8 CA              AS    HU
00B9 5A              LR    HU,A
00BA 4B              LR    A,HL
00BB 15              SL    4
```

```
00BC CC             AS    S
00BD 58             LR    HL,A  SHIFT IN DIGIT
00BE 90 0A          BP    DE10  GO SET DATA ENTERED
00C0 11       DE3   LR    H,DC  PUT DC INTO H
00C1 90 07          BR    DE10  GO SET DATA ENTERED
00C3 20 FF    DE4   LI    H'FF'
00C5 5A             LR    HU,A
00C6 20 C0          LI    H'C0'
00C8 58             LR    HL,A  PUT FFC0 INTO H
00C9 6E       DE10  LISL  6
00CA 4C             LR    A,S
00CB 22 29          OI    H'29' AND OF +,-
00CD 5E             LR    D,A   MAKE SIGN NONZERO (ISAR=3D)
00CE 90 91          BR    DE11
00D0 28 03 1F DE8   PI    TQ    TYPE Q
00D3 90 87          BR    DE7
00D5 28 03 50 DE9   PI    TC1   TYPE CARRIAGE RETURN, LINE FEED
00D8 20 FE          LI    H'FE'
00DA 8E             ADC         DECREMENT DC
                    *
                    * EXECUTE
00DB 6F       E     LISL  7
00DC 4C             LR    A,S
00DD 25 45          CI    C'E'
00DF 94 4A          BNZ   B     IF NOT E, GO TRY B
00E1 44       E0    LR    A,IC
00E2 F6             NS    0C
00E3 81 FD          BP    E0    WAIT FOR I/O TO FINISH
00E5 1A             DI
00E6 2A FF CE       DCI   H'FFCE'
00E9 11             LR    H,DC
00EA 16             LM
00EB 52             LR    YU,A
00EC 16             LM
00ED 53             LR    YL,A  FETCH DC
00EE 2A FF B1       DCI   H'FFB1'
00F1 20 2A          LI    H'2A'
00F3 17             ST          REENTRY:  DCI...
00F4 42             LR    A,YU
00F5 17             ST
00F6 43             LR    A,YL
00F7 17             ST          VALUE OF DC...
00F8 7D             LIS   H'D'
00F9 17             ST          LR P0,Q
00FA 10             LR    DC,H
00FB 02             LR    A,QU
00FC 17             ST
00FD 03             LR    A,QL
00FE 17             ST          PLACE Q FOR RESTORING
00FF 2A FF BB       DCI   H'FFBB'
0102 16             LM
0103 04             LR    KU,A
0104 16             LM
0105 05             LR    KL,A
```

```
0106 09             LR    P,K   RESTORE PC1
0107 2A FF C0       DCI   H'FFC0'
010A 60             LISU  0
010B 68             LISL  0
010C 16         E1  LM
010D 5C             LR    S,A
010E 0A             LR    A,IS
010F 24 C1          AI    H'C1'
0111 0B             LR    IS,A
0112 94 F9          BNZ   E1    RESTORE SCRATCHPAD
0114 2A FF BD       DCI   H'FFBD'
0117 16             LM          FETCH W
0118 E9             XS    J     J/W IN ACC
0119 0B             LR    IS,A  J/W (BITS 5-0) IN ISAR
011A 0A             LR    A,IS  0,J/W IN ACC
011B E9             XS    J     J,W IN ACC
011C 59             LR    J,A   J,W IN J
011D 0A             LR    A,IS  0,J/W IN ACC AGAIN
011E E9             XS    J     J IN ACC
011F 1D             LR    W,J   W IN W (RESTORES INTERRUPT)
0120 59             LR    J,A   J IN J
0121 16             LM
0122 16             LM
0123 0B             LR    IS,A  RESTORE ISAR
0124 2A FF BE       DCI   H'FFBE'
0127 29 00 02       JMP   E2    GO RESTORE ACC, AND REENTER
                *
                * BREAKPOINT
012A 25 42      B   CI    C'B'
012C 94 21          BNZ   C     IF NOT B, GO TRY C
012E 2A FF B5       DCI   H'FFB5'
0131 02             LR    A,QU
0132 5A             LR    HU,A
0133 17             ST
0134 03             LR    A,QL
0135 5B             LR    HL,A  TRANSFER BREAKPOINT TO H...
0136 17             ST          AND STORE
0137 0E             LR    Q,DC  SET Q TO CODE LOCATION
0138 70             LIS   H'0'
0139 50             LR    XU,A
013A 5C             LR    S,A   SET INCREMENT
013B 73             LIS   H'3'
013C 51             LR    XL,A  SET COUNT
013D 28 02 CA       PI    CHQ   COPY OUT OF PROGRAM
0140 20 FC          LI    H'FC'
0142 8E             ADC         SET DC BACK TO TRAP
0143 77             LIS   H'7'
0144 17             ST          TRAP:  LR QL,A...
0145 20 29          LI    H'29'
0147 17             ST          JMP...
0148 70             LIS   H'00'
0149 17             ST
014A 75             LIS   H'05'
014B 17             ST          TO BE (AT 0005)
```

```
014C 90 32          BR    C2
                    *
                    * COPY
                    * THIS ROUTINE GIVES CORRECT RESULTS WHEN 1 OF THE FOLLOWING CONDITIONS
                    * IS MET:
                    *     1. THE NEW BLOCK LOCATION DOES NOT OVERLAP THE ORIGINAL.
                    *     2. THE BLOCK START ADDRESSES MATCH IN BIT 15 (THAT IS, ARE ON
                    *     THE SAME SIDE OF THE 7FFF/8000 BOUNDARY).
                    *     3. THE SHIFT S'-S IS NOT IN THE RANGE -8000.7FFF.
014E 25 43     C    CI    C'C'
0150 94 31          BNZ   D       IF NOT C, GO TRY D
0152 42             LR    A,YU
0153 5A             LR    HU,A
0154 43             LR    A,YL
0155 5B             LR    HL,A    TRANSFER S TO H
0156 02             LR    A,QU
0157 52             LR    YU,A
0158 FA             XS    HU
0159 5C             LR    S,A     SAVE S/S'
015A 03             LR    A,QL
015B 53             LR    YL,A    SAVE S' IN Y
015C 0E             LR    Q,DC    TRANSFER F TO Q
015D 28 03 71       PI    SU      COMPUTE F-S=COUNT IN Q
0160 02             LR    A,QU
0161 50             LR    XU,A
0162 03             LR    A,QL
0163 51             LR    XL,A    TRANSFER COUNT TO X
0164 42             LR    A,YU
0165 06             LR    QU,A
0166 43             LR    A,YL
0167 07             LR    QL,A    TRANSFER S' BACK TO Q (-S STILL IN H)
0168 28 03 E3       PI    AD      COMPUTE S'-S=SHIFT IN Q
016B 02             LR    A,QU
016C FC             XS    S       S/S'/(S'-S)
016D 91 09          BM    C0      IF S'-S NEGATIVE, GO COPY FROM START
016F 11             LR    H,DC    TRANSFER F TO H
0170 28 03 E3       PI    AD      COMPUTE (S'-S)+F=F' IN Q
0173 20 FE          LI    H'FE'   SET DECREMENT
0175 90 05          BR    C1
0177 28 03 71 C0    PI    SU      BACK TO S IN H, S' IN Q
017A 70             LIS   H'0'    SET INCREMENT
017B 5C        C1   LR    S,A
017C 28 02 CA       PI    CHQ
017F 29 00 48 C2    JMP   FC
                    * THE STARRED-OUT COPY AND SUBTRACT ROUTINES BELOW GIVE CORRECT RESULTS
                    * FOR ALL CONDITIONS.
                    * COPY
                    *C    CI    C'C'
                    *     BNZ   D       IF NOT C, GO TRY D
                    *     LR    A,YU
                    *     LR    HU,A
                    *     LR    A,YL
                    *     LR    HL,A    TRANSFER S TO H
                    *     LR    A,QU
```

```
                *       LR      YU,A
                *       LR      A,QL
                *       LR      YL,A    SAVE S' IN Y
                *       LR      Q,DC    TRANSFER F TO Q
                *       PI      SU      COMPUTE F-S=COUNT IN Q
                *       LR      A,QU
                *       LR      XU,A
                *       LR      A,QL
                *       LR      XL,A    TRANSFER COUNT TO X
                *       PI      SU      BACK TO S IN H
                *       LR      A,YU
                *       LR      QU,A
                *       LR      A,YL
                *       LR      QL,A    TRANSFER S' BACK TO Q
                *       PI      SU      COMPUTE S'-S=SHIFT IN Q
                *       BC      C0      IF S'+(10000-S).GE.10000, GO COPY FROM FINISH
                *       PI      SU      BACK TO S IN H, S' IN Q
                *       LIS     H'0'    SET INCREMENT
                *       BR      C1
                *C0     LR      H,DC    TRANSFER F TO H
                *       PI      AD      COMPUTE (S'-S)+F=F' IN Q
                *       LI      H'FE'   SET DECREMENT
                *C1     LR      S,A
                *       PI      CHQ
                *C2     JMP     FC
                *
                * DUMP
0182 25 44      D       CI      C'D'
0184 94 6D              BNZ     M       IF NOT D, GO TRY M
0186 20 12              LI      H'12'
0188 51                 LR      XL,A    SET COUNT FOR LINE OF *S
0189 13                 SL      1
018A 50                 LR      XU,A    SET COUNT FOR NULL LEADER
018B 70                 LIS     H'0'
018C 5C                 LR      S,A
018D 28 03 5D D0        PI      TC      PUNCH NULL
0190 30                 DS      XU
0191 94 FB              BNZ     D0      GO REPEAT
0193 20 2A              LI      C'*'
0195 5C                 LR      S,A
0196 28 03 5D D1        PI      TC      PUNCH *
0199 31                 DS      XL
019A 94 FB              BNZ     D1      GO REPEAT
019C 28 03 50           PI      TC1     PUNCH CARRIAGE RETURN, LINE FEED
019F 20 53              LI      C'S'
01A1 5C                 LR      S,A
01A2 28 03 5D           PI      TC      PUNCH S
01A5 11                 LR      H,DC
01A6 28 03 71           PI      SU      SUBTRACT TO FIND COUNT...
01A9 02                 LR      A,QU
01AA 50                 LR      XU,A
01AB 03                 LR      A,QL
01AC 51                 LR      XL,A    AND TRANSFER TO X
01AD 0E                 LR      Q,DC
```

```
01AE 28 03 1F       PI    TQ     PUNCH DC
01B1 28 03 50       PI    TC1    PUNCH CARRIAGE RETURN, LINE FEED
01B4 20 58    D3    LI    C'X'
01B6 5C             LR    S,A
01B7 28 03 5D       PI    TC     PUNCH X
01BA 70             LIS   0
01BB 52             LR    YU,A   CLEAR CHECKSUM
01BC 78             LIS   H'8'
01BD 53             LR    YL,A   SET BYTE COUNT
01BE 16       D2    LM
01BF 07             LR    QL,A
01C0 28 03 18       PI    TQL    PUNCH BYTE
01C3 33             DS    YL
01C4 94 F9          BNZ   D2     GO REPEAT
01C6 42             LR    A,YU
01C7 5C             LR    S,A
01C8 28 03 40       PI    TH     PUNCH CHECKSUM
01CB 28 03 50       PI    TC1    PUNCH CARRIAGE RETURN, LINE FEED
01CE 41             LR    A,XL
01CF 24 F8          AI    H'F8'
01D1 51             LR    XL,A   DECREMENT BY 8
01D2 82 E1          BC    D3     GO REPEAT
01D4 30             DS    XU
01D5 82 DE          BC    D3     GO REPEAT
01D7 20 12          LI    H'12'
01D9 51             LR    XL,A   SET COUNT FOR LINE OF *S
01DA 13             SL    1
01DB 50             LR    XU,A   SET COUNT FOR NULL LEADER
01DC 20 2A          LI    C'*'
01DE 5C             LR    S,A
01DF 28 03 5D D4    PI    TC     PUNCH *
01E2 31             DS    XL
01E3 94 F8          BNZ   D4     GO REPEAT
01E5 28 03 50       PI    TC1    PUNCH CARRIAGE RETURN, LINE FEED
01E8 70             LIS   H'0'
01E9 5C             LR    S,A
01EA 28 03 5D D5    PI    TC     PUNCH NULL
01ED 30             DS    XU
01EE 94 F8          BNZ   D5     GO REPEAT
01F0 90 8E    D6    BR    C2
                    *
                    * MEMORY
01F2 25 4D    M     CI    C'M'
01F4 94 16          BNZ   MU     IF NOT M, GO TRY MU
01F6 18             COM
01F7 5E             LR    D,A    SET M UPDATE MODE (ISAR=3E)
01F8 28 03 1F M0    PI    TQ     TYPE DC
01FB 0F             LR    DC,Q
01FC 16             LM           FETCH BYTE
01FD 0F             LR    DC,Q   RESTORE DC
01FE 07       M1    LR    QL,A   SAVE BYTE
01FF 28 03 54       PI    TC2    TYPE SPACE
0202 28 03 18       PI    TQL    TYPE BYTE
0205 28 03 54       PI    TC2    TYPE SPACE
```

```
0208 29 00 58        JMP   DE
                     *
                     * MEMORY UPDATE
020B 25 B2     MU    CI    H'B2'
020D 94 0D           BNZ   P       IF NOT MU, GO TRY P
020F 6E              LISL  6
0210 3C              DS    S
0211 92 05           BNC   MU0     IF NO DATA ENTERED, GO INCREMENT DC
0213 03              LR    A,QL
0214 17              ST            UPDATE
0215 90 02           BR    MU1     GO TYPE NEW LINE
0217 16        MU0   LM            INCREMENT DC
0218 0E        MU1   LR    Q,DC
0219 90 DE           BR    M0      GO TYPE NEW LINE
                     *
                     * PORTS
021B 25 50     P     CI    C'P'
021D 94 16           BNZ   PU      IF NOT P, GO TRY PU
021F 18              COM
0220 5E              LR    D,A     SET P UPDATE MODE (ISAR=3E)
0221 03              LR    A,QL
0222 21 0D           NI    H'0D'
0224 5C              LR    S,A     SAVE RATIONALIZED PORT NUMBER
0225 1F              INC
0226 12              SR    1
0227 51              LR    XL,A
0228 C1              AS    XL
0229 C1              AS    XL
022A 1F              INC
022B 51              LR    XL,A    STORE PORT ACCESS ADDRESS IN XL
022C 28 03 40        PI    TH      TYPE PORT NUMBER
022F 28 02 F7        PI    PA
0232 90 C8           BR    M1      GO TYPE LINE
                     *
                     * PORT UPDATE
0234 25 AF     PU    CI    H'AF'
0236 94 0B           BNZ   L       IF NOT PU, GO TRY L
0238 6E              LISL  6
0239 3D              DS    I       (ISAR=3F)
023A 92 B5           BNC   D6      IF NO DATA ENTERED, GO FETCH NEW COMMAND
023C 31              DS    XL
023D 28 02 F7        PI    PA
0240 90 AF     PU0   BR    D6
                     *
                     * LOAD
0242 25 4C     L     CI    C'L'
0244 94 52           BNZ   T       IF NOT L, GO TRY T
0246 50              LR    XU,A
0247 28 03 EC  L2    PI    RC      READ IN CHARACTER
024A 4C              LR    A,S
024B 25 53           CI    C'S'
024D 84 10           BZ    L0      IF S, GO SET DC
024F 25 58           CI    C'X'
0251 84 24           BZ    L1      IF X, GO READ 8 BYTES
```

```
0253 25 2A          CI    C'*'
0255 94 F1          BNZ   L2    IF *.ß.
0257 70             LIS   H'0'
0258 C0             AS    XU
0259 94 ED          BNZ   L2    AND DATA READ,...
025B 29 00 43 L4    JMP   IN0   GO FETCH NEW COMMAND
025E 50       L0    LR    XU,A
025F 28 02 DC       PI    RH
0262 15             SL    4
0263 53             LR    YL,A
0264 28 02 DC       PI    RH
0267 C3             AS    YL
0268 56             LR    QU,A
0269 28 02 DC       PI    RH
026C 15             SL    4
026D 53             LR    YL,A
026E 28 02 DC       PI    RH
0271 C3             AS    YL
0272 57             LR    QL,A
0273 0F             LR    DC,Q  SET DC
0274 90 D2          BR    L2
0276 70       L1    LIS   H'0'
0277 50             LR    XU,A  SET DATA READ
0278 52             LR    YU,A  CLEAR CHECKSUM
0279 78             LIS   H'8'
027A 51             LR    XL,A  SET BYTE COUNT
027B 28 02 DC L3    PI    RH
027E 15             SL    4
027F 53             LR    YL,A
0280 28 02 DC       PI    RH
0283 C3             AS    YL
0284 17             ST          LOAD BYTE INTO MEMORY
0285 31             DS    XL
0286 94 F4          BNZ   L3    GO REPEAT
0288 7F             LIS   H'F'
0289 F2             NS    YU    CLEAN UP CHECKSUM
028A 53             LR    YL,A
028B 28 02 DC       PI    RH
028E E3             XS    YL
028F 84 B7          BZ    L2    IF CHECKSUMS DIFFER,...
0291 0E             LR    Q,DC
0292 28 03 1F       PI    TQ    TYPE QUESTIONABLE ADDRESS...
0295 90 B1          BR    L2    AND GET BACK FAST TO CATCH NEXT X
                    *
                    * TYPE
0297 25 54    T     CI    C'T'
0299 94 A6          BNZ   PU0   IF NOT T, GO FETCH NEW COMMAND
029B 11             LR    H,DC
029C 28 03 71       PI    SU    SUBTRACT TO FIND COUNT...
029F 52             LR    A,QU
02A0 50             LR    XU,A
02A1 53             LR    A,QL
02A2 51             LR    XL,A  AND TRANSFER TO X
02A3 70             LIS   H'0'
```

```
02A4 53             LR    YL,A   SET INDEX FOR LINE
02A5 90 08          BR    T0
02A7 7F       T3    LIS   H'F'
02A8 F3             NS    YL
02A9 94 08          BNZ   T1     IF MULTIPLE OF 10,...
02AB 28 03 50       PI    TC1    TYPE CARRIAGE RETURN, LINE FEED
02AE 0E       T0    LR    Q,DC
02AF 28 03 1F       PI    TQ     TYPE DC
02B2 73       T1    LIS   H'3'
02B3 F3             NS    YL
02B4 94 04          BNZ   T2     IF MULTIPLE OF 4,...
02B6 28 03 54       PI    TC2    TYPE EXTRA SPACE
02B9 28 03 54 T2    PI    TC2    TYPE SPACE
02BC 16             LM
02BD 07             LR    QL,A
02BE 28 03 18       PI    TQL    TYPE BYTE
02C1 33             DS    YL
02C2 31             DS    XL
02C3 82 E3          BC    T3     GO REPEAT
02C5 30             DS    XU
02C6 82 E0          BC    T3     GO REPEAT
02C8 90 92          BR    L4
                *
                * COPY FROM H TO Q
                * THIS ROUTINE COPIES A BLOCK OF X+1 BYTES, STARTING AT ADDRESS H, INTO
                * THE LOCATION STARTING AT ADDRESS Q.  COPYING IS PERFORMED FORWARD OR
                * BACKWARD FROM THESE ADDRESSES.
                * ENTRY:  H=BLOCK START ADDRESS.  Q=NEW LOCATION START ADDRESS.  X=BLOCK
                * LENGTH-1.  RS=0 FOR FORWARD COPYING, FF FOR BACKWARD COPYING.
                * EXIT:  H=BLOCK FINISH ADDRESS.  Q=ADDRESS BEYOND NEW LOCATION FINISH.
                * X=FFFF.  RS=UNCHANGED.  DC,ACC,W USED.
02CA 08       CHQ   LR    K,P
02CB 10             LR    DC,H
02CC 16       CHQ0  LM            FETCH BYTE
02CD 11             LR    H,DC
02CE 0F             LR    DC,Q
02CF 17             ST            PLACE BYTE
02D0 4C             LR    A,S
02D1 8E             ADC           IN/DECREMENT DC'
02D2 0E             LR    Q,DC
02D3 10             LR    DC,H
02D4 8E             ADC           IN/DECREMENT DC
02D5 31             DS    XL
02D6 82 F5          BC    CHQ0   GO REPEAT
02D8 30             DS    XU
02D9 82 F2          BC    CHQ0   GO REPEAT
02DB 0C             PK
                *
                * READ HEX CHARACTER
                * THIS ROUTINE OPERATES THE READER CLUTCH TO READ A HEXADECIMAL DIGIT
                * CHARACTER, CONVERTS IT TO A HALF BYTE, ADDS IT TO THE CHECKSUM IN YU,
                * AND RETURNS WITH IT IN ACC.
                * SEE I/O INITIALIZATION REQUIRED BY TI.
                * ENTRY:  YU=PREVIOUS CHECKSUM.
```

```
                * EXIT:  YU=UPDATED CHECKSUM.  ACC=HALF BYTE.  W USED.
02DC 08         RH      LR      K,P
02DD A5                 INS     PT
02DE 21 40              NI      TO      PRESERVE OUTPUT BIT
02E0 B5                 OUTS    PT      ENGAGE READER CLUTCH
02E1 44         RH0     LR      A,IC
02E2 18                 COM
02E3 94 FD              BNZ     RH0     WAIT FOR INPUT COMPLETE
02E5 45                 LR      A,IB
02E6 21 7F              NI      H'7F'
02E8 25 39              CI      C'9'
02EA 02 03              BNC     RH1     IF 0-9,...
02EC 24 07              AI      H'07'   ADD 7
02EE 24 C9      RH1     AI      H'C9'
02F0 55                 LR      IB,A
02F1 C2                 AS      YU
02F2 52                 LR      YU,A    ADD TO CHECKSUM
02F3 45                 LR      A,IB
02F4 34                 DS      IC      SET INPUT READY AGAIN
02F5 0C                 PK
02F6 2B                 NOP
                *
                * PORT ACCESS
02F7 08         PA      LR      K,P
02F8 03                 LR      A,QL
02F9 5B                 LR      HL,A    SAVE BYTE IN HL
02FA 73                 LIS     H'3'
02FB 06                 LR      QU,A
02FC 41                 LR      A,XL
02FD 07                 LR      QL,A    SET UP PORT ACCESS ADDRESS IN Q
02FE 4B                 LR      A,HL
02FF 0D                 LR      P0,Q
0300 B0                 OUTS    H'0'
0301 A0                 INS     H'0'
0302 0C                 PK
0303 B1                 OUTS    H'1'
0304 A1                 INS     H'1'
0305 0C                 PK
0306 B4                 OUTS    H'4'
0307 A4                 INS     H'4'
0308 0C                 PK
0309 2B                 NOP             DON'T MESS WITH TELETYPE PORT!
030A A5                 INS     H'5'
030B 0C                 PK
030C B8                 OUTS    H'8'
030D A8                 INS     H'8'
030E 0C                 PK
030F B9                 OUTS    H'9'
0310 A9                 INS     H'9'
0311 0C                 PK
0312 BC                 OUTS    H'C'
0313 AC                 INS     H'C'
0314 0C                 PK
0315 BD                 OUTS    H'D'
```

```
0316 AD           INS   H'D'
0317 0C           PK
                  *
                  * TYPE QL
                  * THIS ROUTINE TYPES QL IN 2 HEXADECIMAL DIGIT CHARACTERS.
                  * SEE I/O INITIALIZATION REQUIRED BY TI.
                  * ENTRY:  QL=BYTE.  RS,R(S-1),R(S-2) USEABLE.
                  * EXIT:  QL=UNCHANGED.  ACC,W,RS,R(S-1),R(S-2) USED.
0318 08           TQL   LR    K,P
0319 01                 LR    A,KL
031A 5E                 LR    D,A
031B 00                 LR    A,KU
031C 5E                 LR    D,A
031D 90 11              BR    TQ0
                  *
                  * TYPE Q
                  * THIS ROUTINE TYPES Q IN 4 HEXADECIMAL DIGIT CHARACTERS.
                  * SEE I/O INITIALIZATION REQUIRED BY TI.
                  * ENTRY:  Q=2 BYTES.  RS,R(S-1),R(S-2) USEABLE.
                  * EXIT:  Q=UNCHANGED.  ACC,W,RS,R(S-1),R(S-2) USED.
031F 08           TQ    LR    K,P
0320 01                 LR    A,KL
0321 5E                 LR    D,A
0322 00                 LR    A,KU
0323 5E                 LR    D,A
0324 02                 LR    A,QU
0325 14                 SR    4
0326 5C                 LR    S,A
0327 28 03 40           PI    TH
032A 02                 LR    A,QU
032B 5C                 LR    S,A
032C 28 03 40           PI    TH
032F 03           TQ0   LR    A,QL
0330 14                 SR    4
0331 5C                 LR    S,A
0332 28 03 40           PI    TH
0335 03                 LR    A,QL
0336 5C                 LR    S,A
0337 28 03 40           PI    TH
033A 4D                 LR    A,I
033B 4D                 LR    A,I
033C 04                 LR    KU,A
033D 4C                 LR    A,S
033E 05                 LR    KL,A
033F 0C                 PK
                  *
                  * TYPE HEX CHARACTER
                  * THIS ROUTINE ADDS THE HALF BYTE IN RS TO THE CHECKSUM IN YU, CONVERTS
                  * IT TO A HEXADECIMAL DIGIT CHARACTER, AND TYPES IT.
                  * SEE I/O INITIALIZATION REQUIRED BY TI.
                  * ENTRY:  RS=HALF BYTE (ONLY RIGHT HALF USED).  YU=PREVIOUS CHECKSUM.
                  * EXIT:  RS=CHARACTER.  YU=UPDATED CHECKSUM.  ACC,W USED.
0340 08           TH    LR    K,P
0341 4C                 LR    A,S
```

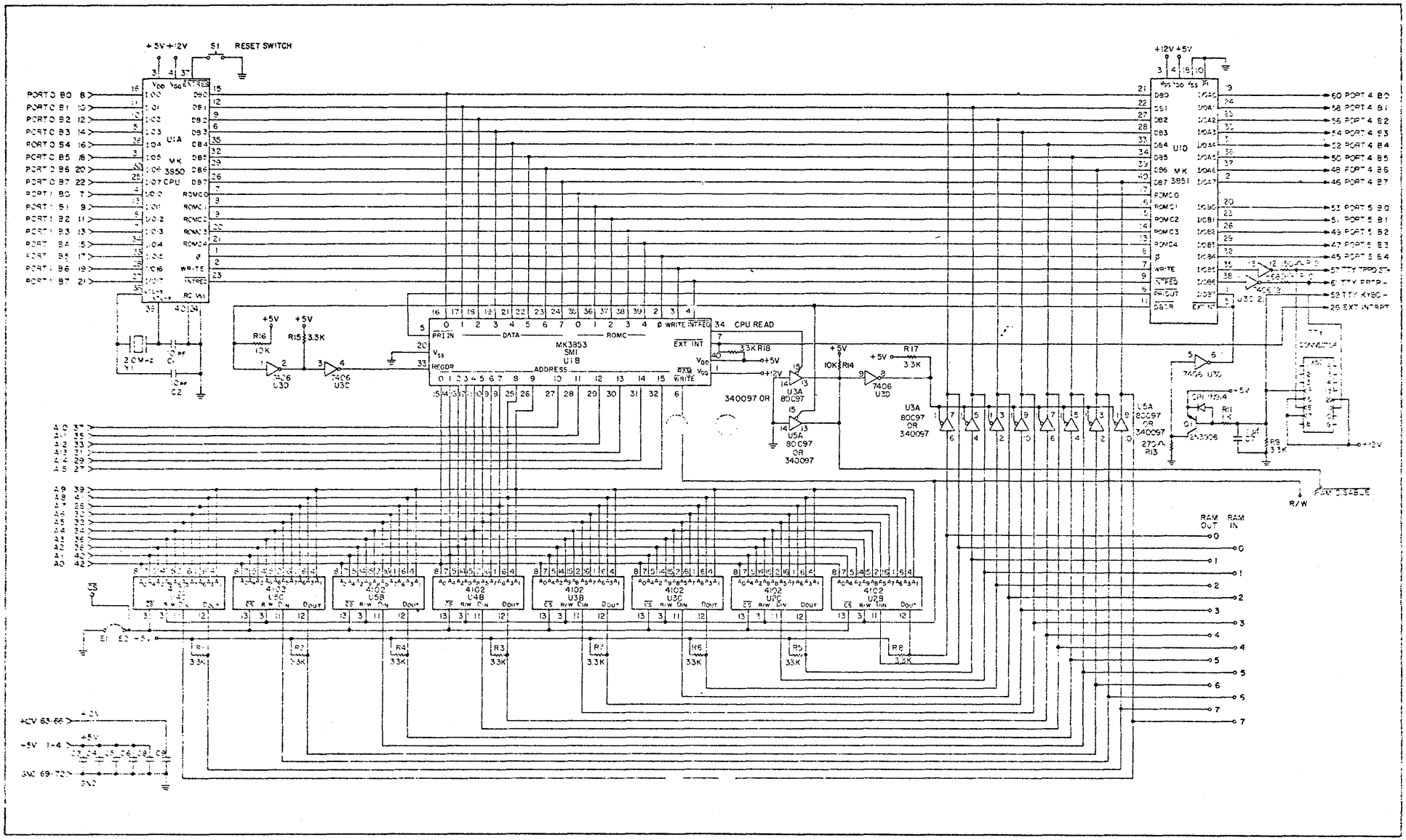
```
0342 C2          AS    YU
0343 52          LR    YU,A  ADD TO CHECKSUM
0344 7F          LIS   H'F'
0345 FC          NS    S
0346 25 09       CI    H'09'
0348 82 03       BC    TH0   IF 0-9,...
034A 24 07       AI    H'07' ADD 7
034C 24 30   TH0 AI    H'30'
034E 90 08       BR    TC4
                 *
                 * TYPE OUT CHARACTER
                 * THIS ROUTINE TYPES A CARRIAGE RETURN, LINE FEED.
                 * SEE I/O INITIALIZATION REQUIRED BY TI.
                 * ENTRY:  RS USEABLE.
                 * EXIT:  RS=A.  ACC.W USED.
0350 08          TC1   LR    K,P
0351 7D          LIS   H'D'
0352 90 07       BR    TC4   GO TYPE CARRIAGE RETURN
                 * THIS ROUTINE TYPES A SPACE.
                 * SEE I/O INITIALIZATION REQUIRED BY TI.
                 * ENTRY:  RS USEABLE.
                 * EXIT:  RS=20.  ACC.W USED.
0354 08          TC2   LR    K,P
0355 20 20       LI    C' '
0357 90 02       BR    TC4   GO TYPE SPACE
0359 7A          TC3   LIS   H'A'
035A 5C          TC4   LR    S,A
035B 90 02       BR    TC0
                 * THIS ROUTINE TYPES THE CHARACTER (8 BITS) IN RS.  IF THE CHARACTER IS
                 * A CARRIAGE RETURN, IT ALSO TYPES A LINE FEED.
                 * SEE I/O INITIALIZATION REQUIRED BY TI.
                 * ENTRY:  RS=CHARACTER.
                 * EXIT:  RS=UNCHANGED (EXCEPT D TO A).  ACC.W USED.
035D 08          TC    LR    K,P
035E 46          TC0   LR    A,0C
035F 18               COM
0360 94 FB            BNZ   TC0   WAIT FOR OUTPUT COMPLETE
0362 4C               LR    A,S
0363 18               COM
0364 57               LR    0B,A  PLACE BYTE INVERTED
0365 1A               DI
0366 20 57            LI    H'57'
0368 56               LR    0C,A  SET OUTPUT COUNT
0369 28 03 7F         PI    TI    CREATE A SOFTWARE INTERRUPT TO TIMER ROUTINE
036C 7D               LIS   H'D'
036D FC               XS    S
036E 84 EA            BZ    TC3   IF CARRIAGE RETURN, GO TYPE LINE FEED
0370 0C               PK
                 *
                 * SUBTRACT H FROM Q
                 * ENTRY:  Q=SUBTRAHEND.  H=MINUEND.
                 * EXIT:  Q=DIFFERENCE.  H=NEGATIVE MINUEND.  CARRY SET ON ADDITION OF
                 * Q AND 10000-H.  ACC USED.
0371 08          SU    LR    K,P
```

```
0372 4A            LR    A,HU
0373 18            COM
0374 5A            LR    HU,A
0375 4B            LR    A,HL
0376 18            COM         COMPLEMENT...
0377 1F            INC
0378 5B            LR    HL,A
0379 4A            LR    A,HU
037A 19            LNK
037B 5A            LR    HU,A  AND INCREMENT H
037C 90 67         BR    AD0
                 * SUBTRACT H FROM Q
                 * THIS ROUTINE PUTS 10000-H IN H AND Q+(10000-H) IN Q WITH CARRY SET
                 * ACCORDINGLY.
                 *SU   LR    K,P
                 *     LR    A,HU
                 *     COM
                 *     LR    HU,A
                 *     LR    A,HL
                 *     COM         COMPLEMENT...
                 *     INC
                 *     LR    HL,A
                 *     LR    A,HU
                 *     LNK
                 *     LR    HU,A  AND INCREMENT H
                 *     BNC   AD0   IF H=0,...
                 *     PK          RETURN WITH CARRY SET
                 *
                 * TIMER INTERRUPT
                 * THIS ROUTINE PERFORMS FULL DUPLEX SERIAL-BIT I/O OF BYTES IB,OB.  BIT
                 * 4 OF PORT 5 SELECTS 110 BAUD (2 STOP BITS) OR 300 BAUD (1 STOP BIT).
                 * ROUTINE WILL RETURN TO CALLING PROGRAM AS SOON AS I/O IS INITIATED.
                 * TIMER WILL BE SET TO INTERRUPT APPROPRIATELY TO PERMIT MULTIPROCESSING
                 * DURING I/O.
                 * INITIALIZATION:  BEFORE FIRST I/O, INITIALIZE IC=FE, OC=FF, AND CALL
                 * TI TO SET INTERRUPTS (CAN ALSO BE PERFORMED BY OUTPUTTING A
                 * CHARACTER).  THEREAFTER:
                 *     IC NONNEGATIVE INDICATES INPUT IN PROGRESS.
                 *     IC=FF INDICATES INPUT BYTE ARRIVED IN IB: DECREMENT IC WHEN
                 *        COLLECTING TO AVOID COLLECTING TWICE.
                 *     IC=FE INDICATES NO NEW BYTE NOR INPUT IN PROGRESS.
                 *     OC NONNEGATIVE INDICATES OUTPUT IN PROGRESS.
                 *     OC=FF INDICATES OUTPUT OF BYTE IN OB COMPLETED.
                 * ENTRY:  IC,OC=COUNTS FOR I/O.  IB,OB=DATA BYTES FOR I/O.
                 * EXIT:  IC,OC=DECREMENTED COUNTS FOR I/O.  IB,OB=SHIFTED DATA BYTES FOR
                 * I/O.  ACC,W (WITH INTERRUPT ENABLED) RESTORED.
037E 28            NOP         TIMER INTERRUPT VECTOR
037F 58          TI    LR    AS,A  SAVE ACC
0380 1E            LR    J,W   SAVE W
0381 20 00         LI    H'00'
0383 B7            OUTS  PC    SET TIMER FOR 24 COUNTS (= 372US + 34US SETTING
0384 A5            INS   PT       DELAY)
0385 21 10         NI    TB
0387 94 04         BNZ   TI0   IF 110 BAUD,...
```

```
0389 20 CA          LI    H'CA'
038B B7             OUTS  PC     SET TIMER FOR 69 COUNTS (= 1069.5US + 66US SETTING
038C 44       TI0   LR    A,IC       DELAY)
038D 18             COM
038E 81 11          BP    TI1    IF INPUT IN PROGRESS,...
0390 34             DS    IC     DECREMENT INPUT COUNT
0391 21 07          NI    H'07'
0393 94 17          BNZ   TI2    IF NOT AT BIT TIME,...
0395 45             LR    A,IB
0396 12             SR    1
0397 55             LR    IB,A   SHIFT OVER BYTE
0398 A5             INS   PT
0399 18             COM          INVERT INPUT BIT
039A 21 80          NI    H'80'  EXTRACT
039C C5             AS    IB
039D 55             LR    IB,A   ADD IN
039E 90 0C          BR    TI2
03A0 A5       TI1   INS   PT
03A1 81 09          BP    TI2    IF START BIT,...
03A3 21 40          NI    TO     PRESERVE OUTPUT BIT
03A5 22 20          OI    TR
03A7 B5             OUTS  PT     DISENGAGE READER CLUTCH
03A8 20 49          LI    H'49'
03AA 54             LR    IC,A   SET INPUT COUNT
03AB 46       TI2   LR    A,OC
03AC 18             COM
03AD 81 29          BP    TI3    IF OUTPUT IN PROGRESS,...
03AF 36             DS    OC     DECREMENT OUTPUT COUNT
03B0 21 07          NI    H'07'
03B2 94 24          BNZ   TI3    IF AT BIT TIME,...
03B4 46             LR    A,OC
03B5 25 56          CI    H'56'
03B7 84 0D          BZ    TI4    IF START BIT TIME, GO CHECK BAUD RATE
03B9 71             LIS   H'01'
03BA F7             NS    OB     EXTRACT DATA BIT
03BB 47             LR    A,OB
03BC 94 12          BNZ   TI5    IF 1, GO OUTPUT BIT
03BE 12             SR    1
03BF 57             LR    OB,A   SHIFT OVER BYTE
03C0 A5             INS   PT
03C1 21 20          NI    TR     PRESERVE READER CLUTCH BIT
03C3 90 12          BR    TI6
03C5 A5       TI4   INS   PT
03C6 21 10          NI    TB
03C8 84 08          BZ    TI7    IF 300 BAUD,...
03CA 20 4E          LI    H'4E'
03CC 56             LR    OC,A   REDUCE COUNT BY 8 TO OUTPUT ONLY 1 STOP BIT
03CD 90 03          BR    TI7    GO OUTPUT START BIT
03CF 12       TI5   SR    1
03D0 57             LR    OB,A   SHIFT OVER BYTE
03D1 A5       TI7   INS   PT
03D2 21 20          NI    TR     PRESERVE READER CLUTCH BIT
03D4 22 40          OI    TO
03D6 B5       TI6   OUTS  PT
```

```
03D7 73        TI3   LIS   H'3'
03D8 B6              OUTS  PI     ENABLE TIMER
03D9 44              LR    A,IC
03DA F6              NS    0C
03DB 81 03           BP    TI8    IF BOTH COUNTS NEGATIVE....
03DD 71              LIS   H'1'
03DE B6              OUTS  PI     DISABLE TIMER
03DF 1D        TI8   LR    W,J    RESTORE W
03E0 48              LR    A,AS   RESTORE ACC
03E1 1B              EI
03E2 1C              POP
               *
               * ADD H TO Q
               * THIS ROUTINE PUTS Q+H IN Q AND SETS CARRY ACCORDINGLY.
               * ENTRY:  Q=ADDEND.  H=AUGEND.
               * EXIT:  Q=SUM.  H=UNCHANGED.  CARRY SET ON ADDITION OF Q AND H.  ACC
               * USED.
03E3 08        AD    LR    K,P
03E4 03        AD0   LR    A,QL
03E5 CB              AS    HL
03E6 07              LR    QL,A
03E7 02              LR    A,QU
03E8 19              LNK
03E9 CA              AS    HU
03EA 06              LR    QU,A   ADD TO Q
03EB 0C              PK
               *
               * READ IN CHARACTER
               * THIS ROUTINE ENGAGES THE READER CLUTCH TO READ A CHARACTER (7 BITS)
               * INTO RS.
               * SEE I/O INITIALIZATION REQUIRED BY TI.
               * ENTRY:  RS USEABLE.
               * EXIT:  RS=CHARACTER.  ACC,W USED.
03EC 08        RC    LR    K,P
03ED A5              INS   PT
03EE 21 40           NI    T0     PRESERVE OUTPUT BIT
03F0 B5              OUTS  PT     ENGAGE READER CLUTCH
03F1 90 02           BR    KC0
               *
               * KEY IN CHARACTER
               * THIS ROUTINE READS A CHARACTER (7 BITS) INTO RS.
               * SEE I/O INITIALIZATION REQUIRED BY TI.
               * ENTRY:  RS USEABLE.
               * EXIT:  RS=CHARACTER.  ACC,W USED.
03F3 08        KC    LR    K,P
03F4 44        KC0   LR    A,IC
03F5 18              COM
03F6 94 FD           BNZ   KC0    WAIT FOR INPUT COMPLETE
03F8 45              LR    A,IB   FETCH BYTE
03F9 34              DS    IC     SET INPUT READY AGAIN
03FA 21 7F           NI    H'7F'
03FC 5C              LR    S,A
03FD 0C              PK

               * INPUT INTERRUPT
03FE 90 80           DC    H'9080' INPUT INTERRUPT: GO TO TIMER INTERRUPT
             END
NUMBER OF ERRORS=  0
```



Schematic of Mostek F8 Survival Kit

10. THE LATEST DEVELOPMENTS
IN PROM AND FPLA PROGRAMMING

DICK WOODS
DATA I/O CORPORATION
Sunnyvale, California

PROM programming techniques have progressed rapidly during the past year and several new innovations have been introduced which will surely be of great significance to PROM users. This paper presents an overview of the latest developments.

GENERIC FAMILY PROGRAM CARDS

The "generic" card set is designed to program several PROMs in a bipolar family for a given manufacturer. Now, instead of requiring a program card set for each PROM to be programmed, it is possible to have one card set for a family of PROMs which are designed around a common fusing specification. This provides a considerable savings in terms of the hardware necessary to program any one particular manufacturer's family of PROMs. As an example, it is now possible to program all of Monolithic Memories' PROMs using one card set and several socket adapters which control the programmer's address limit and byte size. A PROM comparison chart is shown in Table 1, which indicates the generic family of cards and the appropriate socket adapters.

UNIVERSAL CALIBRATOR

A Universal Calibrator is now available which can be used to quickly determine if the manufacturer's critical programming specifications are being met. This task is accomplished by using the Universal Calibrator and any inexpensive digital volt meter and calibration chart. The new Universal Calibrator uses the PROM programmer as a source of voltage during the calibration procedure. Concise calibration instructions are included with each program adapter for calibrating a particular card set. Typical tests and calibrations include program voltage and current levels, read voltage and current levels, address line voltages, read lines, chip enable, etc. Periodic calibration assures maximum program yield and should be performed regularly. A single universal calibrator accepts program adapters for most PROM types manufactured today.

ROM PATTERN TESTING

If masked ROMs are being used for production hardware, it may be desirable to verify their operation after receiving them from the manufacturer. It is possible to use the PROM programmer as an incoming inspection device to verify the functional operation and the pattern of the ROM. This technique is performed by plugging two standard cards into the mainframe of the programmer. These cards are called "read-only cards". They have the ability to address the ROM and read the outputs into

the programmer bus structure. The programmer thus sequences through the addresses comparing the data found in the master device with the data found in the device under test. If there are any differences the programmer will stop, indicating the address and showing the bit pattern of both the master and the device under test. ROM patterns up to 16K size (2Kx8) can be checked using this technique.

PAPER TAPE FORMATS

All of DATA I/O PROM programmers which are equipped with paper tape readers will accept binary or image data as a standard format. If the optional translator card is inserted in the programmer, several other tape formats are then acceptable for translation. These include ASCII BNPF, BHLF, Hexadecimal, and Octal. Two popular forms of Hexadecimal are acceptable, the first being DATA I/O's format, the second, Scientific Microsystems' for use with their ROM simulator. Figures 1, 2, 3 and 4 show examples of binary, BNPF and Hexadecimal tape, respectively.

INTERACTIVE PROGRAMMING OF MOS PROMS

Interactive program cards are now available for programming all MOS PROMs and are designed to insure a known amount of stored charge on each floating gate. The technique used is to charge the floating gate with pulses until threshold is

BINARY PAPER TAPE

- MOST COMPACT FORMAT
- NEEDS AN IDENTIFIER FOR BEGINNING OF TAPE DATA
- A HOLE EQUALS A PROGRAMMED BIT

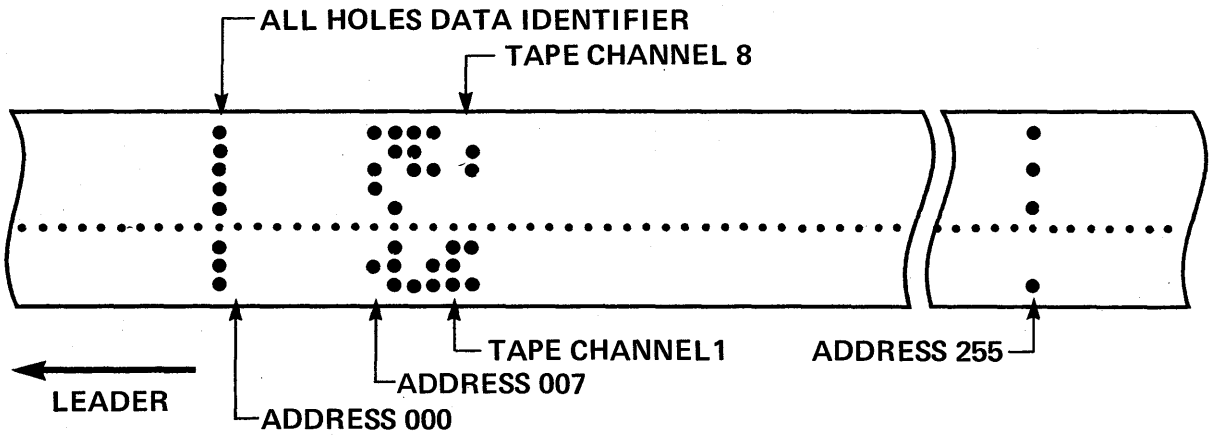


Figure 1

ASCII PAPER TAPE

- BEST FOR TELEPHONE (TWX) TRANSMISSION
- LONGEST TAPE
- BPNF FORMAT IS MOST POPULAR
- TTY CAN BE USED FOR HARD COPY READOUT

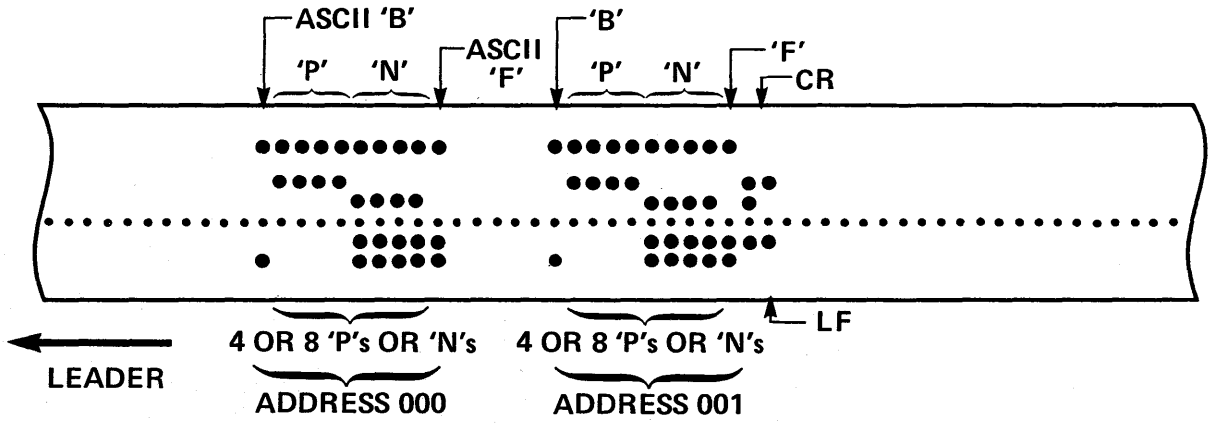


Figure 2

ASCII HEXADECIMAL PAPER TAPE

- MEDIUM LENGTH FORMAT
- EASY PREPARATION ON A TTY
- USES HEX CHARACTERS 0 THRU 9 AND A THRU F
- CONTROL CHARACTERS

	DATA I/O	SMS
TAPE START	SOH	DCI
EXECUTE	SPACE	APOSTROPHE
END OF TEXT	ETX	

Figure 3

ASCII HEXADECIMAL PAPER TAPE

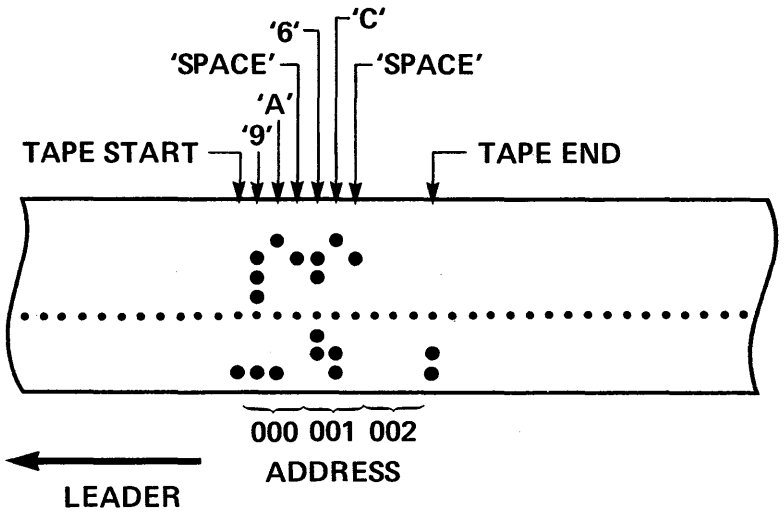


Figure 4

reached. The number of pulses to reach threshold is then multiplied by a constant for the particular PROM being programmed. Pulses are then added so that the total charge is at a known level for proper data retention.

Coupled with the new interactive programming is the ability to easily calibrate a UV erase lamp and PROM programmer so that PROM erasing is properly done. This technique consists of erasing the PROM under the UV lamp in small increments of time and verifying the data on the PROM programmer until the data is just at the threshold point, i.e., has just been erased. The total accumulated time of erasure is now multiplied by a factor of three, which equals the minimum time under the erase lamp for proper erasure of a PROM prior to programming using the new interactive programming technique.

ROMULATOR™ and RAM-PAK™

During the design of any system which will contain PROMs as a memory element, it may become necessary to change the PROM data several times before the proper pattern is achieved. A useful tool which can be used in this situation is a simulator or emulator of the PROM in use. To help the designer, the Romulator and RAM-PAK have been developed. The Romulator provides the data and address readout and a keyboard for data entry. The RAM-PAK is a plug in, high density memory cube approximately 3x3x3 inches in dimension, which can look and

operate like any PROM, be it bipolar, AIM, MNOS, or MOS, and has a capacity to 8K bits. Data can be entered and verified using the data and address display in binary, Hexadecimal or Octal. The RAM-PAK can also be bidirectionally interfaced to DATA I/O PROM programmers as an input or output device. The RAM-PAK, after being programmed by the Romulator or programmer, can be powered by its own DC battery pack to insure data integrity during transit prior to plugging into the host system. The Romulator and RAM-PAK can be interfaced to a user system, allowing a PROM program to be emulated, verified or updated before commiting to a PROM. The Romulator is the ideal approach to system debugging.

MODEL VII

If you have a microprocessor based system and would like to have a universal PROM programmer which can be controlled by the processor, a unit (the Model VII) will be available soon from DATA I/O. This programmer is universal and will program any PROM which exists on the market today. It can be interfaced to your processor system via RS-232 or 8 bit parallel interfaces. The programmer includes 8K of RAM buffer for storage of the data prior to programming. Also, a display of the sequence is included, such as, continuity tests, illegal bit check, program, verify and pass or fail. A switch is provided for local or remote selection so that control can be

handled by the computer or by the operator at the front panel. All previously developed program card sets for bipolar, MOS, AIM or MNOS PROM technologies can be used within the new Model VII.

FPLA PROGRAMMER

During 1975, two field programmable logic arrays (FPLAs) were introduced by Intersil and Signetics. These programmable logic devices are the first of a series from several of the IC manufacturers. DATA I/O has produced a programmer called the Model X, which is capable of programming the existing FPLAs on the market today, and is capable of programming FPLAs which are released in the future. The Model X FPLA Programmer has the capability of programming FPLAs from a variety of input sources such as punched paper tape, mark sense cards or a master FPLA in the copy mode. Table 2 shows the FPLAs which are now available and those which will be available in the near future. The Model X programmer can be used to program the existing devices by merely replacing a program card set within the programmer. These card sets are approved by each FPLA manufacturer before their release for sale. The programmer contains a CRT which is the heart of the system and displays to the operator all pertinent information associated with FPLA programming.

The Model X control system features a custom CPU system with 24K of PROM memory and a 15 word instruction stack register.

All of the 132 instructions are completely flexible, so that as new developments and improvements are made, units in the field may be easily updated. Table 3 shows an example of a typical FPLA truth table in an ASCII format. The input data to the programmer is coded in ASCII to define product term number, input term, output term and activation level. The input data is monitored within the Model X to insure correct format and field length. A listing mode is available so that an ASCII coded paper tape may be observed on the CRT prior to programming the FPLA. The modes of operation of the Model X are divided into three categories; input to RAM, RAM to FPLA, and clear RAM. The input to RAM mode is selected for paper tape reader, mark sense reader or remote input. A keyboard is used to manually enter data to the RAM buffer which is displayed on the CRT or to update the data if a master FPLA has been used to load the data to the RAM buffer. The RAM to FPLA mode is used to program and to compare the contents of the RAM with the FPLA. The clear RAM mode is used to erase the entire contents of the RAM. During the programming cycle the programmer automatically sequences through illegal bit check, program, array verify and logical verify before indicating to the operator that the FPLA has been satisfactorily programmed.

An output interface is available to connect the Model X to an ASR-33 teletype terminal, which can be used as a hard copy printout of the RAM contents. Thus when the programmer

FPLA OVERVIEW

	NUMBER OF INPUTS	PRODUCT TERMS	NUMBER OF OUTPUTS	PACKAGE SIZE (PINS)	AVAILABILITY
INTERSIL	14	48	8	24	NOW
SIGNETICS	16	48	8	28	NOW
AMD	16	48	8	28	EARLY 1976
MMI	14	48	8	24	1Q76

SEVERAL OTHER MANUFACTURERS ARE IN THE DESIGN PROCESS AND WILL RELEASE THEIR PRODUCTS IN 1976

Table 2

An example of a typical FPLA Truth Table is shown below.

TABLE 3

```
STX                                          *A  LLLLLLLL

*P  ØØ  *I  LL---L-------H  *F  -----AA-
*P  Ø1  *I  LLLLLH--------  *F  A----AAA
*P  Ø2  *I  HLLLLH--------  *F  AA---AA-
*P  Ø3  *I  HHLLLH--------  *F  -A---A--
*P  Ø4  *I  LHLLLH------H-  *F  --A--AA-
*P  Ø5  *I  LHLLLH-----HL-  *F  ---A-AA-
*P  Ø6  *I  LHLLLH----HLL-  *F  --AA-AA-
*P  Ø7  *I  LHLLLH---HLLL-  *F  ----AAA-
*P  Ø8  *I  LHLLLH--HLLLL-  *F  --A-AAA-
*P  Ø9  *I  LHLLLH-HLLLLL-  *F  ---AAAA-
*P  1Ø  *I  LHLLLHHLLLLLL-  *F  --AAAAA-
*P  11  *I  LHLLLHLLLLLLL-  *F  --------
*P  12  *I  LL---L------HL  *F  --A--AA-
*P  13  *I  LLHLLH--------  *F  A-A--AAA
*P  14  *I  HLHLLH--------  *F  AAA--AA-
*P  15  *I  HHHLLH--------  *F  -AA--A--
*P  16  *I  LHHLLH-----H--  *F  ---A-AA-
*P  17  *I  LHHLLH----HL--  *F  --AA-AA-
*P  18  *I  LHHLLH---HLL--  *F  ----AAA-
*P  19  *I  LHHLLH--HLLL--  *F  --A-AAA-
*P  2Ø  *I  LHHLLH-HLLLL--  *F  ---AAAA-
*P  21  *I  LHHLLHHLLLLL--  *F  --AAAAA-
*P  22  *I  LHHLLHLLLLLL-H  *F  -----AA-
*P  23  *I  LHHLLHLLLLLL-L  *F  --------
*P  24  *I  LL---L-----HLL  *F  ---A-AA-
*P  25  *I  LLLHLH--------  *F  A--A-AAA
*P  26  *I  HLLHLH--------  *F  AA-A-AA-
*P  27  *I  HHLHLH--------  *F  -A-A-A--
*P  28  *I  LHLHLH----H---  *F  --AA-AA-
*P  29  *I  LHLHLH---HL---  *F  ----AAA-
*P  3Ø  *I  LHLHLH--HLL---  *F  --A-AAA-
*P  31  *I  LHLHLH-HLLL---  *F  ---AAAA-
*P  32  *I  LHLHLHHLLLL---  *F  --AAAAA-
*P  33  *I  LHLHLHLLLL--H  *F  -----AA-
*P  34  *I  LHLHLHLLLL-HL  *F  --A--AA-
*P  35  *I  LHLHLHLLLL-LL  *F  --------
*P  36  *I  LL---L----HLLL  *F  --AA-AA-
*P  37  *I  LLHHLH--------  *F  A-AA-AAA
*P  38  *I  HLHHLH--------  *F  AAAA-AA-
*P  39  *I  HHHHLH--------  *F  -AAA-A--
*P  4Ø  *I  LHHHLH---H----  *F  ----AAA-
*P  41  *I  LHHHLH--HL----  *F  --A-AAA-
*P  42  *I  LHHHLH-HLL----  *F  ---AAAA-
*P  43  *I  LHHHLHHLLL----  *F  --AAAAA-
*P  44  *I  LHHHLHLLLL---H  *F  -----AA-
*P  45  *I  LHHHLHLLLL--HL  *F  --A--AA-
*P  46  *I  LHHHLHLLLL--LL  *F  ---A-AA-
*P  47  *I  LHHHLHLLLL-LLL  *F  --------

                                             ETX
```

ASCII INPUT DATA FORMAT

Input data is configured in ASCII coding to define Product Term #, Input Term, Output Term and Activation Level. Checking of input format is accomplished within the Model X to insure correct length of fields.

is used in an engineering environment, a copy of the FPLA contents can be easily obtained for record keeping purposes.

LOOKING AHEAD

The outlook for the future in programmable devices should see several new PROMs and FPLAs from the manufacturers. In PROMs, look for 1Kx8 and 2Kx4 devices to be released in 1976. For FPLAs, look for at least three new devices this year.

TABLE I

The Programmer People

DATA I/O
Post Office Box 308
1297 N.W. Mall
Issaquah, Washington 98027
(206) 455-3990

COMPARISON CHART
Personality Cards & Socket Adapters
to
Manufacturers' pROMs

Configuration	Manufacturers' Part No. Bold Face = 3 State	Data I/O Program Card Set	Program Socket Adapter	Pro-grammed Logic Level	Read-Only Options: Read-Only Card	Read-Only Options: Read-Only Socket Adapter
ADVANCED MICRO DEVICE		(909-XXXX)	(715-XXXX)		(701-XXXX)	(715-XXXX)
32X8 (FL)	AM27S08, **AM27S09**	1119-1	1034	VOL	1142	1037
256X4 (FL)	AM27S10, **AM27S11**	1176-1	1034	VOL	1142	1027-★
256X8 (MOS)	**1702/AM9702**	1183-1	1047	VOL	1187-6	1033
256X8 (MOS)	**1702A**	1183-1	1047	VOH	1187-6	1033
AMERICAN MICROSYSTEMS						
512X8 (MOS)	**S6834**	1230-1	1033	VOH	1187-14	1033
512X8 (MOS)	**S5204-B**	1230-1	1038	VOH	1187-14	1038
FAIRCHILD SEMICONDUCTOR						
256X4 (FL)	93416, **93426**	1055-9	1034	VOL	1142	1027-★
256X4 (FL)	93417, **93427**	1063-2•	1027-1	VOL	1142	1027-★
512X4 (FL)	93436, **93446**	1063-2•	1027-2	VOL	1142	1027-★
512X8 (FL)	93438, **93448**	1063-2•	1033-2	VOL	1142	1033
256X4 (FL)	10416 ECL	1144-3	1034	VOH	1187-13	1034
HARRIS SEMICONDUCTOR						
32X8 (FL)	HPROM 8256	1051-4	1034	VOH	1142	1037
64X8 (FL)	HPROM 0512	1054-3	1033	VOH	1018	1033
256X4 (FL)	HPROM 1024A, **1024**	1055-3	1034	VOL	1142	1027-★
32X8 (FL)	HM7602/**7603**	1063-4•	1037	VOL	1142	1037
256X4 (FL)	HM7610/**7611**	1063-4•	1027-1	VOL	1142	1027-★
512X4 (FL)	HM7620/**7621**	1063-4•	1027-2	VOL	1142	1027-★
512X8 (FL)	HM7640/**7641**	1063-4•	1033-2	VOL	1142	1033
1024X4 (FL)	HM7642/**7643**	1063-4•	1039	VOL	1142	1039
1024X4 (FL)	HM7644	1063-4•	1042	VOL	1142	1042
ALL (FL)	Diode Matrix	1189	1034	OPEN		
INTEL CORP.						
256X4 (FL)	3601/3601-1	1004	1034	VOH	1142	1027-★
256X4 (FL)	**3621**	1170-1•	1027-1	VOL	1142	1027-★
512X4 (FL)	3602, **3622**	1170-1•	1027-2	VOL	1142	1027-★
512X8 (FL)	3604, **3624**	1170-1•	1043-1	VOL	1142	1043-★
512X8 (FL)	3604-6	1170-1•	1043-2	VOL	1142	1043-★
1024X4 (FL)	3605	1170-1•	1039	VOL	1142	1039
256X8 (MOS)	**1702A/4702A/8702A**	1183-1	1047	VOH	1187-6	1033
512X8 (MOS)	**2704/8704**	1174-1	1033	VOL	1187-10	1033
1024X8 (MOS)	**2708/8708**	1174-1	1033	VOL	1187-10	1033
INTERSIL						
32X8 (AIM)	5600, **5610**	1050-1	1002	VOH	1142	1037
256X4 (AIM)	5603A, **5623**	1050-2	1003	VOH	1142	1027-★
512X4 (AIM)	5604, **5624**	1050-2	1003	VOH	1142	1027-★
512X8 (AIM)	5605, **5625**	1058-1	1033-2	VOH	1142	1033

Configuration	Manufacturers' Part No. Bold Face = 3 State	Data I/O Program Card Set	Program Socket Adapter	Program Logic Level	Read-Only Options: Read-Only Card	Read-Only Options: Read-Only Socket Adapter
		(909-XXXX)	(715-XXXX)		(701-XXXX)	(715-XXXX)
MONOLITHIC MEMORIES						
32X8 (FL)	5330/6330, **5331/6331**	1226-1•	1046	VOL	1142	1046
256X8 (FL)	5335/6335, **5336/6336**	1226-1•	1033-1	VOL	1142	1033
256X4 (FL)	5300/6300, **5301/6301**	1226-1•	1035-1	VOL	1142	1027-★
512X4 (FL)	5305/6305, **5306/6306**	1226-1•	1035-2	VOL	1142	1027-★
512X8 (FL)	5340/6340, **5341/6341** Note: All above are new "-1" series pROMs.	1226-1•	1033-2	VOL	1142	1033
1024X4 (FL)	5350/6350, **5351/6351**	1226-1•	1036	VOL	1142	1036
1024X8 (FL)	5380/6380, **5381/6381**	1226-1•	1033	VOL	1142	1033
MOTOROLA						
64X8 (FL)	2-5003	1054-2	1033	VOH	1018	1033
256X4 (FL)	10149 ECL	1144-2	1034	VOL	1187-13	1034
NATIONAL CASH REGISTER						
256X4 (MNOS)	1105	1053	1033	VOL	1142	1033
1024X4 (MNOS)	2400	1056	1033	VOL	1181	1033
NATIONAL SEMICONDUCTOR						
32X8 (FL)	7577/8577, **7578/8578**	1051-9	1034	VOL	1142	1037
256X4 (FL)	7573/8573, **7574/8574**	1055-12	1034	VOL	1142	1027-★
256X8 (MOS)	**4203/5203Q/5202**	1178-1	1047	VOL	1187-8	1033
256X8 (MOS)	**1702A**	1183-1	1047	VOH	1187-6	1033
512X8 (MOS)	**4204/5204**	1177-1	1033	VOH	1187-1	1033
NEC						
256X4 (AIM)	403D	1058-2	1027-1	VOH	1142	1127-★
SIGNETICS CORP.						
32X8 (FL)	8223	1051-1	1034	VOH	1142	1037
32X8 (FL)	10139 ECL	1051-2	1034	VOH	1142	1037
32X8 (FL)	82-S-23, **82-S-123**	1051-7	1034	VOH	1142	1037
256X4 (FL)	82-S-126, **82-S-129**	1055-10	1034	VOH	1142	1027-★
256X8 (FL)	**82-S-114**	1145-2•	1032	VOH	1187-3	1032
512X8 (FL)	**82-S-115**	1145-2•	1033	VOH	1187-3	1033
256X4 (FL)	10149 ECL	1144-1	1034	VOH	1187-13	1034
512X4 (FL)	82-S-130, **82-S-131**	1055-10	1034	VOH	1142	1027-★
TEXAS INSTRUMENTS						
32X8 (FL)	74188A	1063-3•	1037	VOH	1142	1037
32X8 (FL)	74S188, **74S288**	1063-3•	1037	VOH	1142	1037
64X8 (FL)	74186	1054-1	1033	VOH	1018	1033
256X4 (FL)	74S387, **74S287**	1055-5	1034	VOL	1142	1027-★
256X8 (FL)	74-S-470, **74-S-471**	1063-3•	1028-1	VOH	1142	1028-★
512X8 (FL)	74-S-473, **74-S-472**	1063-3•	1028-2	VOH	1142	1028-★
ALL (FL)	Diode Matrix	1189	1034	OPEN		
TOSHIBA						
512X4 (MOS)	**T3181**	1185-1	1033	VOL	1187-15	1033

NOTES: 1) Shielded cable, P/N 709-1012, is required for use with all socket adapters.
2) ALL -1 socket adapters have a 256 word limit.
3) ALL -2 socket adapters have a 512 word limit.
4) Use single socket receptacle P/N 715-1026, or dual socket receptacle, P/N 715-1029, for all adapters.
5) • Generic Program Cards
6) ★ Dash number does not matter

11. LOGIC STATE ANALYZERS GAIN WIDESPREAD ACCEPTANCE AS MICROPROCESSOR DEBUGGING TOOL

BRUCE FARLY
Hewlett Packard Co.
Colorado Springs, Colorado

Microprocessors, for all their blessings, have added many new complications to the development of equipment and systems.

Ability to debug microprocessor-based designs is becoming of major importance to the design engineer. In particular, the engineer needs to see in real time precisely what is happening in digital systems in an easy-to-read display format. Because they are not dedicated, are of relatively low cost, expandable and capable of being interfaced easily to anything from a breadboard model to a field-installed system, logic state analyzers are finding widespread acceptance as the answer to debugging needs.

For sophisticated applications, the real-time nature of subtle software/hardware interactive problems dictates the need for a measurement tool with sufficient capabilities to meet a wide variety of needs. For example, the 1600A includes a standard 16 word by 16 bit table display, a method of displaying an active as well as a stored table for comparative troubleshooting, an exculsive OR display where the table displays data differences and a map display that provides an overall, macroscopic view of machine operations.

More Than 16 Bits

However, for applications requiring more than 16 bits, two pieces of equipment, such as the 1600A and the 1607A must be combined to provide a 32-bit-wide machine. In this configuration, called a 1600S, both single and dual clock operations allow easier solving of more complex problems such as those found in input/output debug application.

For applications of lesser sophistication, or for a designer wanting to get started with a minimum investment, a lower-cost logic state analyzer such as the 1607A can be interfaced with a standard oscilloscope.

Few would question that real-time measurement of a system under test is far superior to simulation or emulation. In fact, the latter are usually done as a convenience when a system is being designed and not all components are yet running, or are not controllable unless simulated. Performance verification must be done on a system in real time and in a non-interactive mode. (That is, the measurement tool should not affect the measurement itself.) It cannot, therefore, be a part of the system sharing either hardware or software.

By employing logic state analyzers, real-time, non-interactive measurements can begin just as soon as the breadboard is built and can be carried on throughout development, production and field service cycles. Hardware and software can be "brought up" together with the simulation phases totally eliminated, if the designer so wishes.

A typical example of the help that a logic-state analyzer can provide is shown in Fig. 1. This photograph shows a single-shot capture in real time of a Motorola M6800 microprocessor responding to an external interrupt signal. The 16 channels on the left are displaying consecutive addresses, while the other 10 channels from right to left are displaying eight-bit data bus, enable ($VMAO_2$) signal, add read/write line. The brightened trigger word, hexadecimal FFFC was positioned to provide 13 words before trigger and two after to verify that the microprocessor responded correctly to the interrupt. Note there are seven consecutive "write" operations (R/W line low) as the external stack is loaded with the internal CPU registers. The address lines show that the stack pointer begins loading at hex address 07FF with the lower byte of the program counter. The last operations are to read the vector address stored at FFFC and FFFD as 12 and IC, respectively, and begin execution of the program at that location.

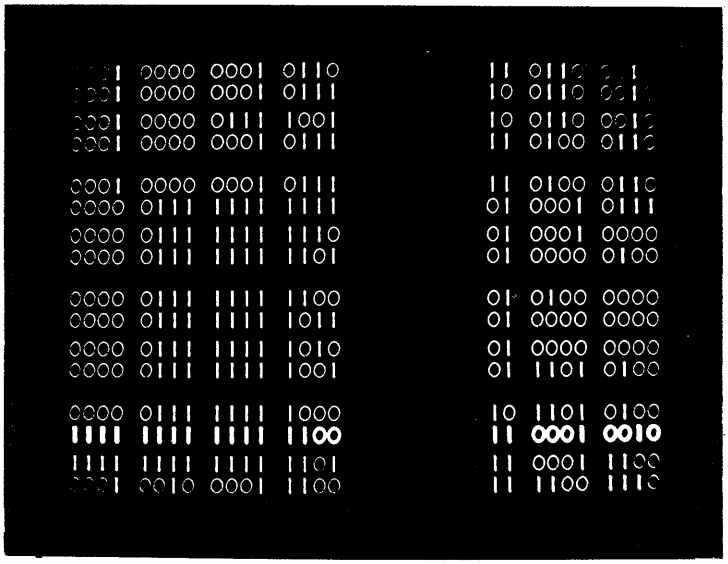


Fig. 1. Typical HP 1600A output

12.

CHARGE-COUPLED DEVICES

DAVE HOUSE, JIM OLIPHANT AND BOB PAPENBERG
Intel Corp.
Santa Clara, California

Until recently, the charge-coupled device (CCD) was considered either a laboratory curiosity or, at best, a memory device of limited application in a few relatively low-volume analog applications. This is no longer the case.

During 1975, four years after the demonstration of operating devices, the CCD completed the transition from research lab to production line. At present, digital storage products are being, or have already been, designed using CCD devices.

The forerunner of the CCD class of devices, the 2416, serves as an example of the capability these devices will offer equipment and systems designers. This serial memory is configured as 64 independent recirculating shift registers of 256 bits each. Any one of the 64 internal shift registers can be accessed in a random fashion by applying the appropriate code to the six address inputs.

The internal memory organization combines both serial and random address memory functions. As shown in Fig. 1, the IC is arranged as 64, 256-bit CCD shift registers. The data in these registers is simultaneously shifted by exercising the four-phase clock signals O1 through O4. After a shift cycle, each of the 64 CCD registers can be selected for an input/output (I/O) function by applying the appropriate six-bit address code and applying chip-enable, chip-select and write-enable signals in the required manner.

The organization of the IC is depicted in Fig. 2. In this diagram, the CCD is visualized as a cylinder comprised of 64 "tracks" (the 64 CCD recirculating shift registers) with each track divided into 256 "sectors" (the 256 CCD data storage cells). The "rate of rotation" of the cylinder is controlled by the four-phase clocks and is in the direction indicated by the "shift direction" arrow shown in Fig. 2. The four-phase clocks always shift the cylinder in the same direction, and cannot be manipulated to reverse the shift direction.

Read/write capability in the CCD is performed by 64 bidirectional data buffers (one data buffer per track). These buffers are identified in position. A shown in Fig. 2 as the shaded column. The cylinder is considered to rotate through the buffers so that each shift of the cylinder, which is controlled by the four-phase clocks, places the next sequential sector of each track "in" the buffer. The buffers shown in column A also provide a refresh to each cell in addition to performing read/write functions. An additional refresh-only buffer is shown in column B of Fig. 2. These buffers are located halfway around the cylinder as shown.

Two basic addressing methods may be used to store data words: storage in a given sector, or storage around a given track.

In the first named method, the desired word is accessed by shifting the cylinder, using the four phase clocks, until the sector (0-255) containing the word is coincident with the read/write buffers (shown as column A). The word is then accessed one bit at a time by addressing the appropriate track with addresses A_0-A_5. This addressing technique is exemplified in the four bit memory word N shown in Fig. 2.

The second addressing method places a word sequentially around the cylinder in a given track. Access to a particular word requires both a four-phase lock shift followed by a data access cycle for each bit of the word. For this case, A_0-A_5 do not change once the desired track is accessed. An example of this addressing technique is shown as four-bit memory word M in Fig. 2.

Because of system addressing problems, it is not generally desirable to combine the two addressing methods at once although this is possible. Addressing by sector is usually the preferred technique. A major advantage of this type of data organization is low four-phase clock driver power required to achieve the maximum serial data transfer rate of 2 megabits/s from a single IC. In most serial applications, the four-phase clock signals only. have to operate at less than a 55-kHz rate to obtain a 2-MHz I/O data rate, because the clocks are used solely to shift/refresh data, not to perform I/O functions. For each shift of the clock, 64 "new" data bits are available in the 64 internal data registers for access through the address, chip-enable and read/write control signals. These data-control signals are very easy to drive due to low input capacitance.

An alternate method of visualizing the organization of the 2416 is shown in Fig. 3. This diagram is derived from the cylinder in Fig. 2, imagining that the cylinder is cut along the line marked C (between sector 0 and 255) and laying the cylinder out flat.

There are two common types of CCD devices, surface-channel and buried-channel. The surface-channel devices stores and transfers charge (data) along the surface of the substrate. The buried-channel type, because of additional substrate doping, stores and transfers the charge (data) further into the bulk of the substrate.

There are some primary differences between surface-channel and buried-channel devices. Surface-channel CCDs have higher charge-carrying capability and lower charge-transfer efficiency at extremely high charge-transfer rates. However, the loss of charge-transfer efficiency occurs at a frequency much higher than the maximum shift-frequency of the 2416. Charge transfer efficiency if defined as the percentage of the total charge packet (data) actually shifted or transferred per shift. Efficiency is typically greater than 99.9 per cent per shift. Surface-channel devices are also more simply fabricated.

The internal memory array is comprised of four-phase, surface-channel, charge coupled structures. The CCD structure is formed by a series of MOS thin field gate oxide devices placed

as shown in fig. 4. These MOS devices do not have the source/-drain diffusions usually associated with other MOS structures. Figure 4 (a), a top view of the storage array, illustrates that the clock phases are laid out perpendicular to the shift-register channels. Electrical isolation between shift-register channels is obtained by channel-stop diffusion and thick-film oxide methods. Data input/output connections to the registers are obtained from n+ diffusions at the ends of the registers.

The CCD stores data as charge, as do all dynamic MOS memories. In many respects, the storage mechanism of the 2416 is very similar to the 4096-bit rams implemented with single transistor cells, such as Intel's 2107B. The storage element may be thought of as resembling a "potential well." This potential well is formed when a positive voltage potential is applied on the clock gates. The positive voltage repels the majority substrate carriers (holes) from the vicinity of the gauge and forms a charge-depletion area under it. This depleted region can accept and store a negative charge packet as long as the gate forming the well remains sufficiently positive with respect to the substrate.

The CCD structure is inherently dynamic, so it must be refreshed periodically to maintain data. The dynamic nature of a CCD results from thermally generated carriers (traditionally called "dark current effect") which act to fill an uncharged potential well, thereby changing that particular cell's logic state.

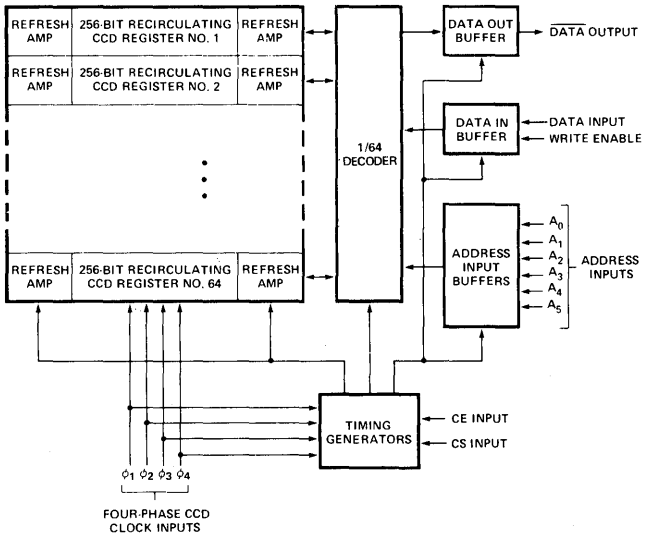


Fig. 1

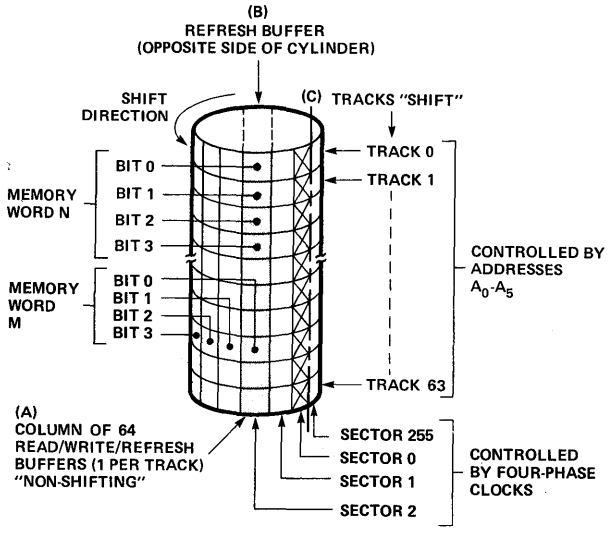


Fig. 2

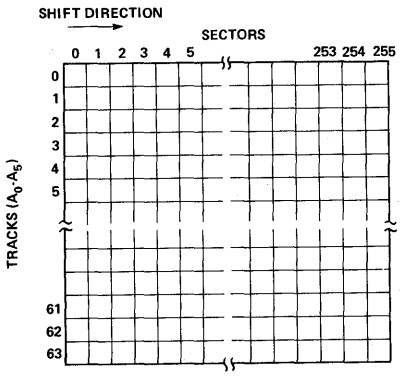


Fig. 3

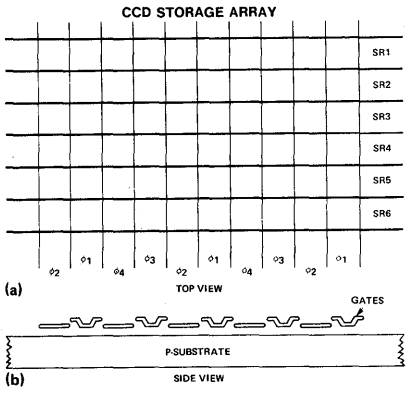


Fig. 4

13.

A MICROCOMPUTER DESIGNED FOR CONTROL

MICHAEL A. LICCARDO
Scientific Microsystems
Mountain View, CA

1. INTRODUCTION

1.1 Features

The SMS MicroController is a microcomputer designed for control. It features:

Execution Speed

- 300 nanosecond instruction time.
- Direct address capability - up to 8192 16-bit words of program memory. Current systems allow maximum of 4096 words of program storage.
- Eight 8-bit general purpose registers.
- Simultaneous data transfer and data edit in a single instruction cycle time.
- n way branch or n entry table lookup in two instruction cycle times
- MicroController instructions operate with equal speed on 1-bit, 2-bit, 3-bit, 4-bit, 5-bit, 6-bit, 7-bit, or 8-bit data formats.

Interface Simplicity and Expandability

- Direct connection to TTL (3-state) I/O (Open Collector outputs are optional).
- I/O expandable to 4096 connection points with storage latch at each point.
- User defined data flow direction with each group of 8 I/O points.

High Density Packaging and Reliable Operation

- The MicroController is implemented completely with LSI circuits.
- The MicroController CPU consists of a single integrated circuit.
- Single +5.0 volt power supply operation.

1.2 Program Versus Data

The storage concept of the MicroController is to separate program storage from data storage. Program storage is implemented in read-only memory in recognition of the fact that programs for control applications are fixed and dedicated. The benefits of using read-only memory are that great speeds may be obtained at lower cost than if read/write memory was used, and that program instructions reside in a non-volatile medium and cannot be altered by system power failures. Data storage for the MicroController is implemented with read/write memory because data in control and other real time applications is dynamic and variable.

The MicroController I/O system is treated as a set of internal registers. Therefore data from external devices may be processed (tested, shifted, added to, etc.) without first moving them to internal storage. In fact, the entire concept is to treat data at the I/O interface no differently than internal data. This concept extends to the software which allows variables at the input/output system to be named and treated in the same way as data in storage.

2. MICROCONTROLLER FUNCTIONAL COMPONENTS

The MicroController is a complete microcomputer system consisting of:

- A central processing unit called the Interpreter.
- Read-only program storage.
- Optional read/write data storage called Working Storage with variable field access to 1 to 8 bits.
- A complete bit addressable input/output system called the Interface Vector.

The MicroController system is shown in Figure 1.1.

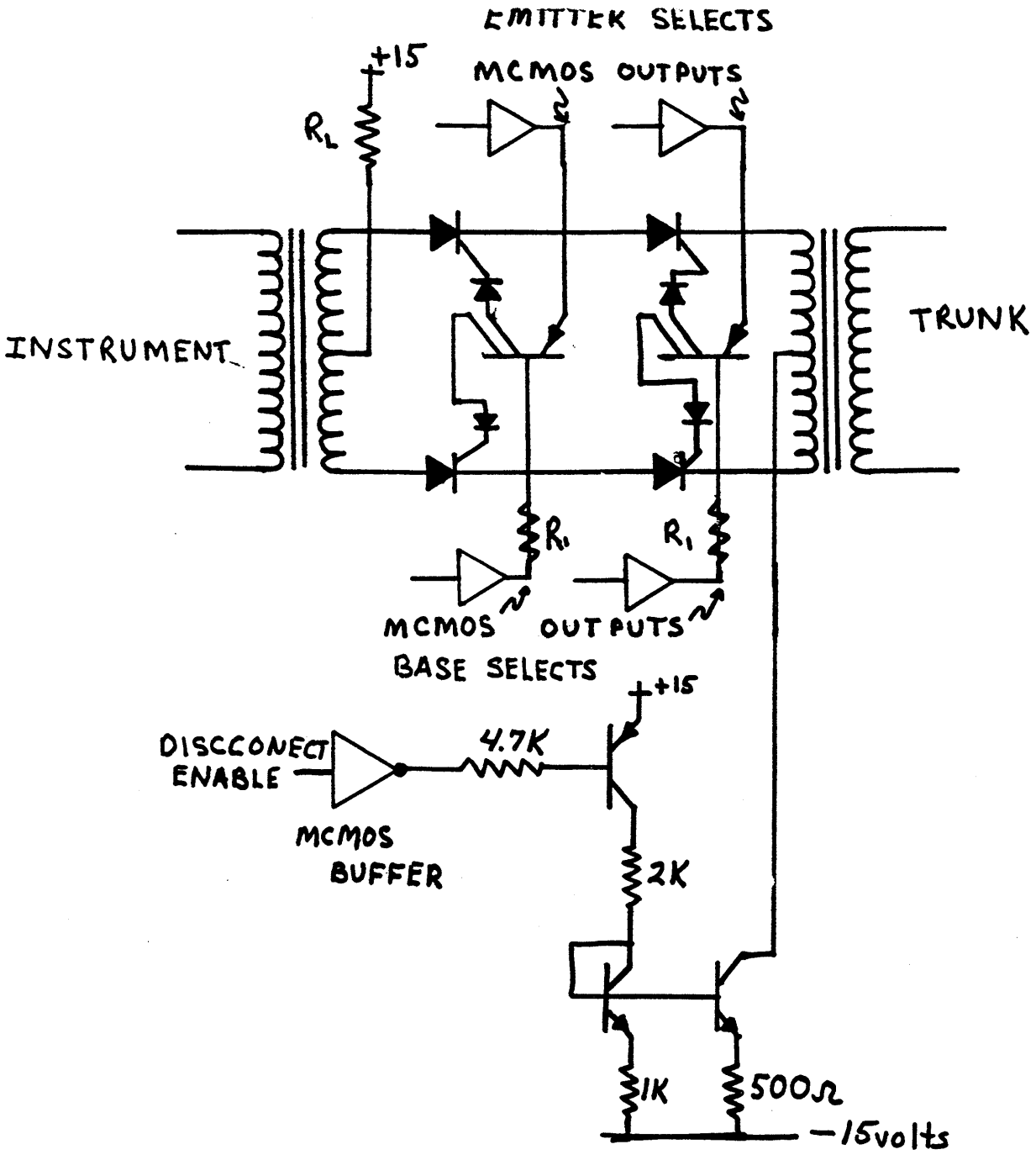
Figure 1.2 illustrates the MicroController architecture. The MicroController CPU contains an Arithmetic Logic Unit (ALU), Program Counter, Address Register, Interface Vector Address Register (IVL), and Working Storage Address Register (IVR). Eight 8-bit general purpose registers are also provided, including seven working registers and an auxiliary Register which performs as a working register and also provides an implied operand for many instructions. The MicroController registers are shown in Figure 1.2 and are summarized below:

Control Registers Include:

Instruction - a 16-bit register containing the current instruction.
Program Storage Address Register (AR) - A 13-bit register containing the address of the current instruction being accessed from Program Storage.
Program Counter (PC) - A 13-bit register containing the address of the next instruction to be read from Program Storage.

Write-only "Registers" Include:

IV Byte Address (IVL) - An 8-bit write-only register used to address an IV byte. IVL is used


EMITTER SELECTS
MCMOS OUTPUTS
+15
R_L
INSTRUMENT
TRUNK
R_1
R_1
MCMOS OUTPUTS
BASE SELECTS
+15
DISCCONECT ENABLE
4.7K
MCMOS BUFFER
2K
1K
500Ω
-15volts

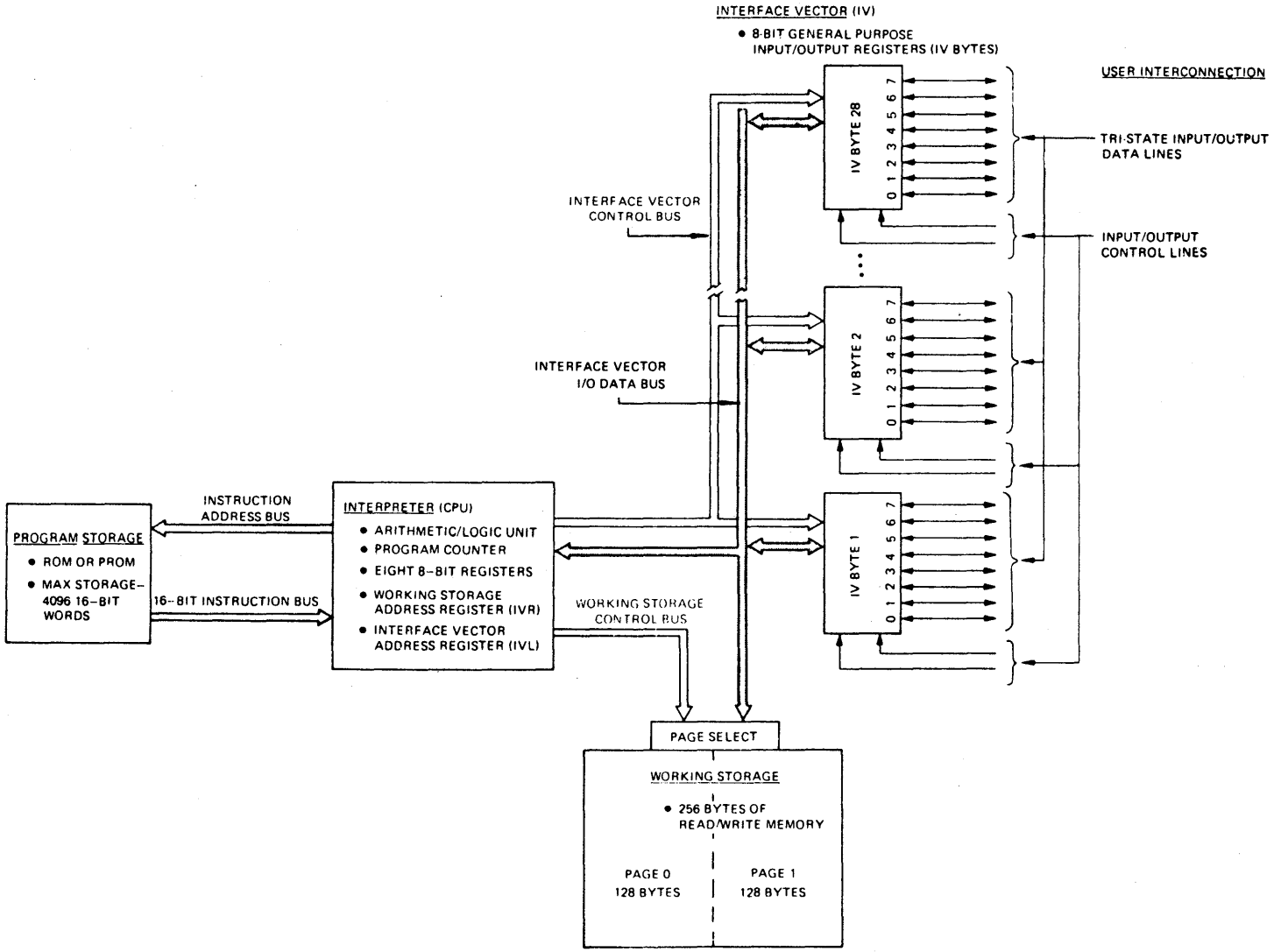


Figure 1.1 MicroController System Diagram

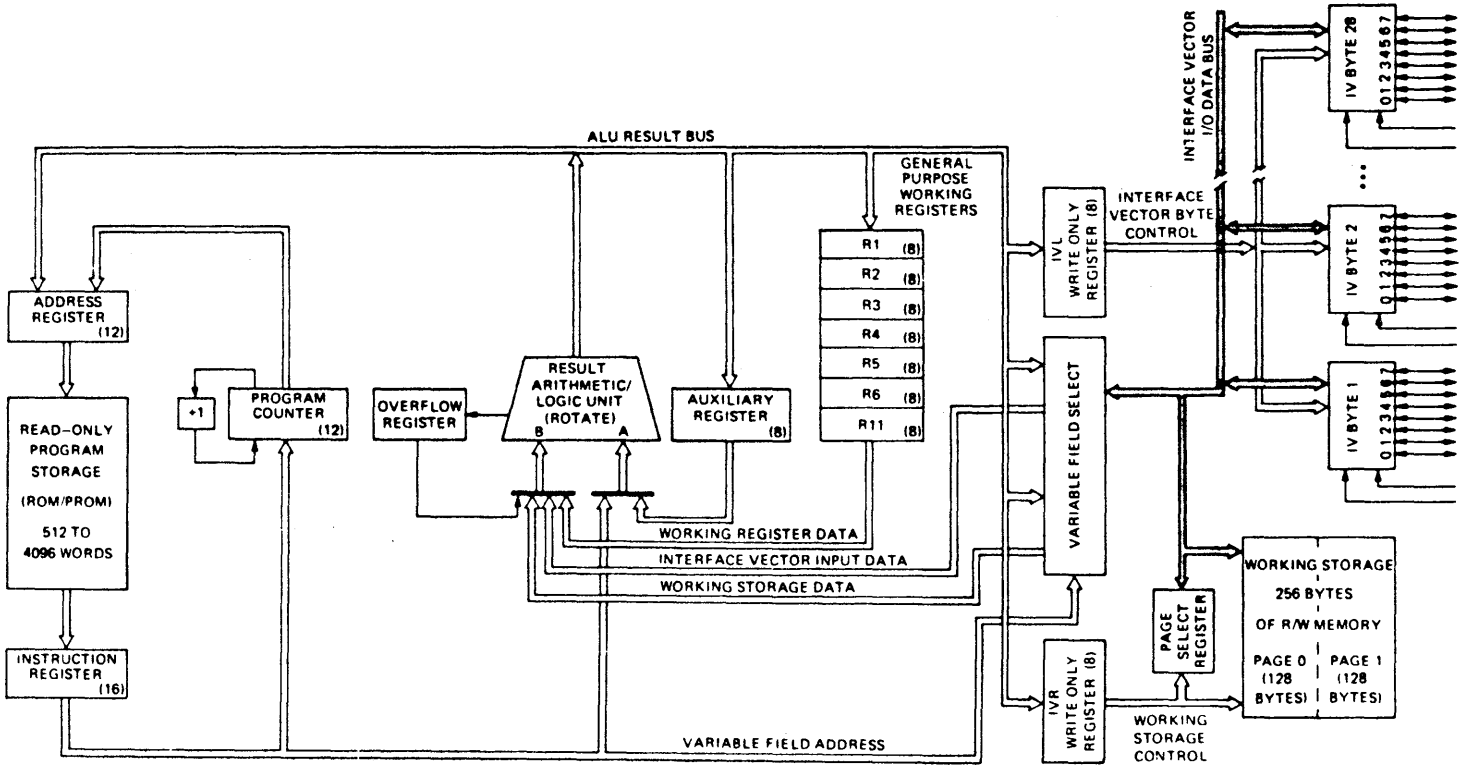


Figure 1.2 MicroController Architecture

only as a destination in an instruction which directly generates an IV byte address.
Working Storage Address (IVR) - An 8-bit write-only register used to address a Working Storage byte. IVR is used only as a destination in an instruction which directly generates a Working Storage byte address.

Data Registers Include:

Working Registers (WR) - Seven 8-bit registers for data storage.
Overflow (OVF) - A 1-bit register that retains the most significant bit position carry from ALU addition operation. Arithmetically treated as 2°.
Auxiliary (AUX) - An 8-bit register. Source of implied operand for arithmetic and logical instructions. May be used as a working register.

A crystal external to the CPU is used to generate the CPU system clock. The CPU executes eight instruction types.

The 16-bit MicroController instructions are stored 512 to 8192 words of read-only Program Storage. Program Storage can be implemented with either mask coded ROMs (Read-Only Memory) or PROMs (programmable Read-Only Memory).

The input/output system, called the interface Vector, serves as the data path over which information is transferred into and out of the MicroController. The basic elements of the Interface Vector are:

- The general purpose 8-bit input/out registers or Interface Vector (IV) Bytes, whose tri-state data path serves as the connection points to the user system.
- The IVL register which addresses an IV Byte.
- Variable field selection which permits 1- to 8-bit field access of a selected IV Byte in a single instruction.

The Interface Vector eliminates the need for costly interface logic and presents a simple, well-defined interconnection point to the user system.

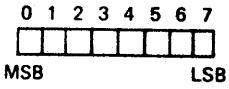
Working Storage is available as an option that provides 256 bytes of read/write memory for program data or input/output data buffering. Working Storage consists of:

- 256-bit bytes of read/write memory organized as two pages, Page 0 and Page 1, of 128 bytes each.
- The Working Storage address register, IVR, which addresses a byte in either Page 0 or Page 1, depending on the state of the Page Select Register.
- The Page Select Register, addressed through IVR, is an 8-bit register used to select Page 0 or Page 1 of Working Storage.
- Variable Field Select which permits 1- to 8-bit field transfers to or from an addressed Working Storage byte in a single instruction.

3. MICROCONTROLLER INSTRUCTION SET

The MicroController has a repetoire of eight instruction classes which allows the user to test input status lines, set or reset output control lines, and perform high-speed input/output data transfers. All instructions are 16 bits in length. Each instruction is fetched, decoded and executed completely in 300 nanoseconds.

Data is represented as an 8-bit byte; bit positions are numbered from left to right, with the least significant bit in position 7.



Within the Interpreter, all operations are performed on 8-bit bytes. The Interpreter performs 8-bit, unsigned, 2's complement arithmetic.

3.1 Instruction Formats

The General MicroController instruction format is:

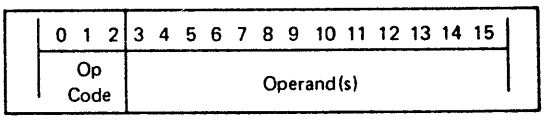


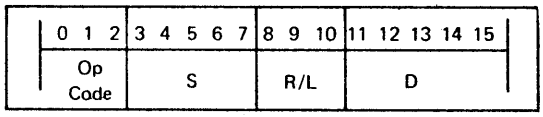
Table 3.1 contains a summary of the MicroController instruction set and description of the operand fields.

All instructions are specified by a 3-bit Operation (Op) Code Field. The operand may consist of the following fields: Source (S) Field, Destination (D) Field, Rotate/Length (R/L) Field, Immediate (I) Operand Field, and (Program Storage) Address (A) Field.

The Instructions are divided into five format types based on the Op Code and the form of the operand(s).

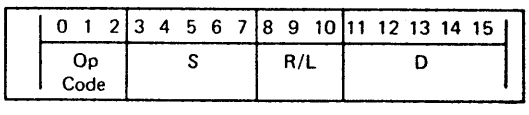
TYPE I

OPERATIONS
(Register to Register)
MOVE AND
ADD XOR



TYPE II

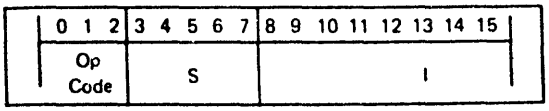
OPERATIONS
MOVE ADD
AND XOR



TYPE III

OPERATIONS

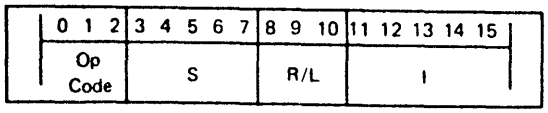
XEC XMIT NZT



TYPE IV

OPERATIONS

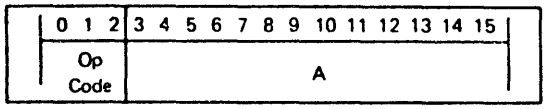
XEC XMIT NZT



TYPE V

OPERATIONS

JMP



3.2 Instruction Fields

Op Code Field - 3-bit Field

The Op Code field is used to specify one of eight MicroController instructions.

OP CODE OCTAL VALUE	INSTRUCTION		RESULT
0	MOVE	S,R/L,D	(S) → D
1	ADD	S,R/L,D	(S) plus (AUX) → D
2	AND	S,R/L,D	(S) Λ (AUX) → D
3	XOR	S,R/L,D	(S) ⊕ (AUX) → D
4	XEC	I,R/L,S or I,S	Execute instruction at current PC offset by I + (S)
5	NZT	I,R/L,S or I,S	Jump to current PC offset by I if (S) ≠ 0
6	XMIT	I,R/L,S or I,S	Transmit literal I → S
7	JMP	A	Jump to program location A

S,D Fields - 5-bit Fields

The S and D fields specify the source and destination of data for the operation defined by the Op Code Field. The Auxiliary Register is the implied source for the instructions ADD, AND and XOR which require two source fields. That is, instructions of the form:

ADD X,Y

implies a third operand, say Z, located in the Auxiliary Register so that the operation which takes place is actually X + Z, with the result stored in Y. This powerful capability means that three operands are referenced in 300 nanoseconds.

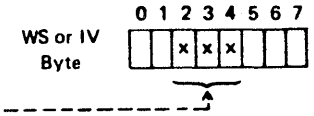
The S and/or D fields may specify a register, or a 1- to 8-bit I/O field, or a 1- to 8-bit Working Storage field. S and D field value assignments in octal are shown in Table 3.2.

R/L Field - 3-bit Field

The R/L field performs one of two functions. specifying either a field length (L) or a right rotation (R). The function it specifies for a given instruction depends upon the contents of the S and D fields:

A. When both S and D specify registers, the R/L field is used to specify a right rotation of the data specified by the S field. (Rotation occurs on the bus and not in the source register.) The register source data is right rotated within one instruction cycle time independent of the number (0 to 7) of bit positions specified in the R/L field.
B. When either or both the S and D fields specify an IV or Working Storage data field, the R/L field is used to specify the length of the data field (within the byte) accessed, as shown below:

OCTAL VALUE	SPECIFICATION
0	Field length = 8 bits
1	Field length = 1 bit
2	Field length = 2 bits
3	Field length = 3 bits
4	Field length = 4 bits
5	Field length = 5 bits
6	Field length = 6 bits
7	Field length = 7 bits



I Field - 5/8-bit Field

The I field is used to load a literal value (a binary value contained in the instruction) into a register, IV or Working Storage data field or to modify the low order bits of the Program Counter.

The length of the I field is based on the S field in XEC, NZT, and XMIT instructions:

A. When S specifies a register, the literal I is an 8-bit field (Type III format).
B. When S specifies an IV or Working Storage data field, the literal I is 50bit field (Type IV Format).

A field - 13-bit Field

The A field is a 13-bit Program Storage address field. This allows MicroController Systems to directly address 8192 instructions, although current offerings are limited to 4096 instructions.

TABLE 3.1

MICROCONTROLLER INSTRUCTION SUMMARY

OPERATION	FORMAT	RESULT	NOTES
MOVE	Type I Type II	Content of data field addressed by S, R/L replaces data in field specified by D, R/L.	If S and D both are register addresses then R/L specifies a right rotate of R/L places applied to the register specified by S.
ADD		Sum of AUX and data specified by S, R/L replaces data in field specified by D, R/L.	
AND		Logical AND of AUX and data specified by S, R/L replaces data in field specified by D, R/L.	
XOR		Logical exclusive OR of AUX and data specified by S, R/L replaces data in field specified by D, R/L.	
XMIT	Type III Type IV	The literal value I replaces the data in the field specified by S, R/L.	If S is IV or WS address then I limited to range 00-37. Otherwise I limited to range 000-377.
NZT		If the data in the field specified by S, R/L equals zero, perform the next instruction in sequence. If the data specified by S, R/L is not equal to zero, execute the instruction at address determined by using the literal I as an offset to the Program Counter.	If S specifies an IV or WS address then I is limited to the range 00 - 37. I is limited to the range 000 - 377 otherwise.
XEC		Perform the instruction at address determined by applying the sum of the literal I and the data specified by S, R/L as an offset to the Program Counter. If that instruction does not transfer control, the program sequence will continue from the XEC instruction location.	The offset operation is performed by reducing the value of PC to the nearest multiple of 32 (if I : 00 - 37) or 256 (if I : 000 - 377) and adding the offset.
JMP	Type V	The literal value I replaces contents of the Program Counter.	I limited to the range 00000 - 07777.

TABLE 3.2

S AND D FIELD SPECIFICATION

Register Specification

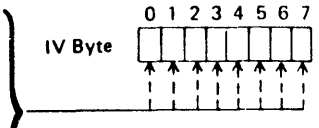
0_8 - 17_8 is used to specify one of seven working registers (R1 - R6, R11), Auxiliary Register, Overflow Register, IVL and IVR write-only registers.

OCTAL VALUE	
00	Auxiliary Register (AUX)
01	R1
02	R2
03	R3
04	R4
05	R5
06	R6
07	IVL Register - IV Byte address write-only register - Specified only in S field of XMIT instruction, or in D field in all other instructions.
10	OVF - Overflow register - Used as an S (source) field only.
11	R11
12	Unassigned
13	Unassigned
14	Unassigned
15	Unassigned
16	Unassigned
17	IVR Register - Working Storage address write-only register - Specified only in S field of XMIT instruction, or in D field in all other instructions.

I/O Field Specification

20_8 - 27_8 is used to specify the least significant bit of a variable length field within the IV Byte previously selected by the IVL register. The length of the field is determined by R/L.

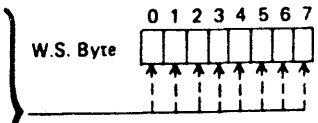
OCTAL VALUE	
20	Field within previously selected IV Byte; position of LSB = 0
21	" = 1
22	" = 2
23	" = 3
24	" = 4
25	" = 5
26	" = 6
27	" = 7



Working Storage Field Specification

30_8 - 37_8 is used to specify the least significant bit of a variable length field within the Working Storage Byte previously selected by the IVR Register. The length of the field is determined by R/L.

OCTAL VALUE	
30	Field within previously selected W.S. Byte; position of LSB = 0
31	" = 1
32	" = 2
33	" = 3
34	" = 4
35	" = 5
36	" = 6
37	" = 7



3.3 Register Operations

When a register is specified as the source, and an IV or Working Storage field as the destination, the least significant bits of the operations (MOVE, ADD, AND, XOR) result are stored. The operation is performed on the entire 8-bit source for a MOVE, or between the 8-bit AUX and the source register for ADD, AND, XOR operations. The least significant bits of the result are stored in the IV or Working Storage data field specified by the D and R/L fields in the instruction.

When an IV or Working Storage field of one to eight bits is specified as the source, and a register as the destination, the 8-bit result of the operation (MOVE, ADD, AND, XOR) is stored in the register. The operations ADD, AND, XOR actually use the IV or Working Storage data field (1-8 bits) with leading zeros to obtain 8-bit source data for use with the 8-bit AUX data during the operation.

Because IVL and IVR are write-only pseudo registers, they can be specified as destination fields only (see Table 3.2). Operations involving IVL and IVR as sources are not possible. For example, it is not possible to increment IVR or IVL in a single instruction, and the contents of IVL or IVR cannot be transferred to a working register, IV Byte, or Working Storage location.

The OVF (Overflow) Register can only be used as a source field; it is set or reset only by the ADD instruction.

4. INPUT/OUTPUT SYSTEM

As seen from previous sections, the Interface Vector is the MicroController's input/output system. It provides a simple interconnection to the user status, control and data lines.

4.1 Addressing Data on the Interface Vector

The Interface Vector is comprised of general purpose 8-bit I/O registers called Interface Vector (IV) Bytes. In the present MicroController offering, the Interface Vector may consist of up to 28 Bytes.

As seen in Figure 1.2, the IVL register serves to select IV Bytes. In order for an instruction to access (read or write) an IV Byte, the address of that byte must be output to the IVL register.

Thus, two instructions are required to operate on an Interface Vector byte:

```
XMIT      ADDRESS, IVL

MACHINE INSTRUCTION
```

Once the IV Byte is selected (addressed), it will remain selected until another address is output to IVL. Since IVL can be used only as a destination field of an instruction, any instruction sending data to IVL can be used to select an IV Byte.

From the user's standpoint, however, all IV Byte outputs can be read by an external device regardless of whether they are selected or not.

Although the address range of IVL is 0 - 377_8, only 28 IV Bytes are available on current system offerings. The addressing for the 28 IV Bytes is 01_8 to 34_8.

4.2 Electrical Characteristics of the Interface Vector

Each IV Byte consists of 8 storage latches which hold data transferred between the Interpreter and the User System, 8 tri-state input/output lines and two inputs/output control lines, called Byte Input Control (BIC) and Byte Output Control (BOC) (Figure 4.1). The control lines functions are summarized in Table 4.2.

TABLE 4.1

FUNCTIONS OF THE BIC AND BOC LINES

CONTROL LINES		FUNCTION
BOC (low true)	BIC (low true)	
H	H	8 I/O lines in high impedance state - disable
L	H	8 I/O lines in output mode - 8 bit storage latch data available in the output lines.
X	L	8 i/O lines in input mode - data can be read by Interpreter.

Table 4.2 contains a summary of the electrical characteristics of the IV Byte.

BIOGRAPHY

Michael A. Liccardo is the Product Manager for Scientific Micro Systems. He is responsible for product planning, application support, and market development for microcomputer system products. He has previous experience in minicomputer system design, software development, and computer interface design. He received a B.S.E.E. and an M.S.E.E. from University of California (Berkeley) and an M.B.A. from Stanford University.

Table 4.2

IV BYTE TERMINAL ELECTRICAL CHARACTERISTICS

CHARACTERISTIC	SYMBOL	LIMITS			UNITS	CONDITIONS
		MIN	TYP	MAX		
"1" Input Current*	I_{1IN}			100	uA	V_{1IN} = 5.5 V
"0" Input Current*	$I_{\emptyset IN}$			–800	uA	$V_{\emptyset IN}$ = 0.50 V
"1" Input Voltage	V_{1IN}	2		5.5	Volts	
"0" Input Voltage	$V_{\emptyset IN}$	–1		0.8	Volts	
Input Clamp Voltage	V_{CIN}			–1	Volts	$I_{\emptyset IN}$ = –5ma
High Output Voltage	V_{1OUT}	2.4			Volts	I_{1OUT} = 1ma
Low Output Voltage	$V_{\emptyset OUT}$			0.5	Volts	$I_{\emptyset OUT}$ = –16ma
Output Short Circuit Current	I_{SO}	–20		–200	ma	$V_{\emptyset OUT}$ = 0 V
Data Input Capacitance	C_{IN}			12	pf	$V_{\emptyset IN}$ = 0 V

* Input current is always present regardless of the state of BIC and BOC.

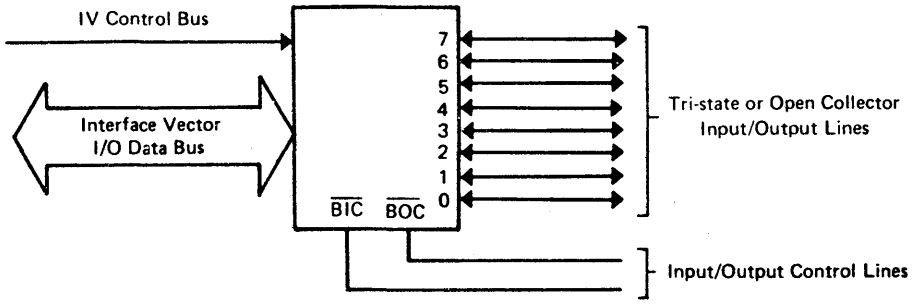


FIGURE 4.1

IV BYTE PROVIDING DYNAMICALLY DEFINED DATA FLOW

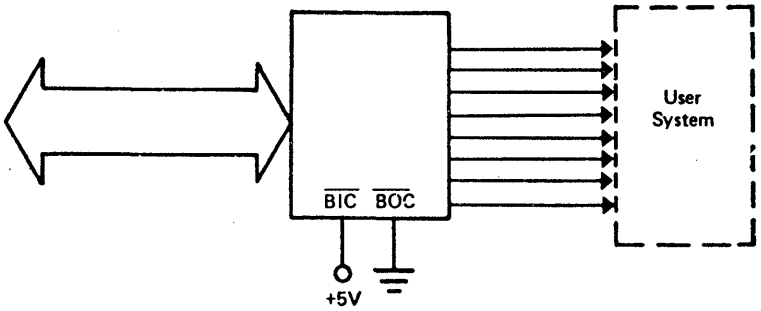


FIGURE 4.2

IV BYTE WIRED FOR USER OUTPUT ONLY

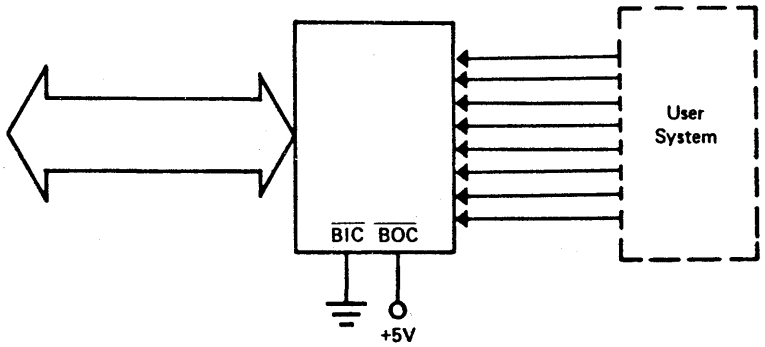


FIGURE 4.3

IV BYTE WIRED FOR INPUT ONLY

A MICROCOMPUTER-BASED CRT TERMINAL

J.E. Bass
Rockwell International Corporation
Microelectronic Device Division
3310 Miraloma Avenue
Anaheim, California 92803

INTRODUCTION

The use of microcomputers as the central control function for CRT terminals offers significant product cost reduction possibilities. While many present CRT products use microcomputers, most of these are used only in the peripheral functions of the system. This paper describes a 1920-character intelligent CRT terminal which uses the Rockwell PPS-8 microcomputer as the primary means of implementation. All system functions are implemented with the PPS-8, including keyboard servicing, CRT refresh from PPS RAM memory, editing, printer interfacing, and communications interfacing.

The paper first describes the functional characteristics of the CRT terminal. Secondly, the PPS-8 microcomputer components are described. Finally, the implementation of the CRT terminal is described in detail.

TERMINAL FUNCTIONAL CHARACTERISTICS

The CRT terminal uses a standard CRT monitor for display. A format of 24 lines of characters with 80 characters per line is provided. Data may be entered through the data entry type alpha-numeric keyboard; external messages may be received through an optional 1200 BPS modem chip. High speed modems may be used in conjunction with our USART-type chip, the Serial Data Controller.

The edit functions include character insert/delete and line insert/delete. Editing is implemented through cursor control in conjunction with cursor control keys and associated function keys on the keyboard. Line Tab, Display Protect, and Line Scroll functions are also provided. A multi-position cursor is provided so that an entire word or phrase may be deleted in one operation instead of one character at a time as is normally done. The display refresh rate is 60 cycles per second, and the monitor is operated in a non-interlaced mode.

Output functions include optional telecommunications and serial party line operations as well as printer options. Telecommunications options provide from 1200 to 9600 BPS operation. The 1200 BPS modem is a single chip configuration with an external analog filter. Printer options consist of 10 to 150 character-per-second alpha-numeric speeds with MOS controllers for full printer control. Higher speed line printers may be interfaced through our PDC I/O chip but require discrete control logic.

The intent of the paper is to illustrate the flexibility of the PPS-8 system in this type of application, and not to describe a finished commercial product. This exercise was undertaken basically for applications know-how. All basic software and hardware design techniques are available to our PPS customers through the Rockwell Applications Engineering staff.

THE PPS-8 MICROCOMPUTER SYSTEM

The PPS-8 microcomputer system is an integrated set of MOS/LSI chips designed primarily for intelligent terminal types of applications. The set is implemented in low cost P-channel technology; system speeds equivalent to, or surpassing, N-channel microcomputers is achieved through system organization. The Rockwell system organization employs a form of distributed processing in which all system chips contain various degrees of intelligence, depending upon their particular function. Some perform dedicated functions, such as printer control or display control; others are programmable through control bytes loaded by the CPU.

We call this approach the Parallel Processing System (PPS). In the Parallel Processing concept, multiple tasks may be executed concurrently in the system. Some tasks will be executed in software; some will be executed in the firmware or dedicated hardware of the intelligent I/O controllers. The CPU becomes the system executive, setting up tasks and assigning them out to the associated controllers. After short data buffer transfers to or from the I/O

THE PPS-8 MICROCOMPUTER SYSTEM (cont.)

controllers, the CPU is free to attend to the higher level task of the executive software. Normally, detailed CPU intervention in the routine I/O functions of keyboard servicing, display servicing, and block transfer of data into or out of the system is not required. With this approach, system performance may be several times that of CPU-oriented microcomputer alternatives.

A brief discussion of each chip used in the CRT Terminal follows. For a full description of all I/O chips available with the PPS-8 microcomputer, please refer to our marketing literature.

CPU

The CPU provides 109 instructions and operates with a four-microsecond instruction cycle. An instruction cycle includes both the instruction fetch and the instruction execute. Most instructions are executed in one cycle. In addition, all Load, Store and Exchange instructions may perform multiple functions in the one cycle to significantly increase system speed. For example, automatic RAM address pointer incrementing/decrementing, testing for loop completion, and switching to a second RAM address pointer can all be accommodated in one basic four-microsecond cycle. This provides for very efficient and fast processing of table-oriented data as is found in an intelligent terminal application. Thus, data bytes may be moved from one table, or buffer, to another at a 12-microsecond/byte rate, including all overhead addressing functions.

The CPU contains many instructions especially useful for intelligent terminal tasks. The ability to set or reset any bit in any byte in RAM memory is provided along with associated conditional branch/skip instructions for any bit condition in RAM memory. The CRT Terminal uses two bits of the 8-bit character code to encode cursor, tab, and protect functions for each individual character or character position, for example. Byte comparisons between the PPS-8 accumulator and addressed RAM locations permit rapid table searching of data with automatic branching or skipping on comparisons or non-comparisons. CPU registers (6) may be loaded directly from ROM memory for subroutine parameters, stored messages, header formats, or form outlines.

The CPU also provides three levels of priority interrupt. The highest priority level is used for power fail detection. The second highest level is useful for a real time clock or a relative time clock. The third level of interrupt is used as the general system interrupt and may be daisy-chained through 15 I/O chips to provide a self-contained priority interrupt structure. The entire interrupt chain may be enabled or disabled, and individual I/O chips in the chain may be individually armed or disarmed to provide a very flexible interrupt structure. This structure is built in the chips, requiring only one external "OR" tie resistor on the interrupt request line. Also, the CPU provides an instruction, Read Interrupt Status, which immediately identifies the I/O device requesting the interrupt and the reason for the interrupt. No software polling of I/O devices is required.

The CPU directly addresses 16K ROM and 16K RAM. An additional chip select line on all ROM's and RAM's provides direct extension to 32K ROM and 32K RAM without any external components. In addition, all ROM's and RAM's directly interface to the CPU address and instruction/data bus without the need for other interface chips. The PPS-8 bus can directly drive up to 350 pF at rated speed. This permits loading the bus with 35-40 chips before being concerned with bus drivers. This is several times the bus driving capability of other systems.

DIRECT MEMORY ACCESS CONTROLLER (DMAC)

The DMA Controller provides eight independent channels of DMA capability. Built-in logic provides a DMA priority structure with Channel 0 having the highest priority and Channel 7 having the lowest priority. Each DMA channel is loaded with a starting address and a block length by the CPU. After that, the CPU is free to execute its main line program as the eight individual block transfers set up in the DMAC are executed. At the end of each individual block transfer, the DMAC notifies its appropriate I/O controller. The associated I/O controller may be programmed to respond to the DMA End of Block condition with an interrupt to the CPU or it may be programmed to ignore the EOB condition. On all intermediate transfers within the associated block of data, the I/O devices (independent of the CPU) monitor their control lines and request DMA service from the DMAC when a request for data transfer is received externally. Thus the CPU sets up the DMA channels (up to eight), goes away to work on other tasks, and is informed by the individual I/O controllers by means of interrupts at the end of their block transfer. Data rates on

DMA transfers may vary from 250,000 bytes per second down to one byte per second or less. Again, the CPU is only involved in setting up the initial channel addresses and block lengths and then in responding after a block of data has been completely transferred. Status indicators in the I/O controllers flag any error condition that may have occurred in the block transfer.

The DMA Controller has a Block Repeat function built in for these cases where a continuous repetition of a data block may be required as in a CRT, for example. DMA Channels 0-6 have a repeat control bit which may be set by the CPU. In this mode, the CPU sets the Repeat Bit flag in the channel to be used in the repeat mode and loads the channel starting address and block length in the primary channel (Channel 0-6) as well as in Channel 7, the Refresh channel. Now, when the primary channel reaches the end of the block of data (Block Length = 0) it checks its repeat flag bit and, since it is set, the primary channel transfers the initial address pointer and initial block length stored in Channel 7 into its registers so that it may repeat the block transfer again. At every End of Block condition, the primary channel will again refresh its initial address and block length from Channel 7, thereby continually repeating the block transfer as desired.

PARALLEL DATA CONTROLLER (PDC)

The PDC is a dual 8-bit programmable I/O controller which as the ability to initiate interrupt requests or DMA transfer requests upon the occurrence of external control line transitions. Each 8-bit port contains two programmable control lines whose functions are programmed by the CPU by means of control bytes. In addition, the mode of each port is programmable; i.e., input, output, or input/output. In addition, the type of data transfer is programmable; i.e., static, clocked, or handshake transfer. The ability of the PDC to request interrupts or DMA service may be armed or disarmed by appropriate control byte bit patterns. In addition, the PDC may be programmed to request an interrupt on the DMAC indication of an End of Block transfer or it may be programmed to ignore the EOB condition. Status registers detect the occurrence of any data transfer error (buffer underrun or buffer overrun) which may occur.

SERIAL DATA CONTROLLER (SDC)

The Serial Data Controller is a full duplex, USART chip. The function of the chip is programmable by means of a control byte loaded by the CPU. Controllable parameters include bits per character (5, 6, 7, or 8), number of stop bits for asynchronous operations (1.0, 1.5, 2.0), and for parity insertion/checking (odd, even, none). The SDC contains five RS-232-C interface control lines for convenience in interfacing to high speed modems. The SDC may be used in party line operation between master and slave terminals, for example. The SDC will transmit up to 250,000 bits per second in the synchronous mode. In this mode, the null character, once loaded into the SDC, will be automatically repeated in transmission until another valid data character is sent. Thus, the system will stay in sync even though a transmit buffer may run empty, provided the last character in the buffer is the null character. Double buffering of both the receive and transmit channels is provided.

The SDC also has a Receive Compare register which continuously compares a byte loaded into it by the CPU with incoming data. Upon comparison, the SDC can be programmed to request an interrupt, thereby notifying the CPU of a valid comparison. This function may be used to search for the terminal's address on the party line or multi-drop communication line, for example. Or it may be used to automatically search and verify a sequence of communication protocol control characters. Using this feature, the SDC can be set up to strip off all null (or sync) characters, identify the terminal address, and then start interrupting the CPU, or else requesting DMA service, on each framed input character. This is done independent of the CPU activity after the compare byte has been loaded. Multiple byte addresses or control sequences can be handled by successive recognition and then loading of the next byte of the sequence. Error checking for buffer overrun/framing errors, parity errors or carrier drop-outs is provided. Thus, after a block transfer, the quality of the entire transfer may be quickly checked by the CPU through reading the SDC quality register.

FLOPPY DISC CONTROLLER (FDC)

The Floppy Disc Controller provides all data formatting required for reading or writing on floppy disc media from byte-oriented RAM storage. Up to four floppy discs may be serviced by one FDC. For Write operations, the FDC provides track address verification and sector address search/comparison logic, preamble and postamble generation, write head current enabling, parallel to serial conversion, and CRC polynominal generation and detecting. For Read operations, the FDC provides track verification and sector address search/comparison logic, CRC polynomial generation from sensed data and comparison with the recorded CRC field, disc format verification, and serial to parallel data conversion.

In addition, the FDC provides a Read Compare register similar to the SDC. This register permits stripping of preamble bits, postamble bits, and address fields from the incoming data stream so that only the addressed sector data is transferred to RAM memory. No CPU processing is required to separate the floppy disc overhead fields (preamble, postamble, address) from the data field of interest.

To implement a full floppy disc memory system, a General Purpose Input/Output chip (GP I/O) is required for disc selection, track position control of the read/write head, head loading/unloading, and status information. External circuitry is needed to combine clock and data for recording and to extract the clock from this data when reading.

The format of the FDC is under software control so that user formats may be used in addition to the compatible IBM format. Non-IBM formats permit much greater data packing density than the IBM format provides. Many users may want to use a packed format for data storage, and then convert to IBM format for interchangeable discs. This can be readily done with the FDC. In fact, each track within the disc may have its own format. In a four disc system, for example, one disc may be recorded in IBM format for interchangeability, while the three other discs are recorded in a packed format for improved data capacity. A software routine would then be used to unpack the packed format and record the data in the less dense IBM compatible format for disc interchange compatibility. Disc commands include Read, Write, Write Format, Read CRC Check, Read Address Field, and Read Status.

GENERAL PURPOSE INPUT/OUTPUT CONTROLLER (GP I/O)

The GP I/O is a programmable chip which provides twelve discrete input lines and twelve discrete output lines. The input and output lines are addressed in groups of four lines each. Thus, input lines are addressed in terms of Group A, Group B, or Group C. Similarly, the output lines are grouped in terms of Group A, Group B, or Group C. Input lines are static; output lines are latched and maintain their levels until re-loaded.

The CPU addresses the GP I/O and commands it to read from individual input groups or to output to the individual output groups. Also provisions are made to "OR" input groups in one instruction, or to "AND" output groups. Thus, the "OR" condition of all input lines, by group, may be read into the CPU in one instruction. Or, the same bit pattern may be applied to all output groups in one instruction.

OTHER CONTROLLERS

Other controllers consist of keyboard controllers, display controllers, and printer controllers. The keyboard controllers provide all strobes, strobe return sensing, key debounce, key rollover, and key buffering functions. The CPU is only required to unload the key buffer once each 50-100 milliseconds. Display controllers provide all multiplexing of digit display information as well as digit select strobes. The CPU merely transmits up to a 16-character display buffer to the chip; the chip does the rest. A combination keyboard/display controller, the GP K/D, is available. This chip is used with up to 64-key keyboards and Panaplex,® Burroughs' Self Scan,® or LED displays. Several Printer Controller chips are also available. For example, a two-chip set is available for control of a 150 cps, alpha-numeric dot matrix printer. The only discrete circuitry required are the power driver transistors. Chip outputs drive one standard TTL load (2.6 ma). A combination Keyboard/Printer chip is also provided to control a 64-key keyboard as well as a 22-column printer.

TELECOMMUNICATIONS DATA INTERFACE CONTROLLER (TDI)

The TDI chip provides a full duplex programmable UART function as well as a 1200 BPS modem function. The modem may be strapped for Bell 202 or CCITT signaling frequency compatibility. The modem design accommodates 1200 BPS transmission over a dial up, unconditioned telephone line. The UART may be programmed for bits per character (8, 16, 64), parity (none, odd, even), and signaling frequency. The TDI has interrupt capability; up to 16 TDI chips may be incorporated in a single system. A serial mode which utilizes the modem function only and disables the UART function is also provided so that the chip can be used as a stand-alone modem.

Some external circuitry is required for full modem implementation. A four pole analog filter is required on the receiver input and a simple operational amplifier with four summing resistors is required on the transmitter output line.

RAM'S

The PPS-8 RAM's are 256 x 8 bit RAM's with full address decoding on each chip. The RAM's are dynamic, but appear static to the user since all refresh is done automatically without interference to the user. Standard 4K x 1 RAM's of the 16 pin configuration can be interfaced to the PPS-8 by means of our 4K Interface Controller Chip. This chip provides all interfacing and refresh functions required for standard memory operation. Two interface chips are required per system. These service up to a 16K x 8 RAM and present only two units of load (10pf) to the PPS Bus. Multiple memory modules may be used. The 4K RAM Controller Chip has the ability to float its output lines so that external control of the 4K RAM memory is possible. Thus, the 4K RAM can be loaded externally at the full 4K RAM speed; then the PPS-8 can operate on this data at rates up to 250,000 bytes per second.

ROM'S

The PPS-8 ROM's are 2048 x 8 bits, also with full address decoding on each chip. Thus, up to 16K of ROM easily fits on one 5½ x 7-inch PC board.

THE CRT TERMINAL

A basic CRT Terminal will first be discussed. Then an expanded terminal with floppy disc memory and printer will be described. Finally, an expansion of the terminal to graphic display functions will be briefly outlined.

The basic organization of the CRT Terminal includes PPS-8 RAM memory as the data storage media. A DMA channel is then used to load one of two line buffers, each being 80 characters in length. As the DMA loads one line buffer, the other line buffer is driving the CRT monitor. As one line buffer completes its line display function, control logic switches to the other line buffer which has been loaded through DMA. The initial line buffer is then refilled by DMA while the second line buffer is driving the display.

The CRT basic terminal implementation with the PPS-8 is illustrated in *Figure 1*. The system is implemented with a CPU, 2560 x 8 RAM Memory, 2K x 8 ROM/PROM, a DMAC, and a PDC. The CPU, PDC, DMAC, and two RAM chips (512 x 8) are contained on our PPS-8 Processor II evaluation board. The additional 2048 x 8 RAM is contained on our RAM evaluation board. One board of TTL logic contains the double line buffers and associated raster control, character generator and associated control logic. Our PROM evaluation board is presently used for program storage. Eventually, this board will be replaced with one ROM chip mounted on an option board which also contains provision for the Floppy Disc Controller, GP I/O, and SDC or TDI modem. The hardware used in the basic CRT terminal is illustrated in *Figure 10.* The display control word is illustrated in *Figure 2.* Two basic versions are shown. One provides upper/lower case characters plus cursor. The other provides the expanded functional controls of Tab and Display Protect.

DISPLAY CONTROL

The display is generated from a standard TV raster as shown in *Figure 3.* Eleven scans comprise the formation of one line of characters as illustrated. The display refreshing is accomplished as shown in *Figure 4A* and *4B.* In *Figure 4A,* Line Buffer 1 is driving the display through the character generator and shift register. It is also recirculating on itself corresponding to the eleven scans required to generate the full line of characters. During this time, Line Buffer 2 is being loaded via DMA through PDC 1. Timing for this operation is illustrated in *Figure 5.* After completion of the present line of display, Line Buffer 2, which has just been loaded with the next line of characters, is switched to the active display mode, and Line Buffer 1 is switched to the load mode. This is illustrated in *Figure 4B.*

SYSTEM TIMING

The timing associated with system operation is shown in *Figure 6a.* All numbers relate to PPS-8 system time required. Refreshing the line buffers required 320 microseconds per line time of 698.5 microseconds. Thus, 46% of system time is required for refreshing. At 9600 BPS, the modem servicing time requires 0.48% of system time for data transfer. This does not include modem control overhead. The time left for editing and system control software is approximately 100,000 system cycle times per second, or 40% of the system time. The expanded system with Floppy disc memory, printer, modem, and/or party line control is illustrated in *Figure 7.* Timing for this expanded system is shown in *Figure 6b.*

GRAPHICS DISPLAY

Addition of a graphics display function is also feasible. For a 125-line by 555-grid, graphic refresh via DMA is feasible. This mode would provide the graphic refresh plus an 800-character refresh capability. To implement the described graphic capability would require an additional 8K of RAM Memory. In addition, a large amount of CPU time would be required for graphic conversion. The PPS-8 implementation for this mode might be as illustrated in *Figure 8.* The additional graphics memory is constructed of standard 4K x 1 RAM chips of the 16 pin configuration. This memory module is interfaced to the PPS-8 CPU by means of a 4K RAM Interface chip now under development. The system in *Figure 8* would be adequate if the graphic data were previously formatted for graphic display prior to loading into the RAM module. Both graphic and character refresh would require approximately 75% of the CPU time. Some additional TTL logic would also be required to "OR" the graphic and character data into the video input of the monitor. If it is desired to do the graphic formatting in the terminal system, a second PPS-8 CPU and its ROM may be added to the system as illustrated in *Figure 9.* In this configuration, CPU-2 and ROM-2 are dedicated to graphics formatting using a 4K RAM Interface chip to access the 4K RAM memory while CPU-1 is providing the character refresh function out of its PPS RAM. When CPU-2 has formatted a graphic display, it then passes control of the 4K RAM module to CPU-1 for display and refresh of the graphic function as well as for a reduced (800) character display and refresh. This mode is feasible since each 4K RAM Interface chip has the ability to float its output lines to the 4K RAM module.

Alternately, a DMAC can be added to CPU-2 so that all graphic formatting display, and refreshing is done by the second system. This will off load System 1 so that all other intelligent terminal functions can be executed in System 1. The addition of System 2 would also permit graphic expansion to 225 x 555 points with the addition of another 8K of RAM memory. System 2 would consist of a single 5 x 7 in. PC board plus the additional 8K of RAM.

SUMMARY

This application illustrates the power and flexibility of the PPS-8 microcomputer system in both the single and multiprocessor modes. The advantages of a complete set of systems-structured, intelligent controllers are demonstrated with this applications study. Very significant product cost reductions may be achieved with this approach, as demonstrated by the minimal amount of hardware required for this CRT application.

The hardware for the basic display terminal is illustrated in *Figure 10.* The addition of the floppy disc and printer require 3 additional chips from those shown. The addition of the graphics capability requires one additional interface chip plus the expanded 8K of RAM plus additional ROM memory for programming . . . probably one of our 2K x 8 ROM chips.

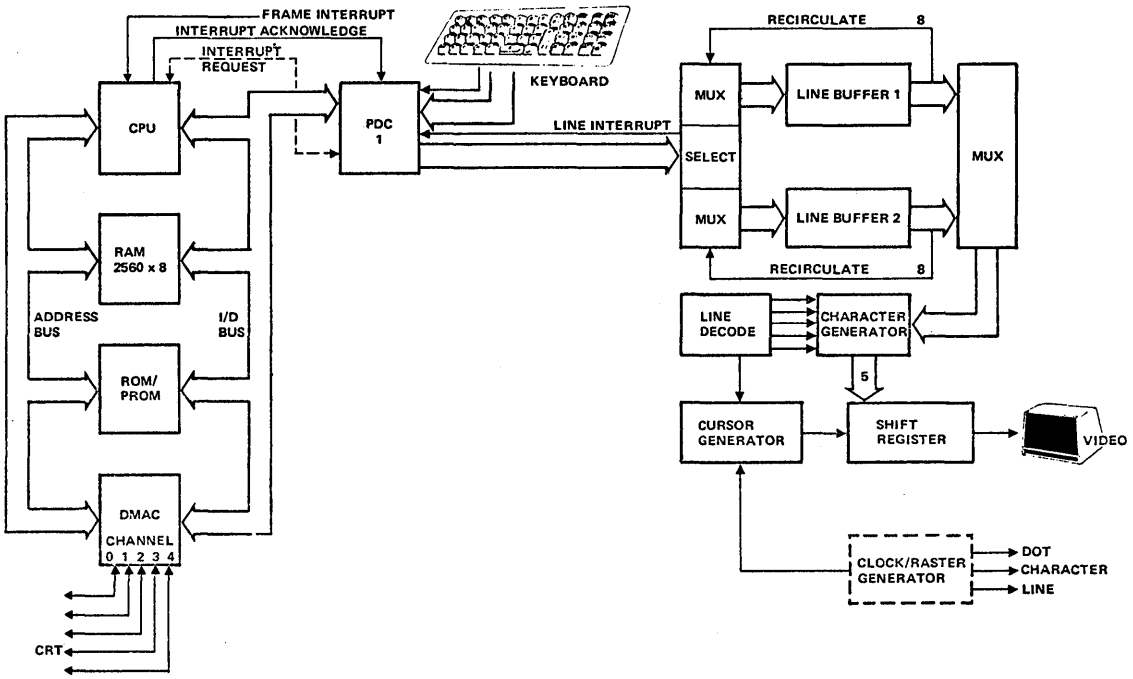


Figure 1. **BASIC CRT SYSTEM**

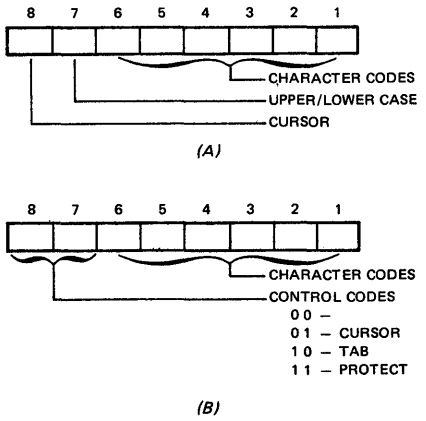


Figure 2. **DISPLAY CONTROL**

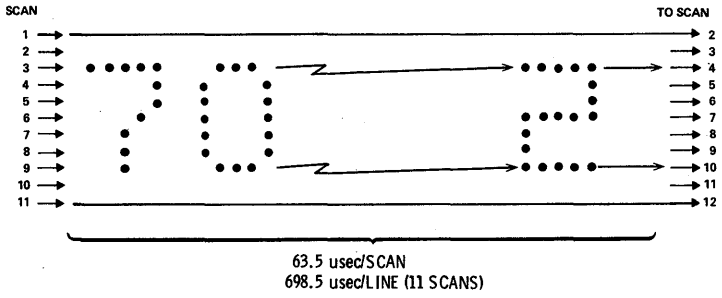


Figure 3. **SCAN TIMING**

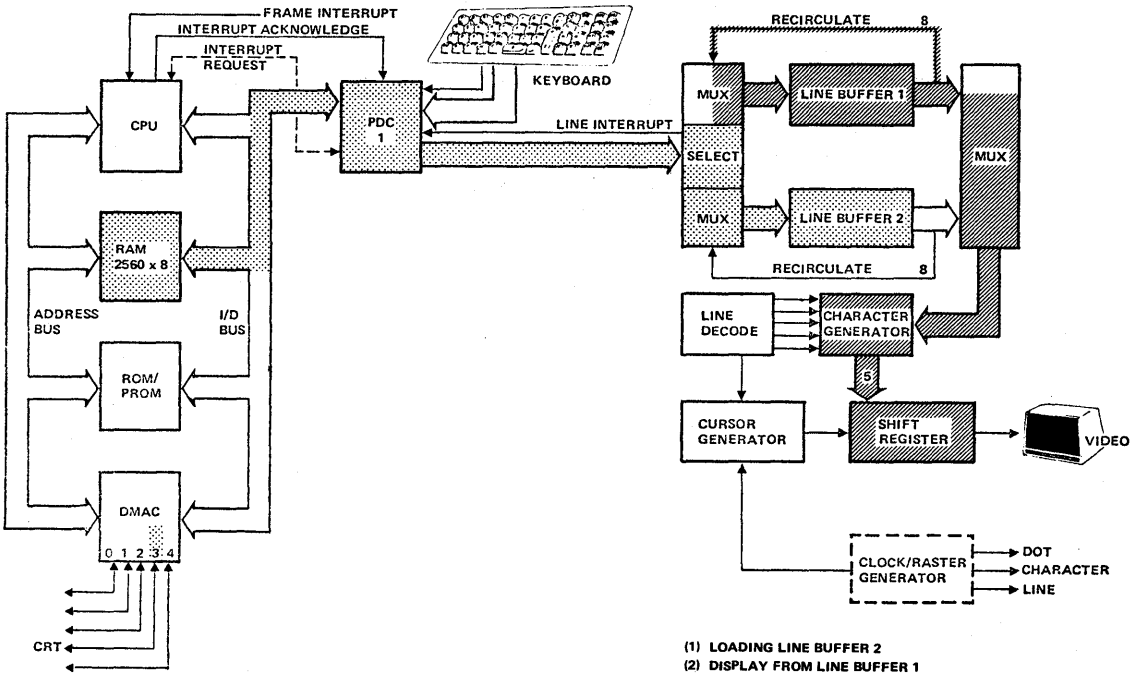


Figure 4a. **PPS-8 CRT SYSTEM**

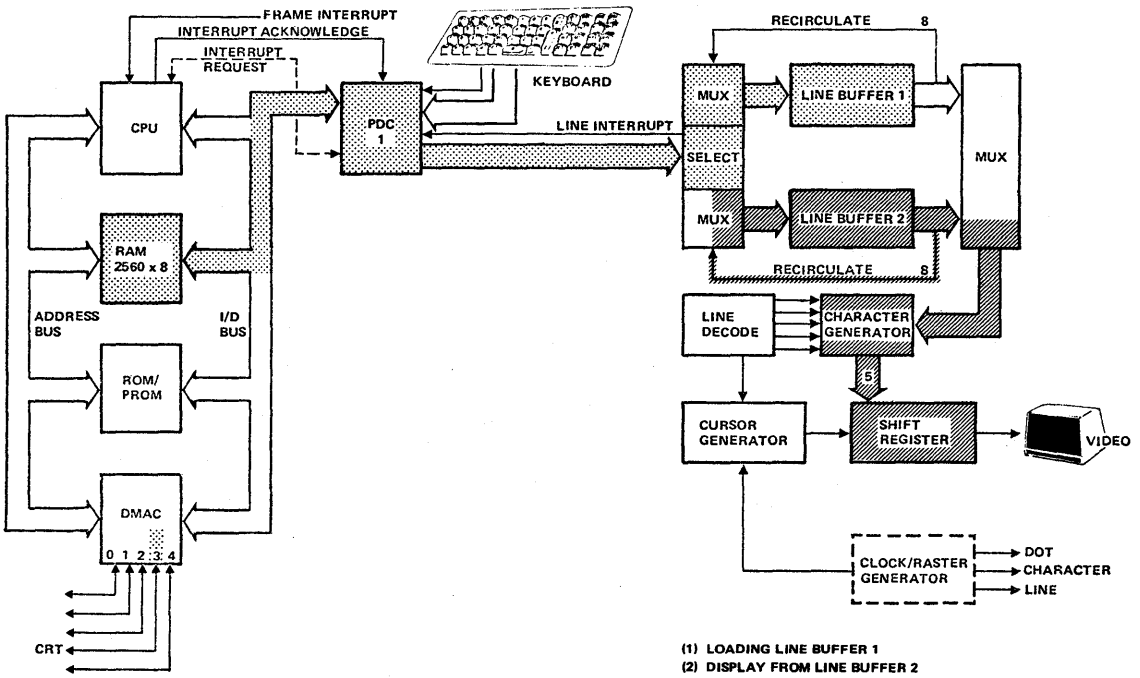


Figure 4b. **PPS-8 CRT SYSTEM**

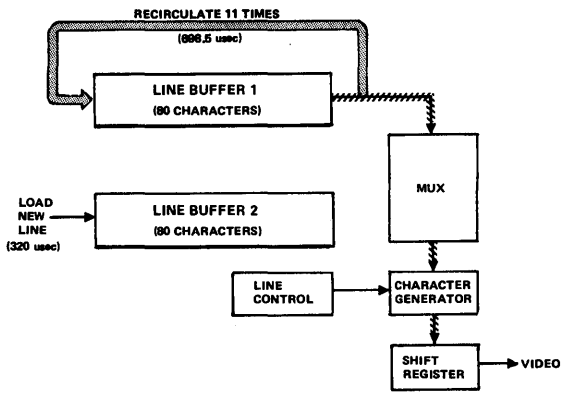


Figure 5. **BUFFER TIMING**

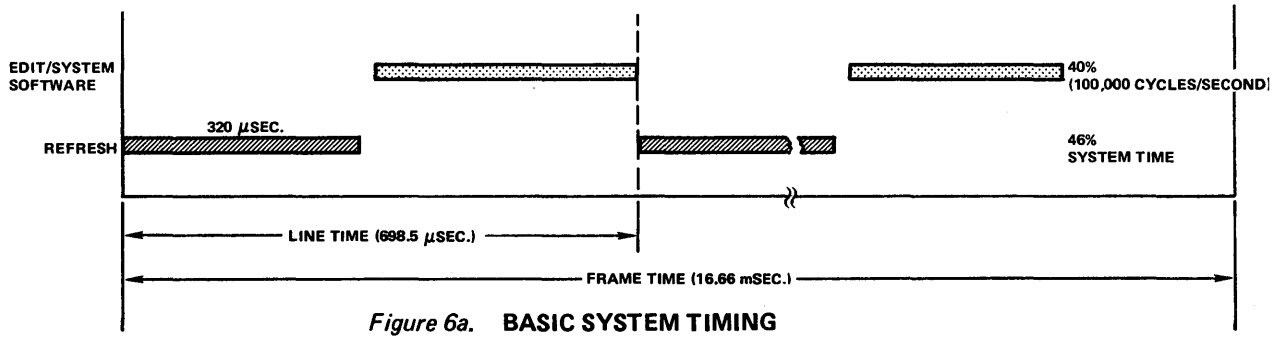


Figure 6a. **BASIC SYSTEM TIMING**

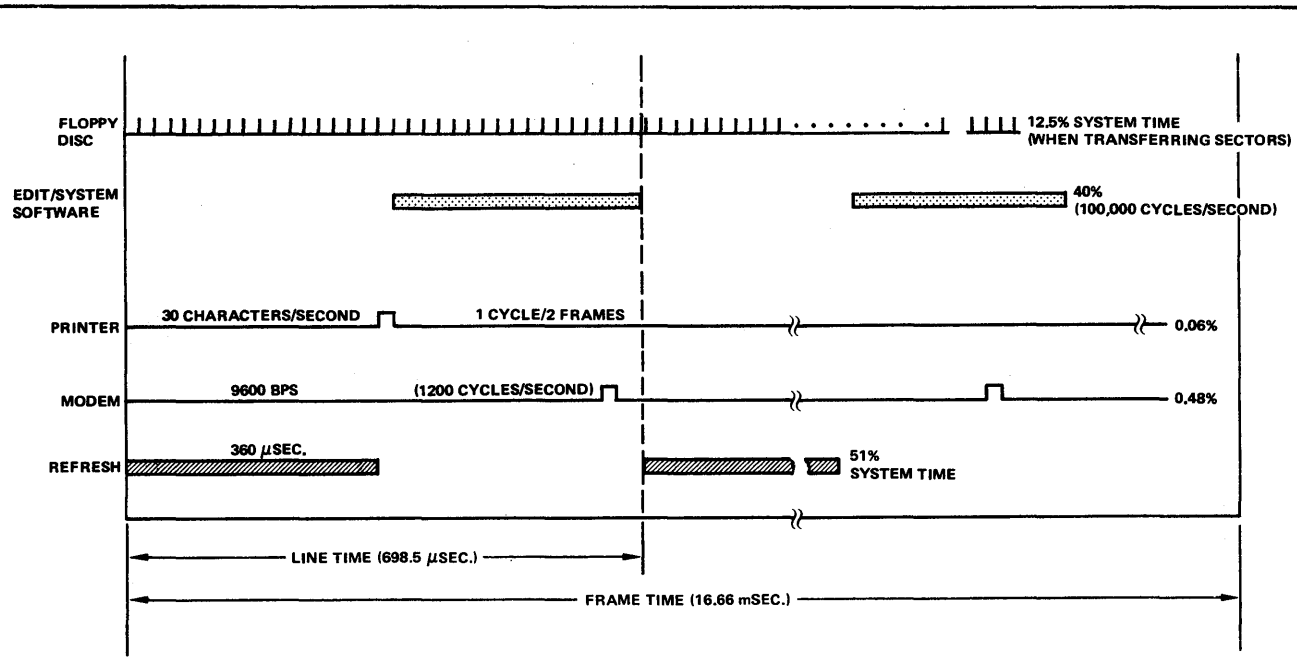


Figure 6b. **EXPANDED SYSTEM TIMING**

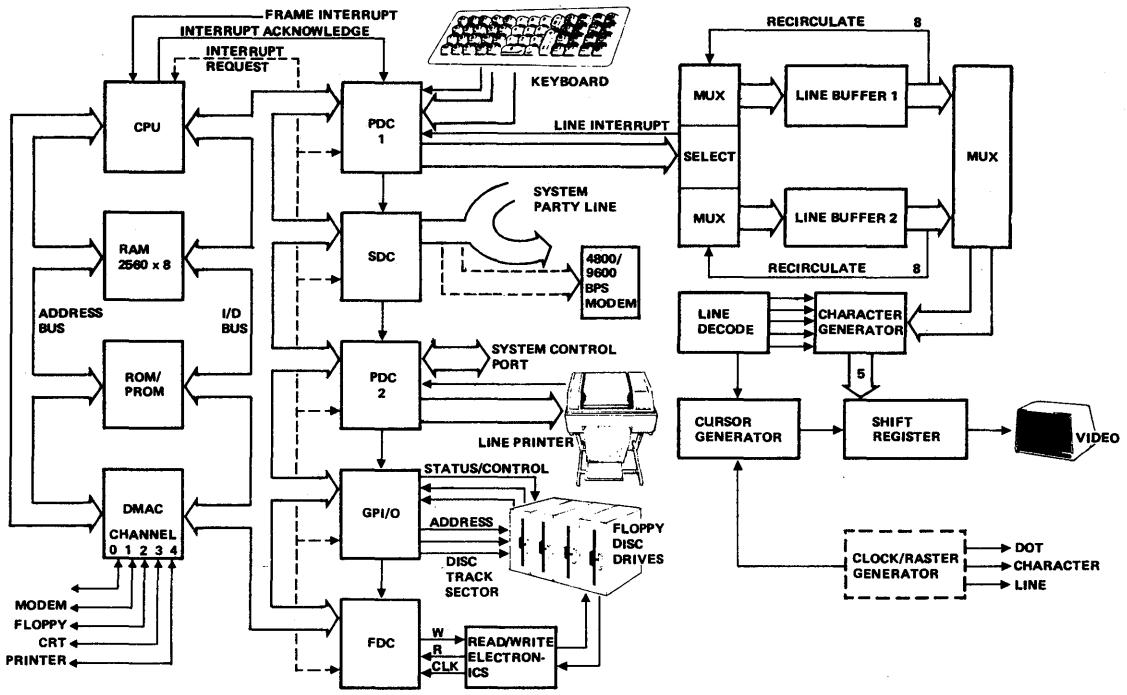


Figure 7. **PPS-8 CRT EXPANDED SYSTEM**

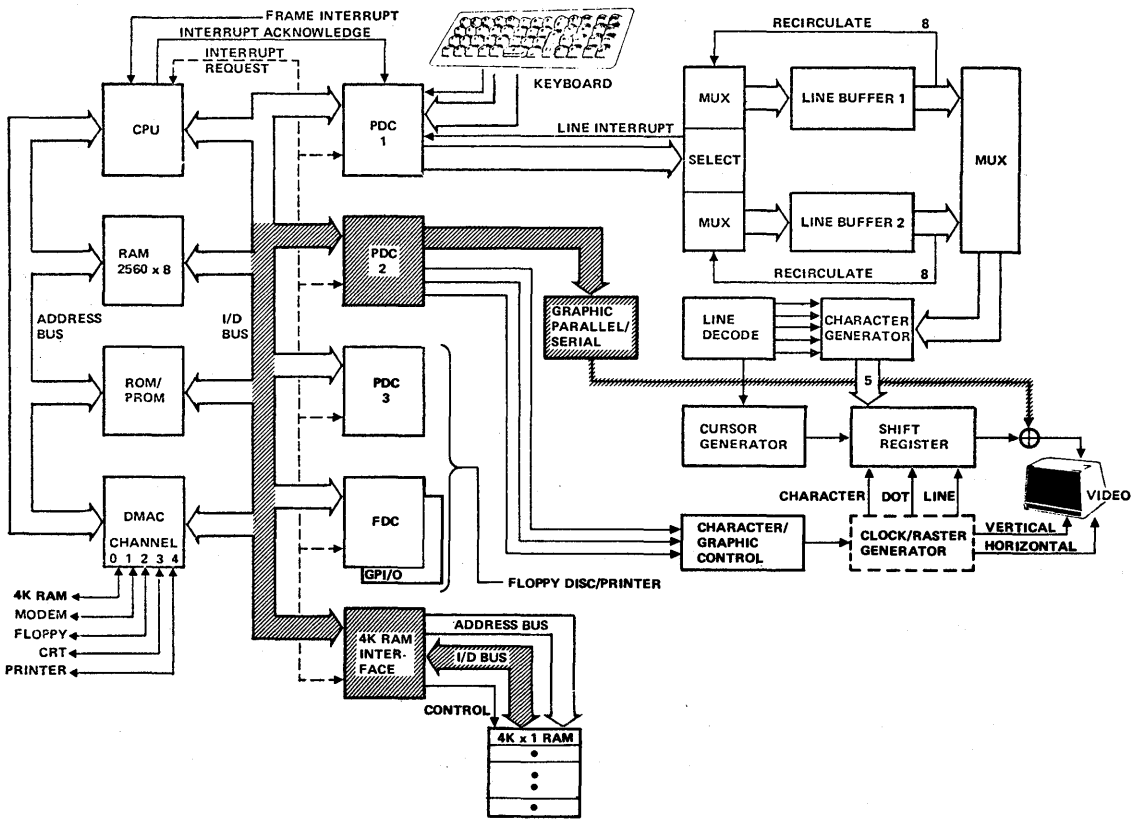


Figure 8. **GRAPHIC DISPLAY CONFIGURATION**

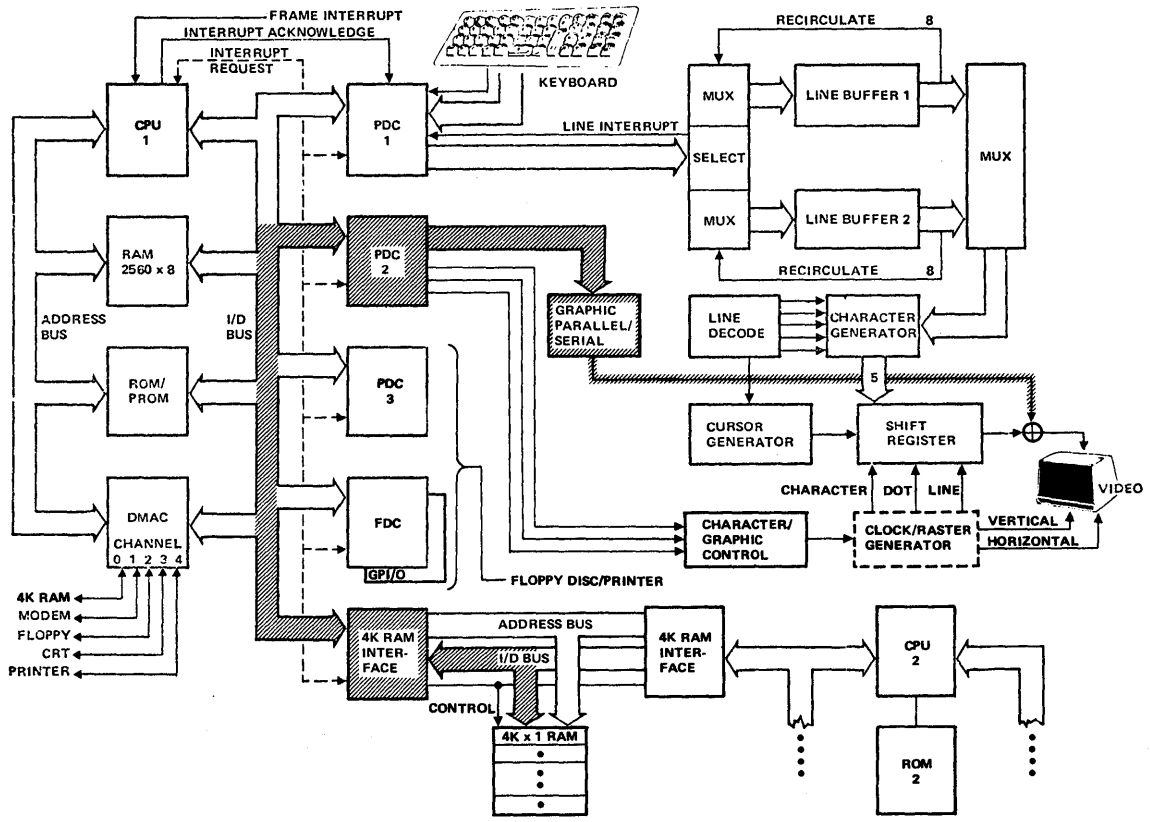


Figure 9. **MULTI-PROCESSOR GRAPHIC DISPLAY**

Figure 10. **CRT TERMINAL HARDWARE**

Part Two

Electronic Engineering TIMES

Linear/Digital Proceedings

Integrated Circuit Applications Conference

See Last Page for Table of Contents

1. THE MONOLITHIC VOLTAGE FREQUENCY CONVERTER

JAMES C. SCHMOOCK
Design Engineer
Raytheon Company
Semiconductor Division
350 Ellis Street
Mountain View, California 94042
415 968-9211

The development of the monolithic I.C. Voltage-to-Frequency Converter offers design engineers a practical low-cost alternate to modular units costing much more.

Voltage-to-frequency converters (VFC) are used for analog-to-digital conversion, modulation, data transmission, and system isolation. The sister device, the frequency-to-voltage converter (FVC), is used for tachometry, frequency translation, and servo control. New uses for these versatile products are being discovered daily.

The RM/RC 4151 monolithic converter can be used in a number of VFC and FVC connections to fill almost all needs, from low cost single supply circuits to precision configurations which preserve transfer function linearity over five decades.

VFC PRINCIPLES

Most VFC's have an output which is a series of pulses. These pulses have a fixed period, and can be either positive or negative. The VFC changes the duty cycle of the output pulse train in response to the input voltage.

The block diagram for the simplest type of VFC which can be constructed using the RM/RC 4151 is shown in Figure 1. The 4151 contains an input voltage comparator, a one-shot, a switched current source, and an open-collector NPN logic output transistor. For this simple VFC configuration the current source output is integrated in the passive network R_B - C_B. One input of the comparator, pin 6, senses the resulting voltage. The input voltage to the VFC, V_I, is applied at the other comparator input, pin 7. If the input voltage is greater, the comparator fires the one-shot. The one-shot output drives the logic output low and also switches on the current source. During the time period, T, when the one-shot is on, the current source will inject a lump of charge $Q = I_O T$ into the R_B - C_B integration network. This increases the voltage V_B by a few millivolts. If V_B has not increased to a voltage greater than V_I, the comparator again fires the one-shot and the current source injects another lump of charge into the integration network. As soon as the voltage V_B is greater than V_I, the voltage comparator keeps the one-shot off and V_B

is left to decay through R_B. During this resting period, the logic output remains high. As soon as V_B decays to the point where $V_B = V_I$, the voltage comparator triggers the one shot and the cycle repeats, giving the AC component of V_B its characteristic sawtooth waveform. In steady state operation the one shot fires just often enough to keep $V_B \geq V_I$. Since the capacitor C_B discharges at a rate proportional to V_B/R_B, the system operates at a frequency proportional to the input voltage. Although this simple VFC configuration is quite useful in many of low-cost applications, it suffers from inaccuracy due to a number of sources. Linearity errors arise primarily from the current source output conductance and variations of the sawtooth voltage V_B. This sawtooth voltage also creates a frequency offset as well as does the input offset voltage of the comparator. Fortunately, all of these inaccuracies are overcome by the precision mode VFC configuration which uses an op-amp integrator to replace the network R_B - C_B.

Figure 2 shows the block diagram for the precision VFC. This circuit actually functions as a current-to-frequency converter, with the input current I_I being drawn from the summing node of the op amp integrator. In steady state operation the lumps of charge coming from the 4151 current source output exactly balance the input current.

Since the op amp inverting input is held at zero volts, this circuit is easily transformed into a voltage-to-frequency converter by the input resistor R_B. The input voltage must be negative, because the resetting current I_O is positive. A negative input voltage V_I causes the integrator output to ramp upwards at a rate of V_I/R_BC_I volts per second. As soon as the integrator output reaches the voltage V, the 4151 input voltage comparator fires the one shot, which turns on the current source. The current source injects a lump of charge $Q = I_OT$ into the integrator summing node. During this time period, T, the output of the integrator will ramp downward at the rate of $(V_I/R_B - I_O)C^{-1}$ volts per second. Any nonlinearity due to the 4151 current source output conductance is eliminated in this circuit because the integrator holds the current source output at zero volts. The only offset present in the voltage-to-frequency transfer function is due to the op amp input offsets. If an op amp with offset trim provision is used, this circuit will remain linear for inputs all the way down to zero.

RM/RC 4151 CIRCUIT DESCRIPTION

The 4151 VFC is easy to use and apply if you understand the operation of it through the block diagram, Figure 1. Many users,

though, have expressed the desire to understand the workings of the internal circuitry. Figure 3 shows the schematic of the 4151. The circuit can be divided into five sections: the internal biasing network, input comparator, one shot, voltage reference, and the output current source.

The internal biasing network is composed of Q_{39} - Q_{43}. The N channel FET Q_{43} supplies the initial current for zener diode Q_{39}. The NPN transistor Q_{38} senses the zener voltage to derive the current reference for the multiple collector current source Q_{41}. This special PNP transistor provides active pull-up for all of the other sections of the 4151.

The input comparator section is composed of Q_1 - Q_7. Lateral PNP transistors Q_1 - Q_4 form the special ground-sensing input which is necessary for VFC operation at low input voltages. NPN transistors Q_5 and Q_6 convert the differential signal to drive the second gain stage Q_7. If the voltage on input pin 7 is less than that on threshold pin 6, the comparator will be off and the collector of Q_7 will be in the high state. As soon as the voltage on pin 7 exceeds the voltage on pin 6, the collector of Q_7 will go low and trigger the one shot.

The one-shot is made from a voltage comparator and an R - S latch. Transistors Q_{12} - Q_{15} and Q_{18} - Q_{20} form the comparator, while Q_8 - Q_{11} and Q_{16} - Q_{17} make up the R - S latch. One latch output, open-collector reset transistor Q_{16}, is connected to a comparator input and to R_O terminal pin 5. Timing resistor R_O is tied externally from pin 5 to $+V_{CC}$ and timing capacitor C_O is tied from pin 5 to ground. The other comparator input is tied to a voltage divider R_3 - R_5 which sets the comparator threshold voltage at 0.667 V_{CC}. One-shot operation is initiated when the collector of Q_7 goes low and sets the latch. This causes Q_{16} to turn off, releasing the voltage at pin 5 to charge exponentially towards $+V_{CC}$ through R_O. As soon as this voltage reaches 0.667 V_{CC}, comparator output Q_{20} will go high causing Q_{10} to reset the latch. When the latch is reset, Q_{16} will discharge C_O to ground. The one shot has now completed its function of creating a pulse of period $T = 1.1\ R_OC_O$ at the latch output, Q_{21}. This pulse is buffered through Q_{23} to drive the open-collector logic output transistor Q_{32}. During the one-shot period the logic output will be in the low state. The one-shot output is also used to switch the reference voltage by Q_{22} and Q_{24}.

The low T.C. reference voltage is derived from the combination of a 5.5V zener diode with resistor and diode level shift networks.

A stable 1.89 volts is developed at pin 2, the emitter of Q_{33}. Connecting the external current-setting resistor R_S = 14.0kΩ from pin 2 to ground gives 135μA from the collectors of Q_{33} and Q_{34}. This current is reflected in the precision current mirror Q_{35} - Q_{37} and produces the output current I_O at pin 1. When the R - S latch is reset, Q_{22} and Q_{24} will hold the reference voltage off, pin 2 will be at 0 volts, and the output current will be off. During the one-shot period T, the latch will be set, the voltage at pin 2 will go to 1.89 volts, and the output current will be switched on.

USING THE RM/RC 4151 VOLTAGE-TO-FREQUENCY CONVERTER

One of the advantages to using the RM/RC 4151 is its extreme flexibility. It operates on a single supply from 8.0 to 22 volts, the open-collector output is compatible with all popular logic types, and nearly every aspect of its performance can be altered by appropriate choice of component values. Full scale output frequency can be programmed from 1.0Hz to 100kHz.

Figure 4 shows the complete applications circuit corresponding to the block diagram of Figure 1. The resistor R_S tied from pin 2 to ground sets the output current from pin 1. With the nominal values of R_S = 14.0kΩ, the output current I_O will be about 135μA. If it is necessary to trim scale factor, use the resistor-pot combination as shown. The load resistor R_L tied to pin 3 should be connected to the appropriate pull-up voltage for the logic type used, up to +22 volts. The timing network, R_OC_O, is connected to pin 5. Keep R_O and C_O within the ranges $6.8k\Omega \leq R_O \leq 680k\Omega$ and $.001\mu f \leq C_O \leq 1.0\mu f$. Calculate the one-shot period using the formula $T = 1.1\ R_OC_O$. Using the design equations given in Figure 4, it can be determined that the full scale frequency output is 10kHz for the maximum input voltage of +10 volts. Although this circuit configuration is the least accurate of all the ones presented here, it is quite useful because of its low cost, and because it operates from a single positive supply. It accepts positive input voltage from 0 to +10.

Figure 5 shows another single-supply configuration which uses the RC3403A ground sensing op amp to provide increased accuracy. Input voltage range for this circuit is 0 to +10 volts. The pair of 1kΩ resistors attenuate the input voltage by a factor of two. In addition, the nominal value of R_S has been increased to 28kΩ, so that the voltage at 4151 pin 1, the current source output, varies only over the range 0 to +5 volts. This is one reason for the increased linearity of this circuit.

The circuit features a unique offset trim adjustment which uses the RC3403A bias current to develop a voltage across R_B' which cancels the offset voltage. To use this technique it is necessary to have an op amp with stable input bias current such as the RC3403A.

For maximum accuracy, use the circuit of Figure 6, the precision mode VFC connection. This is the complete applications circuit corresponding to the block diagram of Figure 2. Input voltage range is from 0 to -10 volts and full scale frequency is 10kHz. The RC 4131 op amp specified has an offset trim adjustment, so with the offset nulled this circuit retains linearity all the way down to 0Hz. The 4151 operates from only the positive supply, but the op amp specified requires the positive and negative supply. A ground sensing op amp may be used instead, in which case the negative supply can be eliminated, although input voltage range is still negative. Also, no ground sensing op amp to date has built-in offset trim. The diode connected across the op amp stops a potential latch-up condition by preventing the voltage at 4151 pin 7 from going more than 300mV below ground. This circuit as shown achieves typical linearity error of only 0.02 percent. Refer to Table 1 for a performance comparison of the different VFC configurations.

COMPARISON OF VFC APPLICATIONS CIRCUITS

Table 1 compares the VFC applications circuits for typical linearity, frequency offset, response time constant for a step input from 0 to 10 volts, sign of input voltage, and whether the circuit will operate from a single positive supply or split supplies.

	FIGURE 4	FIGURE 5	FIGURE 6
Linearity	1%	0.2%	0.02%
Frequency Offset	+10Hz	0	0
Response Time Constant	135mS	10μS	10μS
Input Voltage	+	+	-
Single Supply	yes	yes	yes
Split Supply	---	---	yes

TABLE 1

The Monolithic Voltage Frequency Converter
James C. Schmoock

FREQUENCY-TO-VOLTAGE CONVERSION

Making an FVC with the RM/RC4151 is just as easy as making a VFC. All of the same equations for scale factor, period, and voltage range still apply. See Figure 7 for the simple single supply FVC circuit. The incoming frequency signal triggers the 4151 input voltage comparator which fires the one shot. The one shot switches on the current source for a period of time $T = 1.1\ R_O C_O$. The current pulses or lumps of charge come out of pin 1 and are integrated in the R_B - C_B network to produce the output voltage.

In order to correctly trigger the one shot, the 4151 input voltage comparator must be turned on for a time period which is less than the period of the one shot, T. The comparator is considered to be on when the voltage on 4151 pin 7 is greater than the voltage on pin 6.

Figure 7 also shows how to trigger the 4151 input with a square wave. With no signal present, resistive voltage dividers tied to 4151 pins 6 and 7 keep the input comparator in the off state. An incoming 5 volt p-p square wave is differentiated by the .022µf capacitor and the resistive voltage divider. This produces a pulse at pin 6 which turns on the comparator for a time less than the one-shot period of 75µS.

Another scheme for input signal conditioning is shown in Figure 8. Here, a D-type flip-flop acts as a differentiator. As soon as the input signal turns on the 4151, the pulse output from pin 3 resets the flip-flop, making a very short pulse. For any of the FVC configurations, an external voltage comparator can be used to square-up small sinusoidal waveforms.

The precision FVC circuit of Figure 9 gives the best accuracy and linearity. As with all 4151 applications, scale factor can be programmed by choice of component values. The op amp integrator improves the linearity of this circuit because it keeps 4151 pin 1 at zero volts. This prevents any change of I_O due to output conductance of the 4151 switched current source. Like all frequency-to-voltage converters, there is a tradeoff between response time and output ripple. If C_I = 0.1µf, the ripple will be 100mV. The response time constant is $\tau_R = R_B C_I$ = 10mS.

PROGRAMMING THE 4151

The 4151 can be programmed to operate with a full scale frequency anywhere from 1.0Hz to 100kHz. In the case of the VFC configuration, nearly any full scale input voltage from 1.0V

and up can be tolerated if proper scaling is employed. Here is how to determine component values for any desired full scale frequency.

1. Set R_S = 14kΩ or use a 12k resistor and 5k pot as shown in the figures. (The only exception to this is Figure 5.)

2. Set $T = R_O C_O = 0.75\ (\frac{1}{f_O})$ where fo is the desired full scale frequency. For optimum performance make $6.8k\Omega \leq R_O \leq 680k\Omega$, and $.001\mu f \leq C_O \leq 1.0\mu f$.

3. a) For the circuit of Figure 4 make $C_B = 10^{-2}\ (\frac{1}{f_O})$ Farads. Smaller values of C_B will give faster response time, but will also increase frequency offset and nonlinearity.

   b) For the active integrator circuits make $C_I = 5 \times 10^{-5}\ (\frac{1}{f_O})$ Farads. The op-amp integrator must have a slew rate of at least $135 \times 10^{-6}\ (\frac{1}{C_I})$ where the value of C_I is again given in Farads, and the op-amp slew rate is in volts per second.

4. a) For the circuits of Figures 4 and 5 keep the values of R_B and R_B' as shown and use an input attenuator to give the desired full scale input voltage.

   b) For the precision mode circuit of Figure 6, set $R_B = \frac{V_{IO}}{100\mu A}$ where V_{IO} is the full scale input voltage. Alternately the op amp inverting input (summing node) can be used as a current input with full scale input current I_{IO} = 100μA. For frequencies over 10kHz, bypass pin 6 to ground with .01μf.

5. For the F-V converters, pick the value of C_B or C_I to give the optimum tradeoff between response time and output ripple for the particular application.

DESIGN EXAMPLES

I Problems

Design a precision VFC with fo = 100kHz and V_{IO} = -10 volts. Use the applications circuit of Figure 6.

1. Set R_S = 14.0kΩ.

2. $T = 0.75\ (\frac{1}{10^5}) = 7.5\mu S$. Let R_O = 6.8kΩ and C_O = 0.001μf.

3. $C_I = 5 \times 10^{-5} \left(\frac{1}{10^5}\right) = 500pf$. Op amp slew rate must be at least $SR = 135 \times 10^{-6} \left(\frac{1}{500pf}\right) = 0.27\ V/\mu S$.
4. $R_B = \frac{10V}{100\mu A} = 100k\Omega$.
5. Bypass 4151 pin 6 to ground with $0.01\mu f$.

II Problem

Design a very low cost average temperature indicator with digital readout. See Figure 10 for the block diagram of the solution. Use the RM/RC 4151 precision VFC configuration of Figure 6 to convert the output voltage of a temperature transducer to a low frequency which can be counted by a calculator chip. One of the new solid state temperature transducers or a more conventional diode op amp circuit can be used to sense temperature. An absolute value circuit can be used ahead of the VFC for transducers which have outputs that swing positive and negative for temperatures above and below zero. The sign bit from the absolute-value circuit triggers the calculator chip to count up or down. Refer to the literature for more detailed instructions on how to use low cost calculator chips as event counters. Since calculator chips can only count very low frequencies, let's program the 4151 for a full scale frequency of 1.0Hz, using the circuit of Figure 6.

1. Set $R_S = 14.0k\Omega$.
2. $T = 0.75 \left(\frac{1}{1}\right) = 0.75$ seconds. Let $R_O = 680k\Omega$ and $C_O = 1.0\mu f$.
3. $C_I = 5 \times 10^{-5} \left(\frac{1}{1}\right)$ Farad $= 50\mu f$.
4. $R_B = 100k\Omega$.

III Problem

Design a tachometer with bar-graph LED readout. Maximum rate is 5000 RPM and supply voltage is 12.0 volts. Accuracy requirement is ±5 percent, and output response time constant should be less than 100mS.

Figure 11 shows the solution to this one. The single-supply FVC circuit of Figure 7 is used here to convert the pulse train into a voltage proportional to the frequency.

It is assumed that the input signal has already been conditioned using one of the techniques mentioned earlier. As the 4151 output voltage increases with increasing RPM, low cost quad voltage comparators light up green, yellow, and red LED's which make up the bar graph. The bar graph can be made as large as desired by using more comparators and LED's. The 4151 is to be programmed for a full scale frequency of 5000 RPM (83.3Hz).

1. Set $R_S = 14.0k\Omega$.

2. $T = 0.75\ (\frac{1}{83.3}) = 9mS$. Let $R_O = 82k\Omega$ and $C_O = 0.1\mu f$.

3. Full scale output voltage is 10.0 volts. $R_B = \frac{10V}{100\mu A} = 100k\Omega$.

4. Output response time constant $\tau_R \leq 100mS$. Therefore $C_B \leq \frac{\tau R}{R_B}$ or $C_B \leq 2\mu f$. Worst case ripple voltage $V_R = \frac{9mS X 135\mu A}{2.0\mu f} = 608$ mV. A ripple of 608 mV out of 10 volts would cause some blurring of the display between adjacent LED's, but it would still be possible to achieve 5 percent accuracy with a display of 20 LED's.

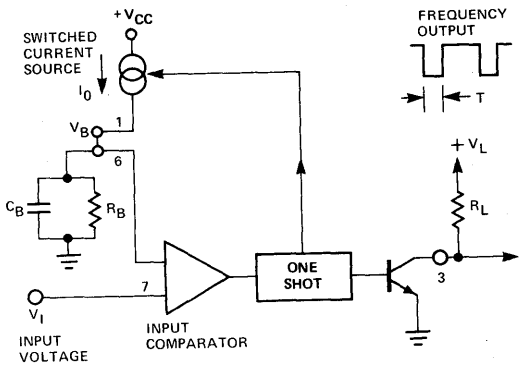
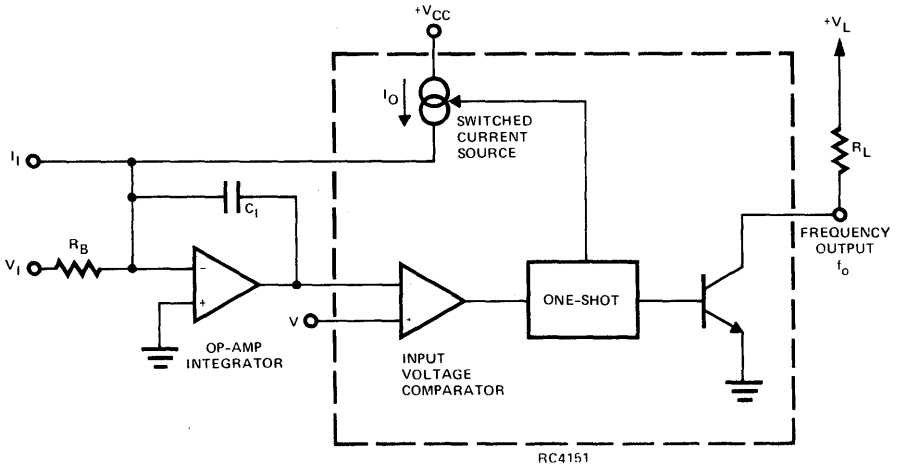


FIGURE 1 - RM 4151 VFC BLOCK DIAGRAM



PRECISION VFC BLOCK DIAGRAM

FIGURE 2 - RM 4151 PRECISION VFC BLOCK DIAGRAM

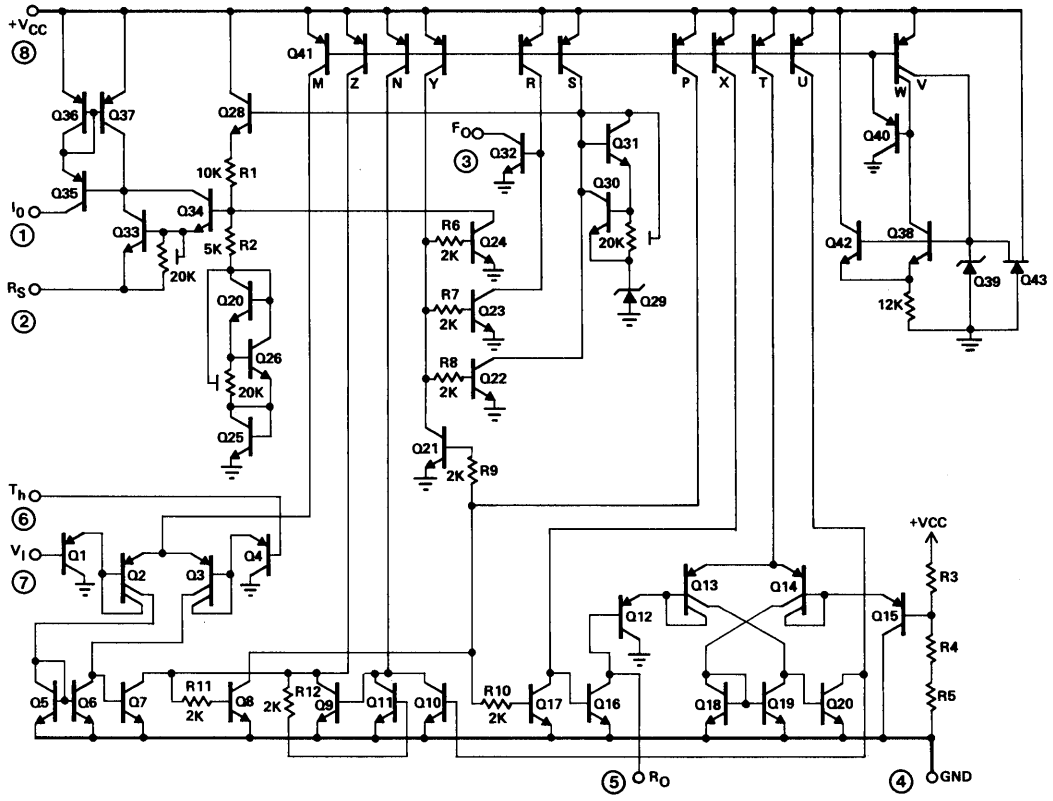


FIGURE 3 - RM 4151 SCHEMATIC DIAGRAM

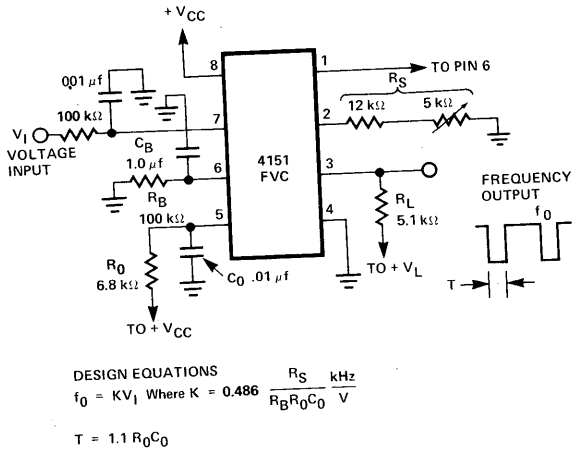


FIGURE 4 - SINGLE SUPPLY VFC

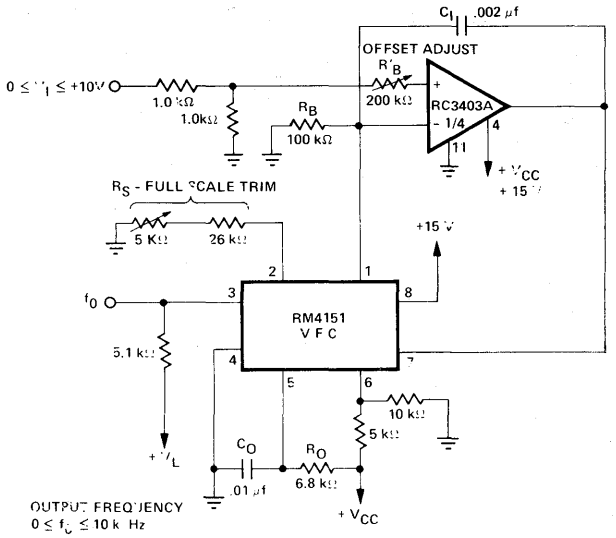


FIGURE 5 - PRECISION SINGLE SUPPLY VFC

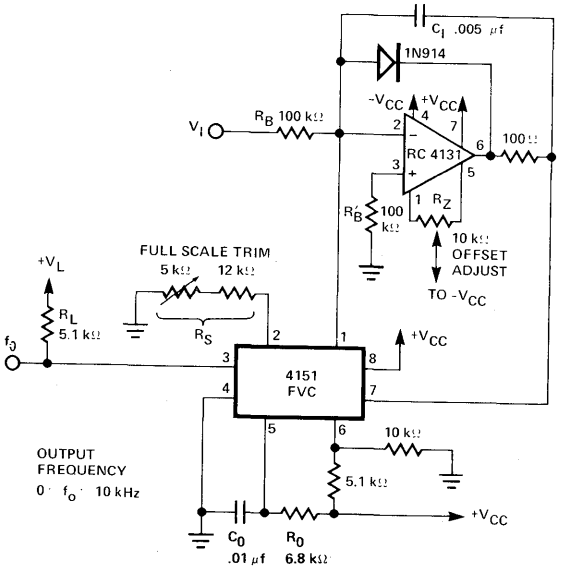


FIGURE 6 - PRECISION VFC

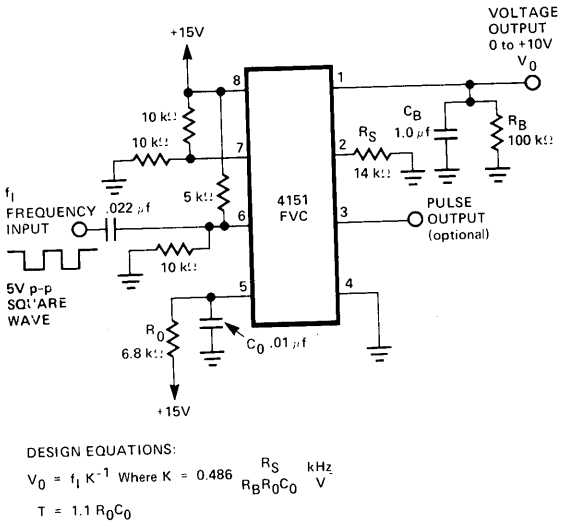


FIGURE 7 - SINGLE SUPPLY FVC

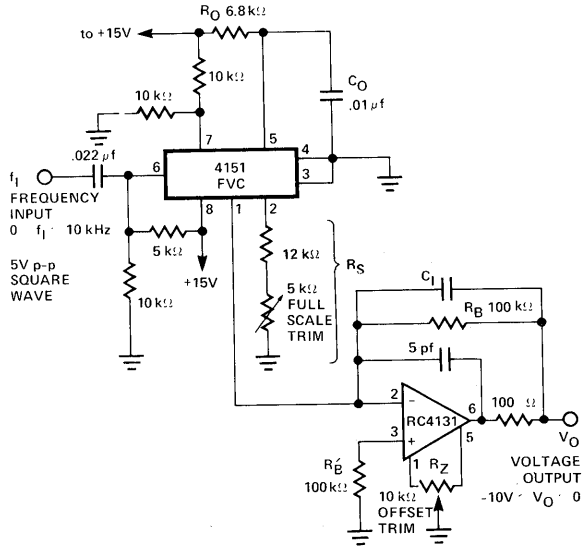
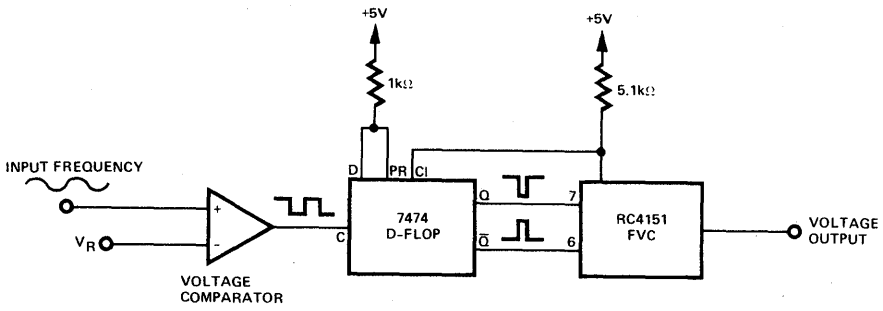
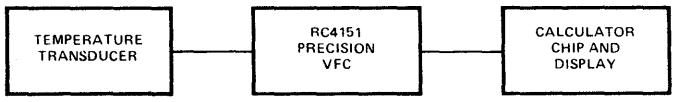


FIGURE 8 - PRECISION FVC



INPUT SIGNAL CONDITIONING FOR FVC

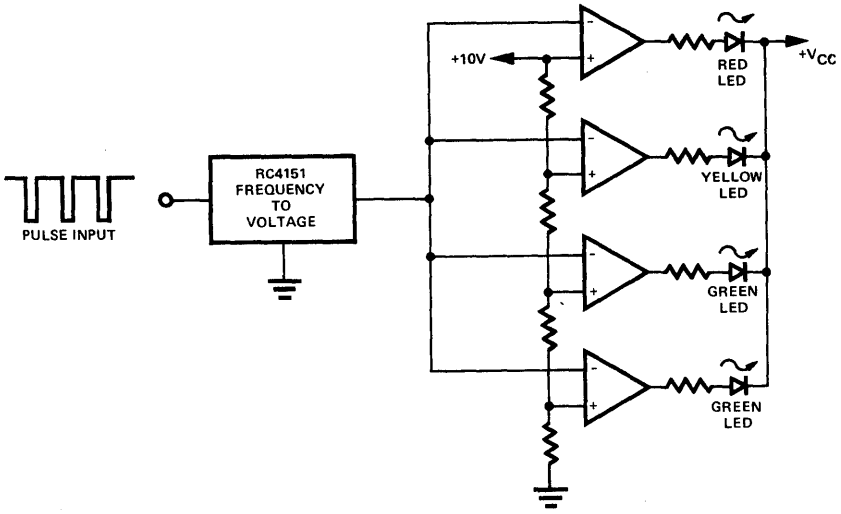
FIGURE 9 - FVC INPUT CONDITIONING



SCALE FACTOR 1 Hz/V

LONG TERM AVERAGE TEMPERATURE WITH DIGITAL READOUT

FIGURE 10 - LOW COST AVERAGE TEMPERATURE INDICATOR



TACHOMETER WITH LED DISPLAY

FIGURE 11 - TACHOMETER WITH LED DISPLAY

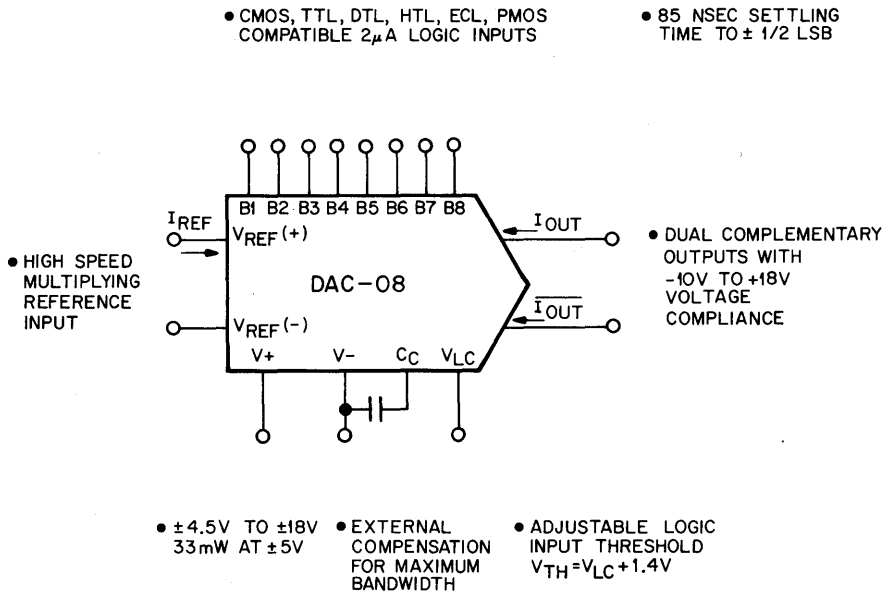
2.

DAC-08 APPLICATIONS COLLECTION

JOHN SCHOEFF & DONN SODERQUIST
Precision Monolithics
Santa Clara, CA

There has been a trend in recent years toward providing totally dedicated Digital-to-Analog Converters with limited applications versatility. This application note describes a new type of monolithic DAC designed for an extremely broad range of applications, the Precision Monolithics DAC-08.

Several unique design features of this low cost DAC combine to provide total applications flexibility. Principal among them are: dual complementary, true current outputs; universal logic inputs capable of interfacing with any logic family; 85 nsec settling time; high speed multiplying capability; and finally, the ability to use any standard system power supply voltages. A description of these features is given followed by specific applications using each feature.



THE FLEXIBLE D/A CONVERTER

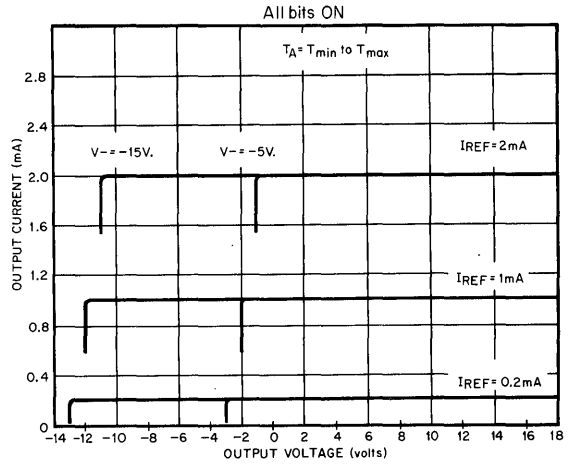
OUTPUT

HIGH VOLTAGE COMPLIANCE CURRENT OUTPUTS

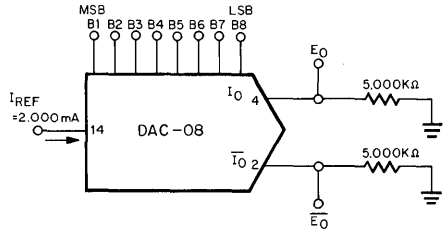
Many older current-output DAC's actually have resistive outputs which must be terminated in a virtual ground. The DAC-08, however, is a true digitally-controlled current source with an output impedance typically exceeding 20 megohms.

Its outputs can swing between −10V and +18V with little or no effect on full scale current or linearity. Some of the applications that require high output voltage compliance include:

1) Precise current transmission over long distances.
2) Programmable current sources.
3) Analog meter movement driving.
4) Resistive termination for a voltage output without an op amp.
5) Capacitive termination for digitally-controlled integrators.
6) Inductive termination with balanced transformers, transducers and headsets.

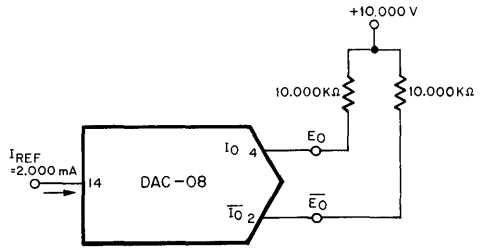


OUTPUT CURRENT VS. OUTPUT VOLTAGE (OUTPUT VOLTAGE COMPLIANCE)



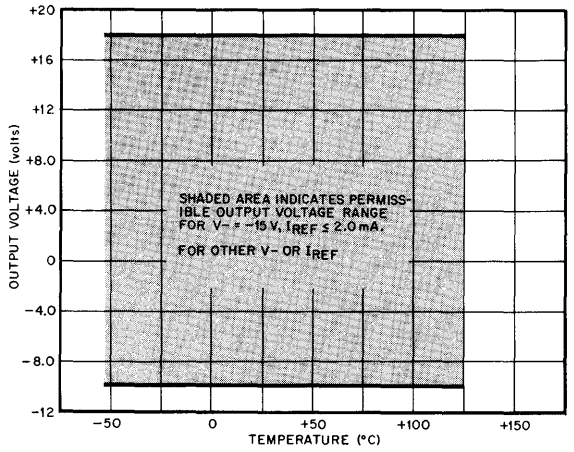
	B1	B2	B3	B4	B5	B6	B7	B8	I_O mA	$\overline{I_O}$ mA	E_O	$\overline{E_O}$
FULL SCALE	1	1	1	1	1	1	1	1	1.992	.000	-9.960	.000
FULL SCALE-LSB	1	1	1	1	1	1	1	0	1.984	.008	-9.920	-.040
HALF SCALE+LSB	1	0	0	0	0	0	0	1	1.008	.984	-5.040	-4.920
HALF SCALE	1	0	0	0	0	0	0	0	1.000	.992	-5.000	-4.960
HALF SCALE -LSB	0	1	1	1	1	1	1	1	.992	1.000	-4.960	-5.000
ZERO SCALE +LSB	0	0	0	0	0	0	0	1	.008	1.984	-.040	-9.920
ZERO SCALE	0	0	0	0	0	0	0	0	.000	1.992	.000	-9.960

BASIC UNIPOLAR NEGATIVE OPERATION

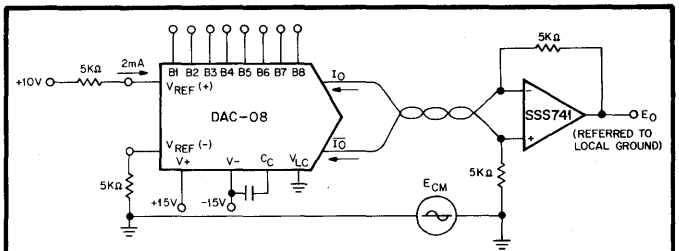


	B1	B2	B3	B4	B5	B6	B7	B8	E_O	$\overline{E_O}$
POS FULL SCALE	1	1	1	1	1	1	1	1	−9.920	+10.000
POS FULL SCALE-LSB	1	1	1	1	1	1	1	0	−9.840	+ 9.920
ZERO SCALE+LSB	1	0	0	0	0	0	0	1	−0.080	+ 0.160
ZERO SCALE	1	0	0	0	0	0	0	0	0.000	+ 0.080
ZERO SCALE-LSB	0	1	1	1	1	1	1	1	+0.080	0.000
NEG FULL SCALE+LSB	0	0	0	0	0	0	0	1	+9.920	− 9.840
NEG FULL SCALE	0	0	0	0	0	0	0	0	+ 10.000	− 9.920

BASIC BIPOLAR OUTPUT OPERATION



OUTPUT VOLTAGE COMPLIANCE VS. TEMPERATURE



	B1	B2	B3	B4	B5	B6	B7	B8	E_O
POS FULL SCALE	1	1	1	1	1	1	1	1	+9.920
POS FULL SCALE-LSB	1	1	1	1	1	1	1	0	+9.840
(+) ZERO SCALE	1	0	0	0	0	0	0	0	+0.040
(−) ZERO SCALE	0	1	1	1	1	1	1	1	−0.040
NEG FULL SCALE+LSB	0	0	0	0	0	0	0	1	−9.840
NEG FULL SCALE	0	0	0	0	0	0	0	0	−9.920

- PROVIDES ISOLATION FROM GROUND LOOPS
- SYMMETRICAL ±10V OUTPUT
- USEFUL WITHIN SYSTEMS BETWEEN BOARDS
- TRUE COMPLEMENTARY/DIFFERENTIAL CURRENT TRANSMISSION
- HIGH SPEED ANALOG SIGNAL TRANSMISSION

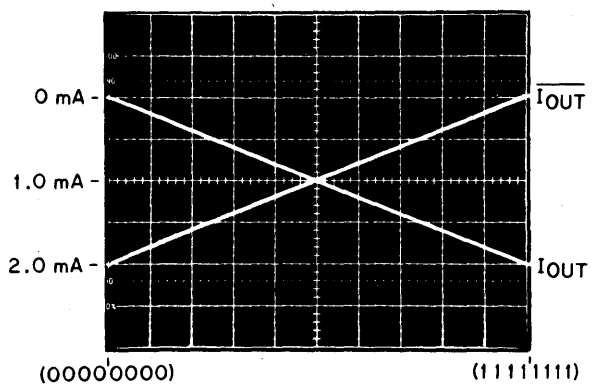
HIGH NOISE IMMUNITY CURRENT TO VOLTAGE CONVERSION

DUAL COMPLEMENTARY OUTPUTS

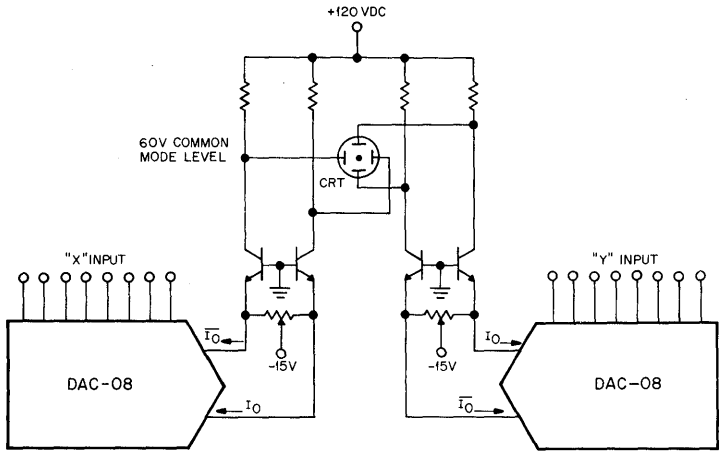
Conventional DAC's have a single output, so they cannot drive balanced loads and are limited to a single input code polarity. The DAC-08 was designed to overcome these limitations

Input coding of positive binary or complementary binary is obtained by a choice of outputs, I_O for positive-true or $\overline{I_O}$ for negative-true. In many applications both are used either independently or in combination. Dual complementary outputs allow some very unusual and useful DAC applications:

1) CRT display driving without transformers.
2) Differential transducer control systems.
3) Differential line driving.
4) High speed waveform generation.
5) Digitally controlled offset nulling of op amps.

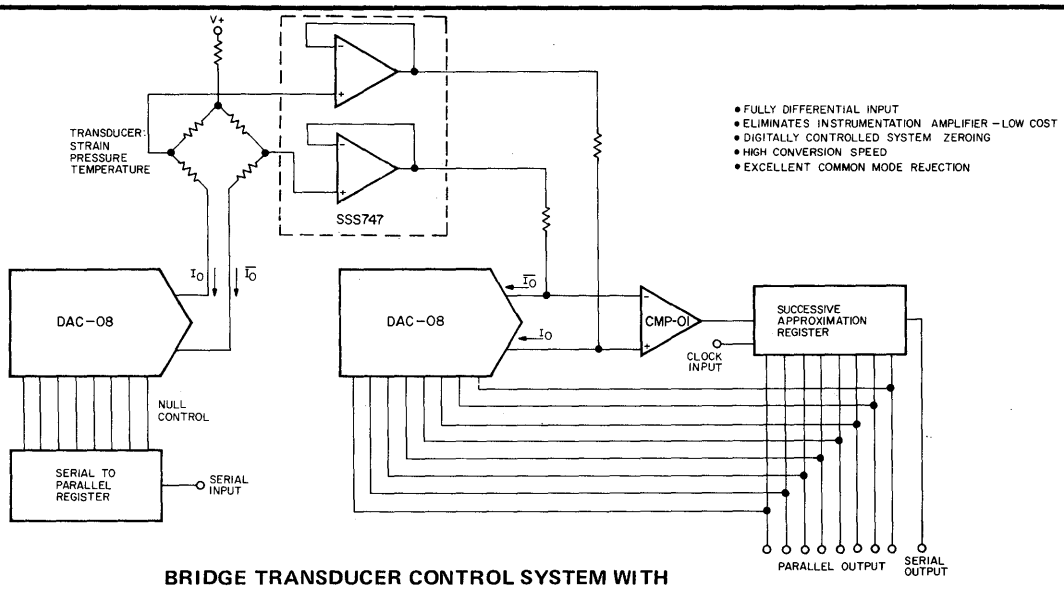


TRUE AND COMPLEMENTARY OUTPUT OPERATION



- FULL DIFFERENTIAL DRIVE LOWERS POWER SUPPLY VOLTAGE
- ELIMINATES INVERTING AMPLIFIERS AND TRANSFORMERS
- INDEPENDENT BEAM CENTERING CONTROLS

CRT DISPLAY DRIVER

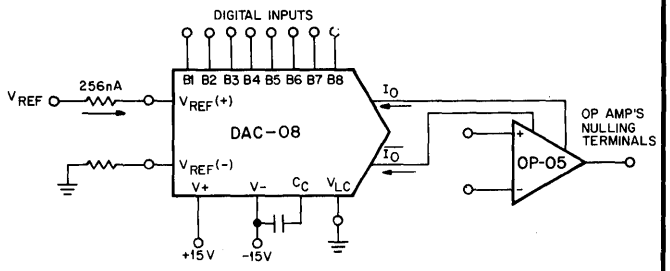


- FULLY DIFFERENTIAL INPUT
- ELIMINATES INSTRUMENTATION AMPLIFIER – LOW COST
- DIGITALLY CONTROLLED SYSTEM ZEROING
- HIGH CONVERSION SPEED
- EXCELLENT COMMON MODE REJECTION

BRIDGE TRANSDUCER CONTROL SYSTEM WITH FULL DIFFERENTIAL INPUT

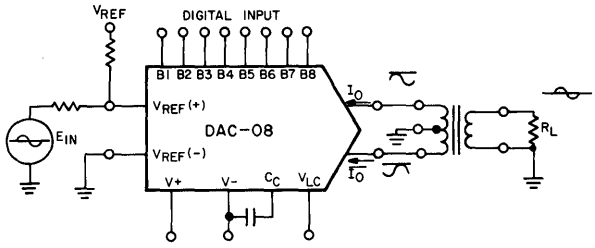
OUTPUT

DUAL COMPLEMENTARY OUTPUTS



- DAC OUTPUT IS 1nA PER STEP
- REPLACES NULLING POTENTIOMETER
- WORKS WITH OP-07, OP-05, SSS725
- V_{OS} NULLED BELOW NOISE LEVEL

DIGITALLY CONTROLLED OFFSET NULLING



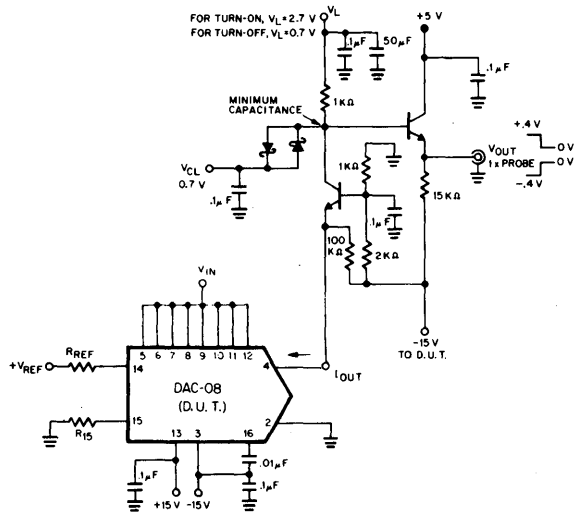
- BIPOLAR OUTPUT WITH OFFSET BINARY CODING
- PROVIDES DC ISOLATION BETWEEN SOURCE AND LOAD
- HIGH VOLTAGE OUTPUT CAPABILITY
- USEFUL WITH PULSE OR SINE WAVE REFERENCE INPUT
- USEFUL WITH PULSE OR SINE FUNCTION DIGITAL INPUT

BALANCED TRANSFORMER DRIVE

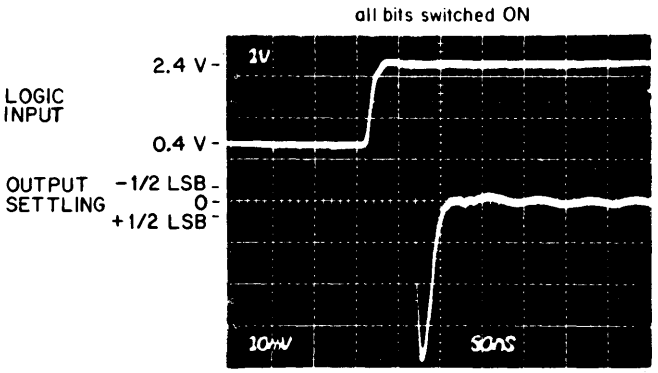
HIGH SPEED

Sub-microsecond settling times are common in current-output DAC's. Many DAC's settle in 500 nsec; 300 nsec is not unusual. But 85 nsec settling time for a low cost DAC is exceptional, and this characteristic allows use of the DAC-08 in formerly difficult and expensive-to-build applications:

1) 1μsec, 2μsec and 4μsec A/D's. (These are completely described in AN-16, available upon request)
2) 15 MHz Tracking A/D's.
3) ECL compatible applications.
4) Video displays requiring a low-glitch DAC.
5) Radar pulse height analysis sytems.



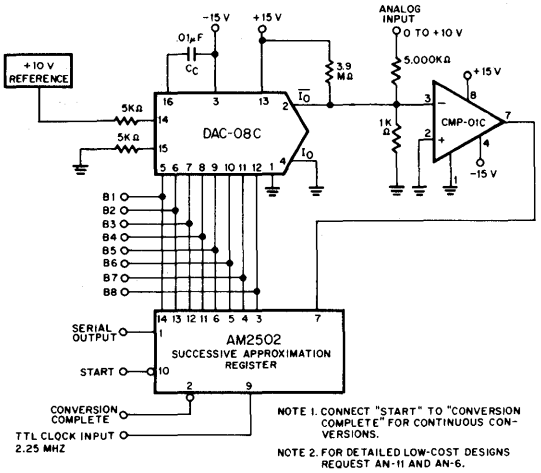
SETTLING TIME MEASUREMENT CIRCUIT



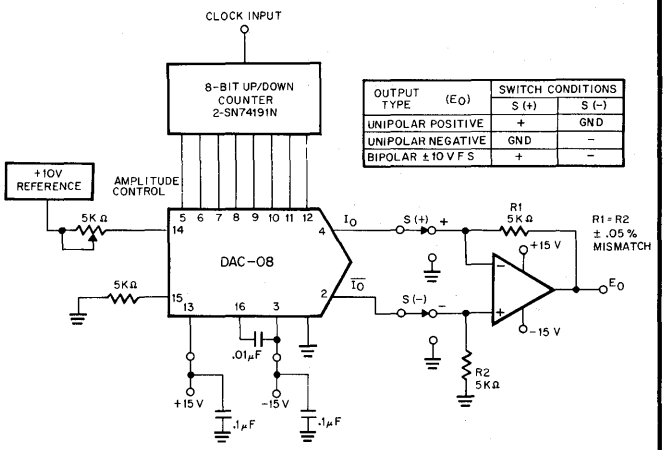
50 nsec/division

I_{FS} = 2mA R_L = 1K Ω
1/2 LSB = 4μA

FULL SCALE SETTLING TIME

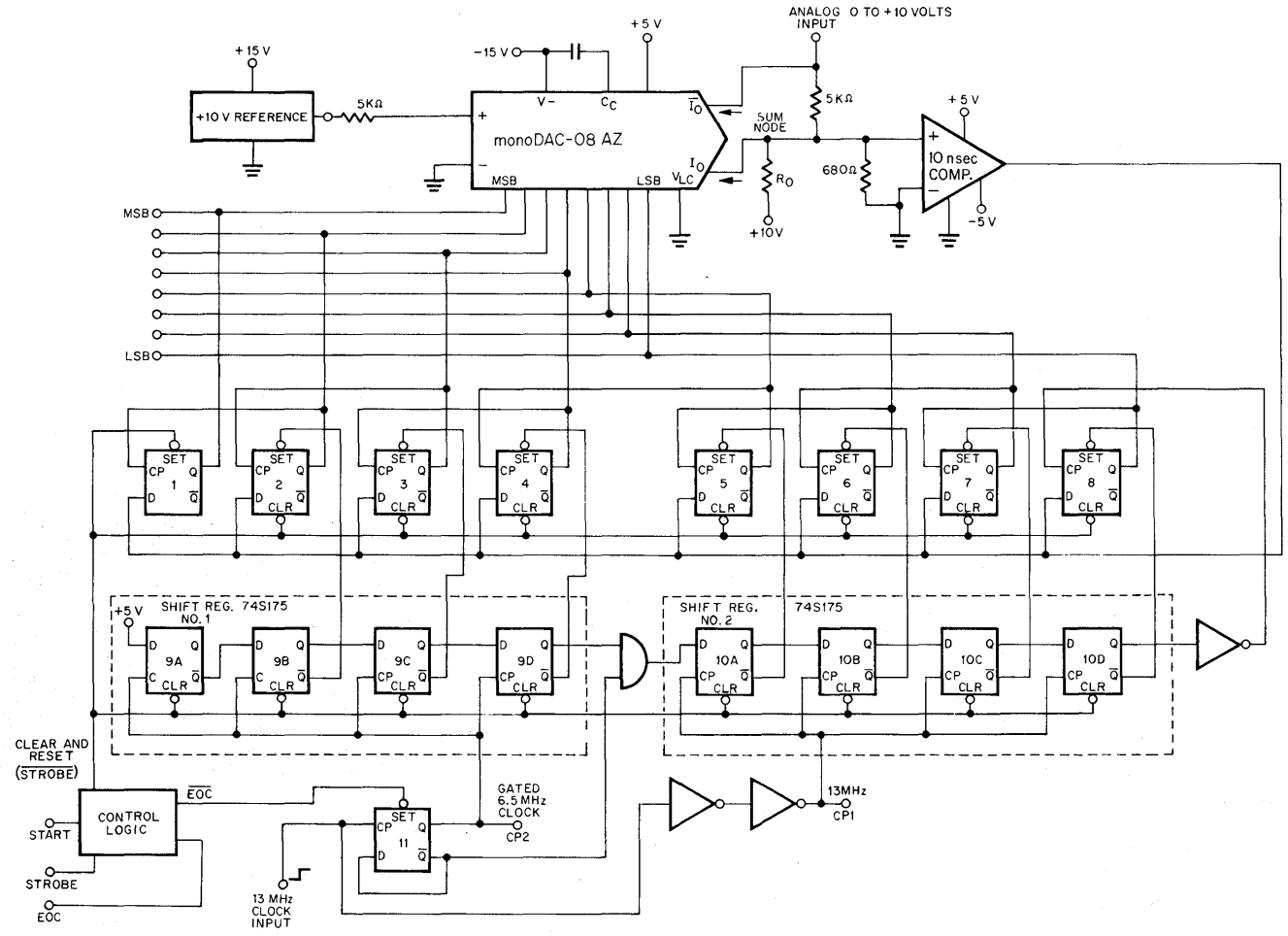


3 IC LOW COST A/D CONVERTER



NOTES:
1. BIPOLAR OUTPUT IS SYMMETRICAL AROUND ZERO, ADJUSTABLE PEAK TO PEAK AMPLITUDE.
2. FOR TRIANGLE WAVE, COUNT UP TO FULL, REVERSE AND COUNT DOWN.
3. FOR POSITIVE-GOING SAWTOOTH, COUNT UP TO FULL, CLEAR, REPEAT.
4. FOR NEGATIVE-GOING SAWTOOTH, COUNT DOWN, CLEAR, REPEAT.
5. FOR OTHER WAVEFORMS, USE A ROM PROGRAMMED WITH THE DESIRED FUNCTION.
6. 85 nsec SETTLING TIME PERMITS WAVEFORM PERIOD OF 25.6 μsec, OR 39 KHz REPETITION RATE.

HIGH SPEED WAVEFORM GENERATOR



SIMPLIFIED SCHEMATIC 1μsec A/D

LOGIC INPUTS

ADJUSTABLE INPUT LOGIC THRESHOLD

Most DAC's have TTL or CMOS compatible inputs which require complicated interfaces for use with ECL, PMOS, NMOS or HTL logic. By contrast, the DAC-08, with typical logic input current of 2μA and an adjustable input logic threshold, interfaces easily with any logic family in use today. The logic input threshold is 1.4V positive with respect to pin 1; for TTL pin 1 is therefore grounded; for other families pin 1 is connected as shown in the interfacing figure. An adjustable threshold and a −10V to +18V input range greatly simplify system design especially with other-than-TTL logic:

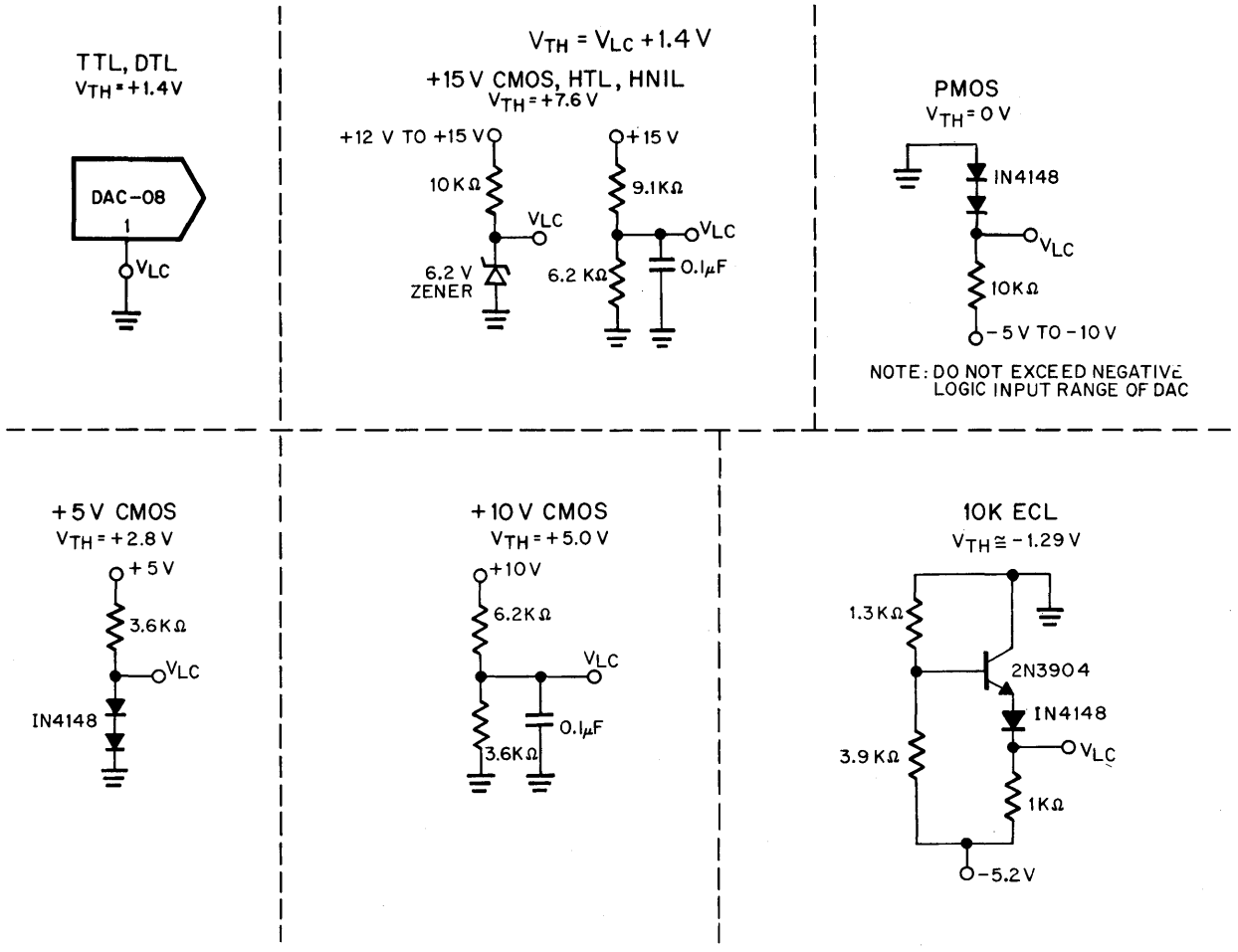
1) ECL applications without level translators.

2) Direct interfaces with Hi-Z RAM outputs.

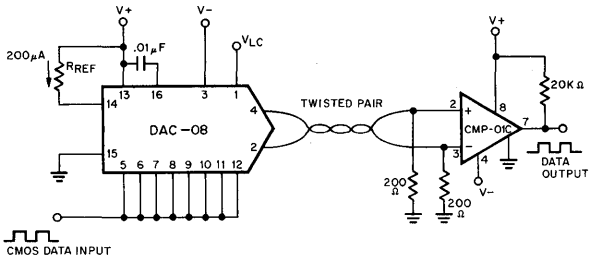
3) CMOS applications without static discharge considerations.

4) HTL or HNIL applications without level translators.

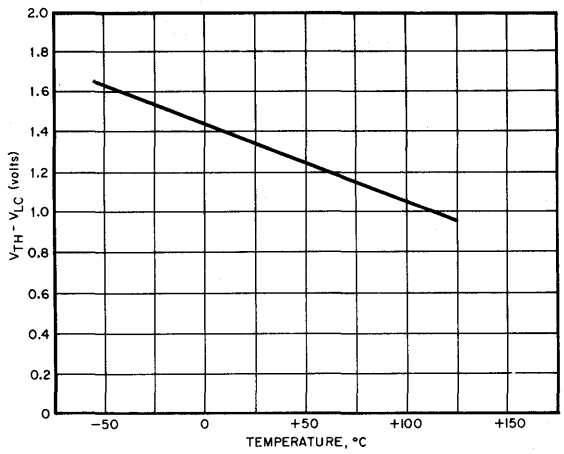
5) System size, weight, and cost reductions.



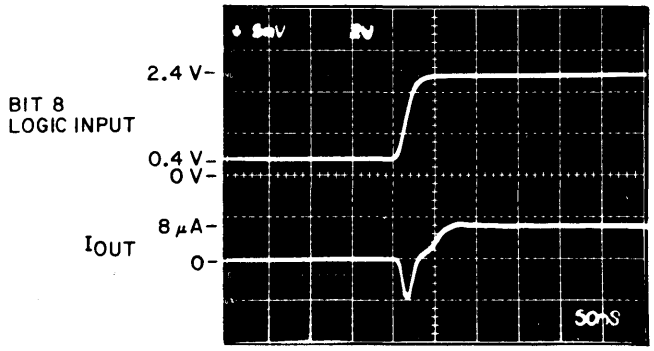
INTERFACING WITH VARIOUS LOGIC FAMILIES



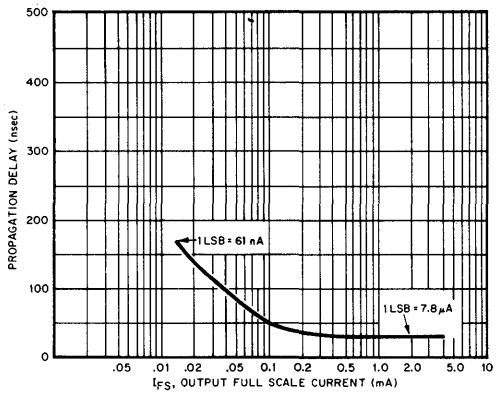
CMOS DIFFERENTIAL LINE DRIVER/RECEIVER



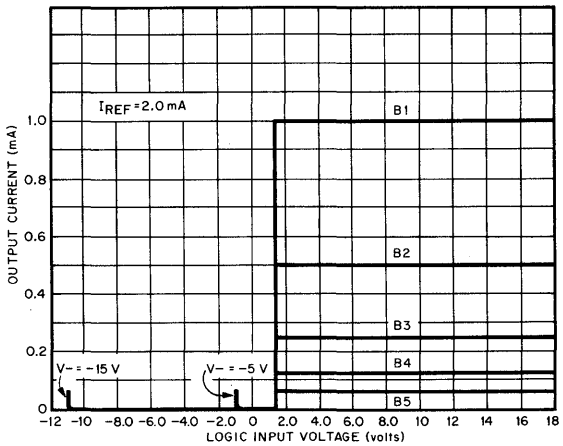
$V_{TH} - V_{LC}$ VS. TEMPERATURE



LSB SWITCHING

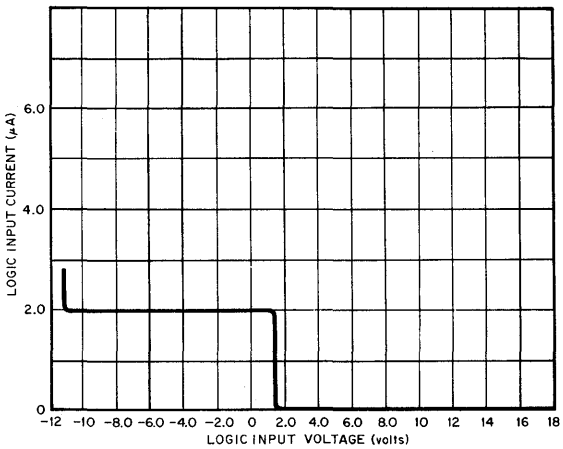


LSB PROPAGATION DELAY VS. I_{FS}



NOTE: B1 THROUGH B8 HAVE IDENTICAL TRANSFER CHARACTERISTICS. BITS ARE FULLY SWITCHED, WITH LESS THAN 1/2 LSB ERROR, AT LESS THAN ±100mV FROM ACTUAL THRESHOLD. THESE SWITCHING POINTS ARE GUARANTEED TO LIE BETWEEN 0.8 AND 2.0 VOLTS OVER THE OPERATING TEMPERATURE RANGE (V_{LC} = 0.0V).

BIT TRANSFER CHARACTERISTICS

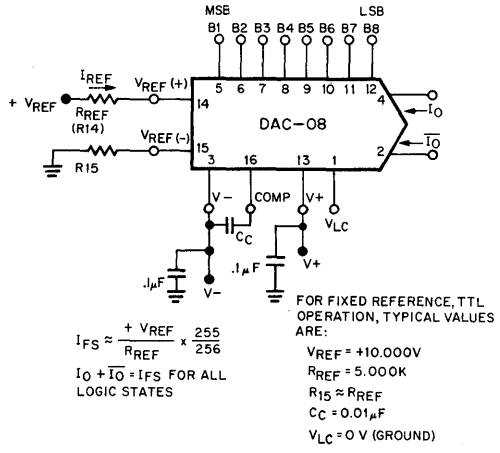


LOGIC INPUT CURRENT VS. INPUT VOLTAGE

MULTIPLYING CAPABILITY

Fixed internal references are included in many DAC's, but they limit the user to non-multiplying, single polarity reference applications and do not allow a single system reference. To achieve the design goals of low cost and total applications flexibility, the DAC-08 uses an external reference. Positive or negative references may be applied over a wide common mode voltage range. In addition, the full scale current is matched to the reference current eliminating calibration in most applications.

1) Digitally controlled full scale calibration.
2) 8 x 8 multiplication of 2 digital words.
3) Digital Attenuators/Programmable gain amplifiers.
4) Modem transmitters to 1 MHz.
5) Remote shutdown and party line DAC applications.



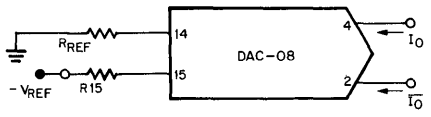
$$I_{FS} \approx \frac{+V_{REF}}{R_{REF}} \times \frac{255}{256}$$

$I_O + \overline{I_O} = I_{FS}$ FOR ALL LOGIC STATES

FOR FIXED REFERENCE, TTL OPERATION, TYPICAL VALUES ARE:

$V_{REF} = +10.000V$
$R_{REF} = 5.000K$
$R_{15} \approx R_{REF}$
$C_C = 0.01\mu F$
$V_{LC} = 0$ V (GROUND)

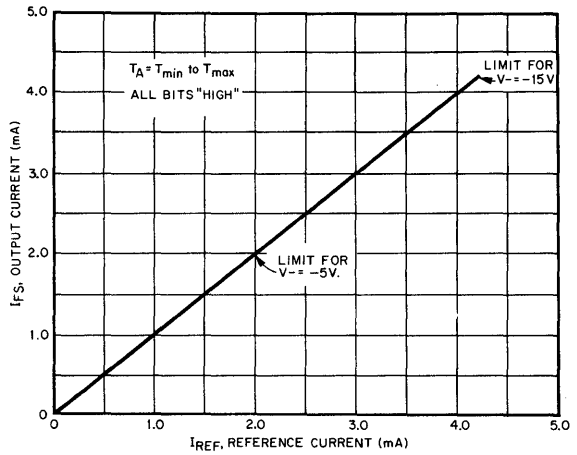
BASIC POSITIVE REFERENCE OPERATION



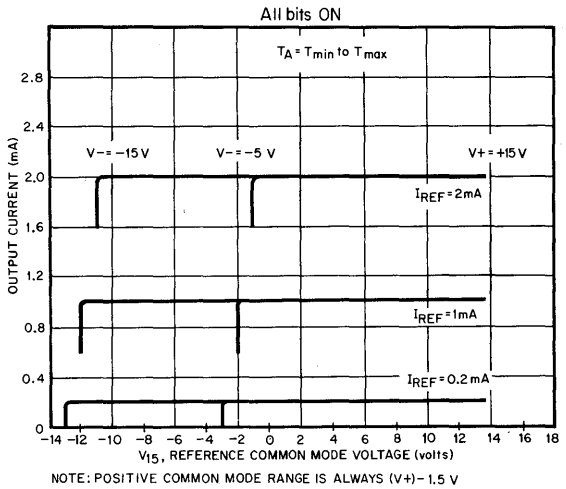
$$I_{FS} \approx \frac{-V_{REF}}{R_{REF}} \times \frac{255}{256}$$

NOTE 1. R_{REF} SETS I_{FS}; R15 IS FOR BIAS CURRENT CANCELLATION.

BASIC NEGATIVE REFERENCE OPERATION

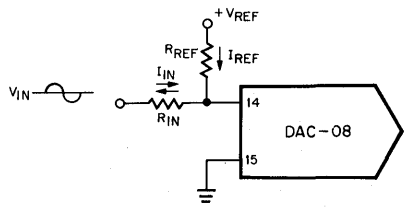


FULL SCALE CURRENT VS. REFERENCE CURRENT

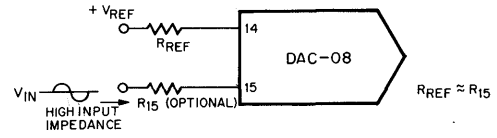


NOTE: POSITIVE COMMON MODE RANGE IS ALWAYS (V+) - 1.5 V

REFERENCE AMP COMMON MODE RANGE

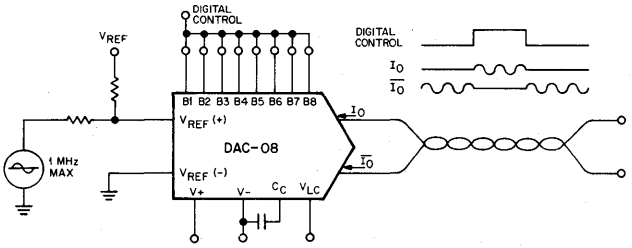


● $I_{REF} \geq$ PEAK NEGATIVE SWING OF I_{IN}



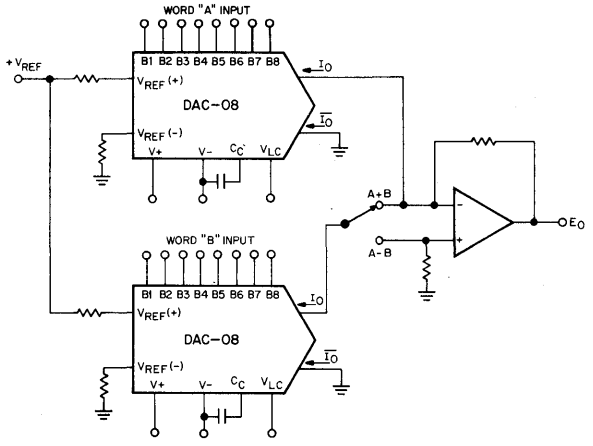
● $+V_{REF}$ MUST BE ABOVE PEAK POSITIVE SWING OF V_{IN}

ACCOMODATING BIPOLAR REFERENCES



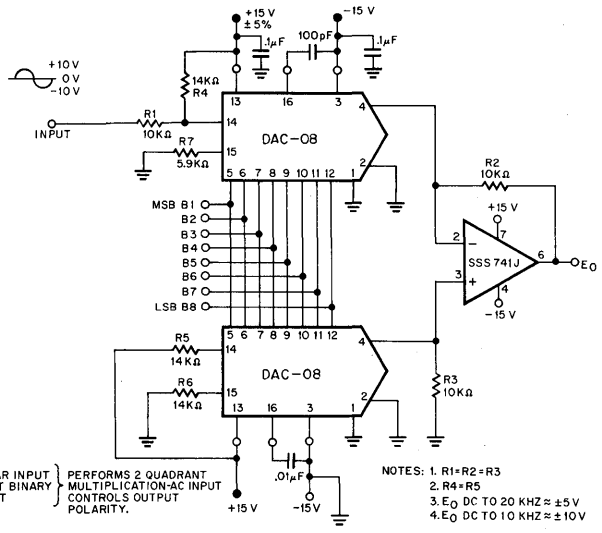
- AC VOLTAGE TO DIFFERENTIAL CURRENT CONVERSION
- DC TO 1MHz INPUT RANGE
- OUTPUT DRIVES TWISTED PAIR DIRECTLY
- CMOS COMPATIBLE

MODEM TRANSMITTER

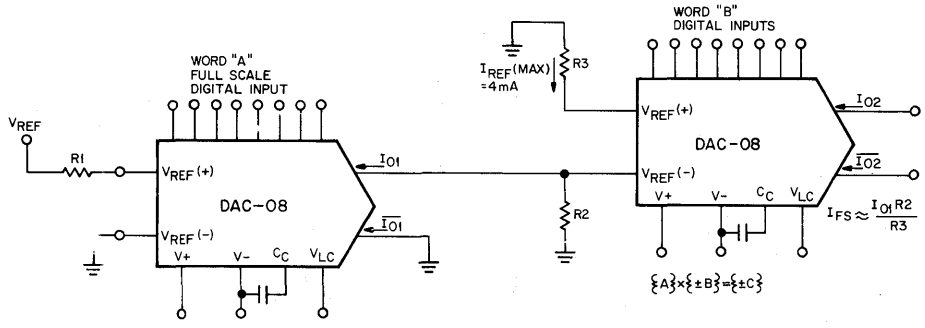


- FAST-85 NSEC PLUS OP AMP SETTLING TIME
- ANY LOGIC FAMILY FOR WORD "A" OR "B"
- BIPOLAR OUTPUT
- ELIMINATES SEVERAL LOGIC PACKAGES

DIGITAL ADDITION OR SUBTRACTION WITH ANALOG OUTPUT



DC-COUPLED DIGITAL ATTENUATOR/ PROGRAMMABLE GAIN AMPLIFIER



- I_{FS} IS THE PRODUCT OF 2 DIGITAL INPUT WORDS
- MAY BE USED AS A 8×8 DIGITAL MULTIPLIER WITH ANALOG OUTPUT
- ELIMINATES DAC AFTER DIGITAL MULTIPLICATION
- FUNCTIONS WITH ANY LOGIC FAMILY
- NOTE: LIMIT WORD "B" INPUT RISE AND FALL TIMES TO 200NSEC MINIMUM

DIGITALLY CONTROLLED FULL SCALE CALIBRATION (MULTIPLIER)

POWER SUPPLY REQUIREMENTS

The DAC-08 works with ±4.5V to ±18V supplies allowing use with all standard digital and analog system supply voltages plus most battery voltages. With only 33mW of power dissipation at ±5V and 85nsec settling time, it has a lower speed power product than CMOS DAC's. Power dissipation is almost constant over temperature, and bypassing is accomplished with 0.01μF capacitors—no large electrolytics are required. These power supply requirements allow:

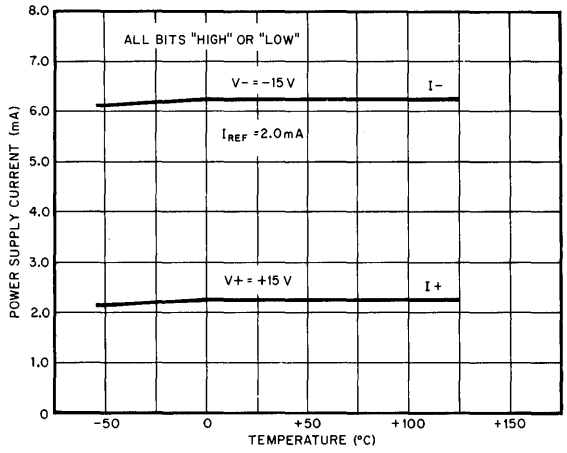
1) Battery operation.

2) Use of unregulated or poorly regulated power supplies.

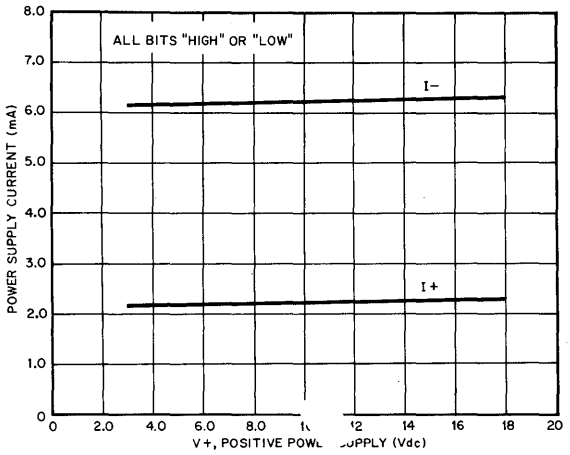
3) Use in space-limited areas due to small bypass capacitors.

4) Use in constant power dissipation applications.

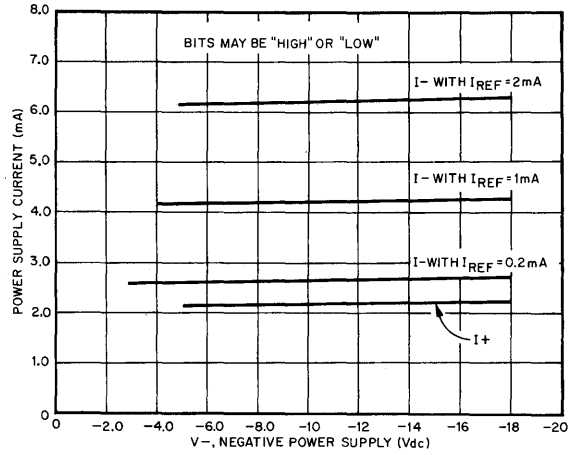
5) Common digital and analog power supplies.



POWER SUPPLY CURRENT VS. TEMPERATURE



POWER SUPPLY CURRENT VS. V+



POWER SUPPLY CURRENT VS. V–

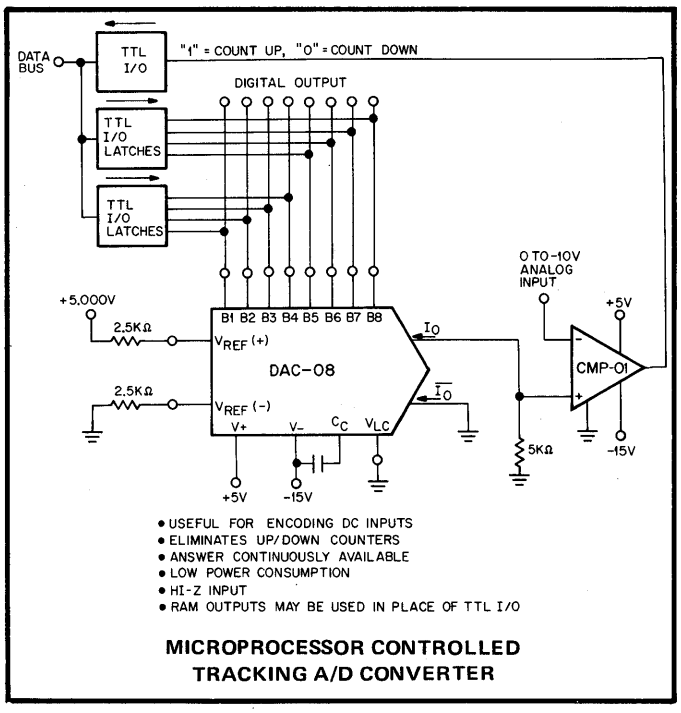
OTHER APPLICATIONS

MICROPROCESSOR APPLICATIONS

The ability to use μP power supply voltages and the ability to interface with any logic family make the DAC-08 especially useful in μP applications:

1) Tracking A/D converters.
2) Successive approximation A/D converters.
3) Direct drive from Hi-Z MOS RAM outputs.

By programming the ROM's with the successive approximation or the tracking A/D algorithm, all of the logic for A/D conversion is contained in the μP. This is a very inexpensive approach, since there is no need for the usual A/D conversion logic packages.



- USEFUL FOR ENCODING DC INPUTS
- ELIMINATES UP/DOWN COUNTERS
- ANSWER CONTINUOUSLY AVAILABLE
- LOW POWER CONSUMPTION
- HI-Z INPUT
- RAM OUTPUTS MAY BE USED IN PLACE OF TTL I/O

MICROPROCESSOR CONTROLLED TRACKING A/D CONVERTER

OTHER APPLICATIONS: The following list summarizes just a few of the many applications for this flexible DAC. Consult the factory for further information.

A/D CONVERTERS

Tracking (Servo)
Successive Approximation
Ramp (Staircase)
Microprocessor Controlled
Ratiometric (Bridge Balancing)

TEST SYSTEMS

Transistor Tester (Force I_B and I_C)
Resistor Matching (Use both outputs)
Programmable Power Supplies
Programmable Pulse Generators
Programmable Current Source
Function Generators (ROM Drive)

ARITHMETIC OPERATIONS

Analog Division by a Digital Word
Analog Quotient of Two Digital Words
Analog Product of Two Digital Words—Squaring
Addition and Subtraction with Analog Output
Magnitude Comparison of Two Digital Words
Digital Quotient of Two Analog Variables
Arithmetic Operations with Words from Different Logic Families

GRAPHICS AND DISPLAYS

Polar to Rectangular Conversion
CRT Character Generation
Chart Recorder Driver
CRT Display Driver

DATA TRANSMISSION

Modem Transmitter
Differential Line Driver
Party Line Multiplexing of Analog Signals
Multi-level 2-Wire Data Transmission
Secure Communications (Constant Power Dissipation)

CONTROL SYSTEMS

Reference Level Generator for Setpoint Controllers
Positive Peak Detector
Negative Peak Detector
Disc Drive Head Positioner
Microfilm Head Positioner

AUDIO SYSTEMS

Digital AVC and Reverberation
Music Distribution
Organ Tone Generator
Audio Tracking A/D

CONCLUSION

High voltage compliance complementary current outputs, universal logic inputs and multiplying capability make the Precision Monolithics DAC-08 the most versatile monolithic high speed DAC available today.

APPLICATION NOTES AVAILABLE UPON REQUEST			
AN-2	"Monolithic Chip Assembly Information"	AN-12	"Temperature Measurement Method Based On Matched Transistor Pair Requires No Reference"
AN-5	"Applying a Monolithic 10-Bit D/A Converter"	AN-13	"The OP-07 Ultra-Low Offset Voltage Op Amp—A Bipolar Op Amp That Challenges Choppers, Eliminates Nulling"
AN-6	"A Low Cost, High-Performance Tracking A/D Converter"	AN-14	"Interfacing Precision Monolithics Digital-To-Analog Converters With CMOS Logic"
AN-9	"Instrumentation Operational Amplifier with Low Noise, Drift, Bias Current"	AN-15	"Minimization of Noise in Operational Amplifier Applications"
AN-10	"Simple Precision Millivolt Reference Uses No Zeners"	AN-16	"High Speed A/D Conversion Using The DAC-08"
AN-11	"A Low Cost, Easy-To-Build Successive Approximation Analog-To-Digital Converter"		

3. DTL Peripheral/Power Drivers - "A Giant Step Backward"

Reverting to DTL Adds New Dimensions To Sprague Peripheral/Power Drivers

Paul R. Emerald
Manager, Applications Engineering
Digital Products
Sprague Electric Company
Worcester, Mass.

Introduction

Although not widely known in the industry, Sprague Electric was one of the pioneers with Peripheral/Power Driver ICs; the Sprague UHP 400/500 series quad drivers being introduced in 1970. Concurrently Texas Instruments was bringing out its 75450 series; and, although the circuit types are quite similar, each is targeted at somewhat different markets. The TI dual drivers are high speed, lower output breakdown devices chiefly sold in 8 lead mini DIPs; while the Sprague power drivers were introduced as quad, high voltage, moderate speed devices supplied in a 14 lead DIP.

The design and manufacturing of a quad driver meant rather dramatic deviations from any standard bipolar processing/packaging/circuit design techniques in use at the time. To allow the use of four high current outputs in the same DIP necessitated developing high beta, high current, high voltage output NPNs; chiefly done to minimize package power dissipation when confronted with a quad 14 LDIP rather than an 8 lead dual mini DIP driver. Reducing the I_{CC} drive power for the output transistor via improvements in high current beta significantly affected the overall package power.

Further it was necessary to utilize a copper alloy lead frame for the DIP; another requirement if the units were to be used in systems requiring the four outputs to be energized continuously and simultaneously. The standard Kovar (iron alloy) in use definitely would not allow this, but the change to a copper alloy package frame reduced the DIP Øja (junction to ambient rise) from +125°C/W (Kovar) to a figure of +60°C/W (copper). The combination of the high beta process and dramatic improvement in package technology allowed the Sprague quad power drivers to simultaneously and continuously switch 250mA in all four outputs, in a +70°C ambient without exceeding a junction temperature of +150°C.

The Sprague Electric UHP 400/500 series quad drivers have been an industry standard for the past few years. Schematically the UHP 407/507 is shown in Figure 1; these ICs are available with NAND, NOR, AND, and OR logic gates and with three output breakdown minimums, the 400 series at 40 volts, 400-1 series at 70 volts, and 500 series with 100 volt output breakdown. Additionally there are four basic relay driver types which incorporate internal transient suppression diodes for use with inductive loads. All types are guaranteed to sink a minimum of 250mA with a $V_{CE}(sat)$ of less than 0.7 volts, and all are compatible with TTL and DTL logic families.

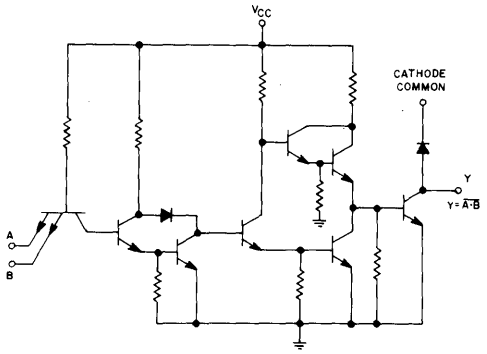


FIGURE 1 407/507 SCHEMATIC

Electromechanical Loads

Primarily the Sprague high current/high voltage drivers have been designed for use with electro-mechanical loads. No attempt has been made to produce a high speed device; moderate switching speeds result in less noise generation during output transitions. Virtually all peripheral loads are much slower than any semiconductor switching device, and it is neither necessary nor desirable to utilize high speed switching for interface to electromechanical loads. Needs for decoupling and/or critical PC board layout are diminished with slower switching devices coupled with the serious attempts to minimize logic circuit power (I_{CC} 0).

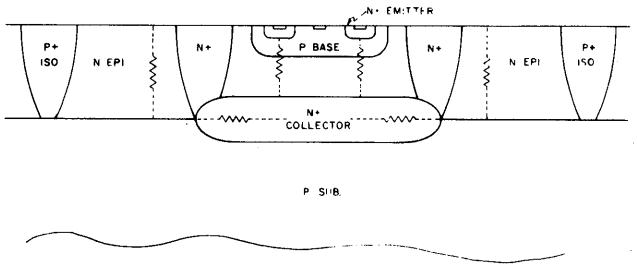
High Voltage/High Current ICs

Many design and process changes were imposed to create the Sprague drivers; to obtain the high voltage output NPNs it was necessary to modify the N doped epitaxial layer. The epi layer has been increased in both thickness and in resistivity to provide higher output breakdown. These changes meant that these ICs more resembled linear circuit processing than TTL digital process techniques.

Transistor design tolerances were quite dramatically changed to sustain the much higher "OFF" voltages required of these circuit applications. Much deeper junctions with greater curvature and rounded transistor diffusions (another element of the curvature) were part of the design modifications for improving breakdown voltages. Through use of these techniques it became possible to achieve breakdowns of 100 volts very repeatably.

A combination design/process addition was also instituted to minimize the output NPN saturation voltage. The use of higher resistivity epitaxial layers would have adversely affected the collector/emitter $V_{CE(sat)}$ were it not for the use of a highly doped, low resistivity collector diffusion. Necessary was the addition of a low resistivity N+ collector plug which is driven sufficiently to contact the N+ buried (floating) collector diffusion beneath the NPN base.

FIGURE 2 CROSS SECTION MONOLITHIC POWER NPN



The cross sectional view of Figure 2 indicates the series collector resistance associated with the buried N+ collector plug. Without the low resistance collector diffusion the series resistance between the buried N+ collector and the metalization on the chip surface would be much higher. Through use of the N+ collector diffusion, which also serves as an N+ surface guard ring to prevent unwanted inversion of the epi, it is possible to achieve an R_{sat} of less than 1 ohm (very large, high current circuits) to a more standard 2 to 3 ohms collector resistance.

Package Availability

The original quad power drivers (400/500 series) have been supplied to commercial/industrial users in the previously mentioned plastic DIP using a copper lead frame, thus allowing use of all four outputs simultaneously and continuously at 250mA each. Also supplied, primarily for military systems, have been two types of fully hermetic packages; a 14 lead ceramic DIP, and a 14 lead flat-pack - both manufactured to meet the requirements of MIL STD 883.

The newer additions to this product family, the DTL UDN 3600M and 5700M dual drivers, have added a new package type. The series of dual devices is supplied in a copper frame 8 lead mini DIP, and the 3600 series units are pin for pin with the National LM 3611 etc., as well as the higher speed TI types (75451 etc.).

The new DTL quads (UDN 5700A series) also have added a new package - the 16 lead copper frame DIP. Use of the 16 LDIP was largely predicated upon the desire for all gates to have separate inputs, rather than a pair connected (strobed) together, thus somewhat improving the versatility of the quads. All the newer plastic power drivers benefit greatly from the improved Øja of the copper frame package.

The new DTL types with their improved electrical parameters will also be supplied in fully hermetic packages to MIL 883 for use in adverse environments or military systems requiring a full temperature range unit and/or hermeticity. Some reduction in package dissipation potential results, but the ease of interface to logic families such as CMOS with the newer DTL circuits will benefit those unable to obtain standard TTL or CMOS logic with either sufficient output current and/or the high breakdown capability (80 volts) of these ICs.

Standard package power curves are shown in Figures 3A (plastic) and 3B (hermetic). The curves of Figure 3A compare the two copper lead frame DIPs with the capability of the Kovar frame packages used for standard lower power circuits. Any area beneath the appropriate package curve represents allowable average power and is plotted against ambient temperature to a +85°C limit. It is quite apparent that both the mini DIP at +80°C/W and the 14 or 16 lead quads at 60°C/W with copper frames are much superior to the Kovar DIP with a rating of +125°C/W.

FIGURE 3A UHP/UDN PLASTIC PACKAGES

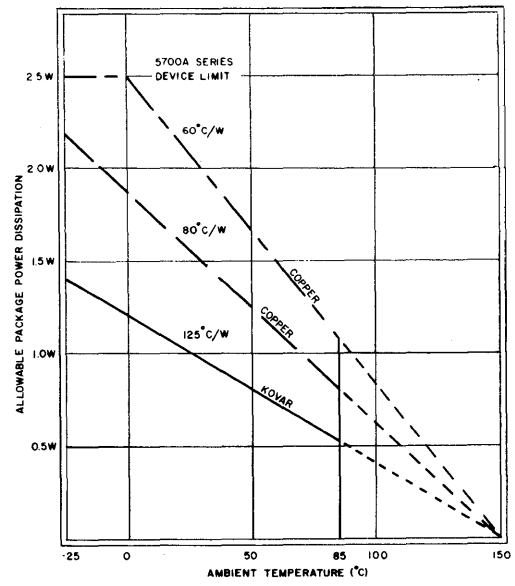
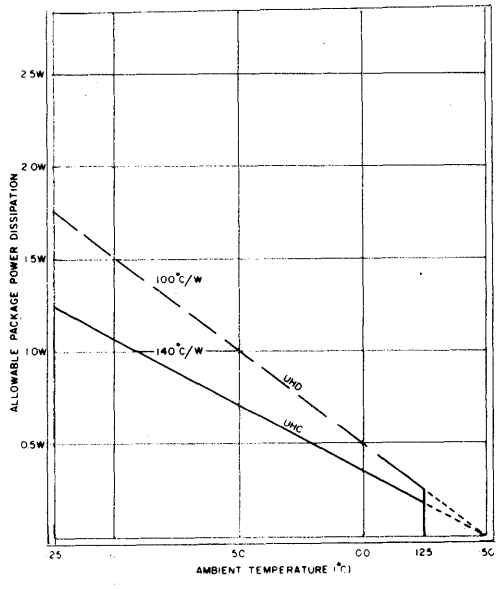


FIGURE 3B UHC/UHD 400/500 SERIES



The curves of Figure 3B apply to the hermetic packages used with the series 400/500 quads. This shows that the differential in the materials used and the size of the package have a considerable bearing on the package dissipation limits. The flatpack is the poorest of all, and should not be used unless there is insufficient space (chiefly height) for the hermetic DIP. With either type the average power is reduced, particularly if the ambient temperature is to reach +125°C.

These Øja rating curves show the maximum junction temperature rise with a unit of power applied; the plastic types are generally restricted to uses below +85°C, while the hermetic units may be used up to +125°C. In either case the upper junction limit will not exceed a +150°C level, and is shown by the dashed lines extending to zero (0)power at +150°C. Operating within these ambient temperature/average power limits will insure that the junction temperature does not exceed +150°C, since variations in manufacturing and electrical parameters are guardbanded.

DTL Logic - "A Giant Step Backward"

Through ongoing evolutionary improvements the power drivers have seen a series of improvements; the most notable being a conversion to a DTL logic gate in the UDN 3600 and 5700 series. Others have been made affecting the high current output NPN and a small reduction in the typical I_{CC} 0 power.

Converting the logic gate (Figure 4 - schematic 5707/5712) to a diode input (DTL) has brought new potential applications into the picture, particularly the use with CMOS and PMOS circuitry. Prime advantages are associated with the much higher input voltages allowed (up to 30V), and the extremely low logic "0" input current (100μA maximum vs 1.6mA for a standard TTL). This comparison is between the new, improved Sprague DTL types and the TI 75451 types and the LM 3611 series by National. The earlier Sprague UHP 400 family had an industry low of 800μA (1/2 TTL load), and that has subsequently been reduced further to a 100μA maximum (200μA on strobe inputs) thus allowing use with CMOS and PMOS logic; while the TI and National TTL types have the standard 1.6mA TTL maximum.

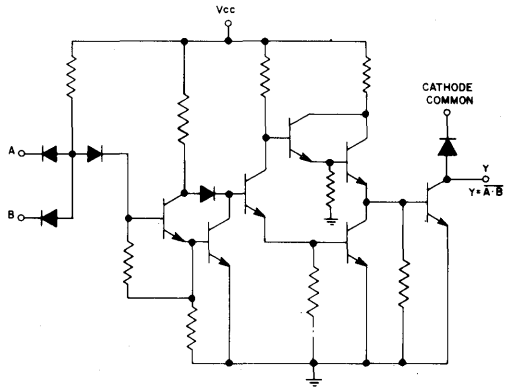


FIGURE 4 5707/5712 SCHEMATIC

Lamp Interface - Inrush Current

One type of application these IC drivers are well suited to is the switching of incandescent lamps. Of great concern to those using semiconductor devices for switching lamp filaments is the high inrush (surge) current associated with cold lamp filaments. When switched with either mechanical or electro-mechanical switches this inrush current may reach a value that is 1000% to 1200% of the nominal steady state current. This 10-12X the nominal can be disastrous to lower current ICs since they are unable to sustain the instantaneous currents of a cold filament and are prone to destruction; usually resulting from secondary breakdown during turn on of the output.

A technique frequently employed to obviate this type of failure with lower current devices is the use of "warming" resistors across the switching output. Maintaining the filament partially warmed reduces the inrush (surge) current in the transistor, but it complicates designs considerably when a large number of lamps are used. The high current peripheral/power drivers are able to sustain these momentary inrush currents; and, hence, it is <u>not</u> necessary to add one "warming" resistor for each lamp.

The Sprague power drivers have been widely used since 1970 in systems employing them as lamp drivers, and the most typical is a parallel pair of 28 volt, 40mA lamps (#327 or #387) switched by each NPN output. Even though none of the standard IC peripheral/power drivers available will be in saturation under conditions of lamp inrush, they have been designed to sink currents of 300mA (well saturated) and are able to sustain the momentary high power. The transistor design and process chosen precludes these drivers from failure due to secondary breakdown when used in a conventional manner.

The current at which the output comes out of saturation is related to the transistor design (chiefly emitter periphery) and process related variables: beta of output NPN, diffused resistor tolerances (determine base current of output), etc. As the graph of inrush current shown in Figure 5 indicates, the time necessary to reach a current level within the device saturation level is less than 5 milliseconds. Even assuming the use of two (2) lamps in parallel, which is the typical case (current value is thus skewed by a factor of two), the outputs of these power drivers will need only sustain an inrush period of approximately 2 to 3 milliseconds (intersection at .12A). The newer DTL units with a guaranteed output saturation voltage at 300mA are somewhat better than the earlier UHP 400/500 series, although neither has given trouble with inrush currents when used with #327 or #387 lamps.

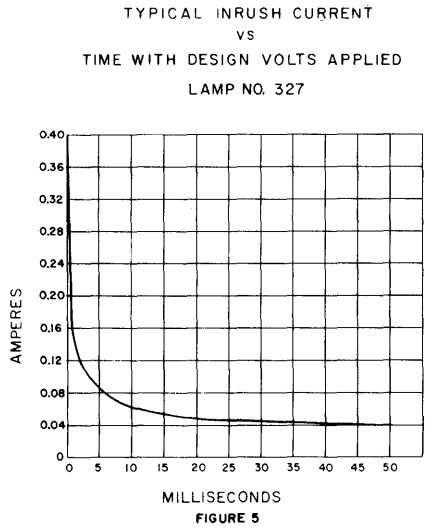


FIGURE 5

Lamp Interface - DC

Perhaps the most frequent application (shown Figures 6A and 6B) of these ICs has been the interface to aircraft type lamps such as the #327 and #387, both of which are rated at 28 volts and 40mA. Usually two are connected in parallel (reasons of redundancy), and run from a 28 volt DC supply. As previously mentioned the full inrush is not likely to be reached due to the limitations of the device output sinking capability and its modest switching speeds (compared with TTL logic). Maximum inrush currents of a 10-12X nominal are typically reduced to a value approximating half that level (5-6 times steady state).

Though use of one of the relay driver versions with its integral suppression diodes a simple lamp test may be accommodated without the need to use an input on the logic gate. Figure 6A shows the use of an input from each gate for lamp test purposes, but the same function may be achieved using the suppression diodes switched to ground as in Figure 6B. By switching the diode common cathode line with an electromechanical or transistor switch the need for the logic input is obviated; a scheme that requires only one connection per package rather than four (quads) or two (duals).

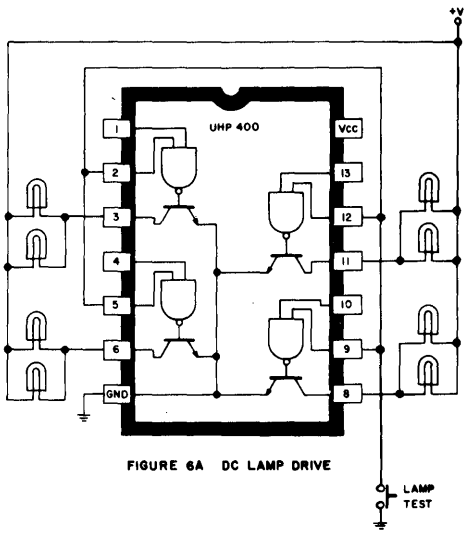


FIGURE 6A DC LAMP DRIVE

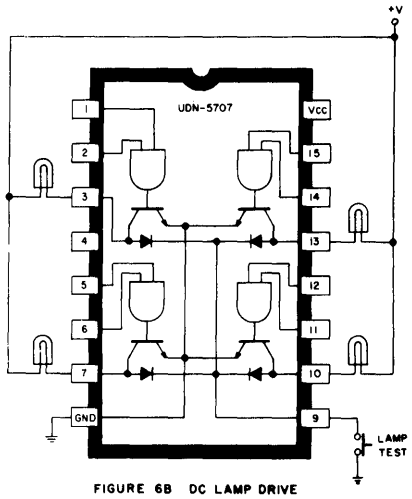


FIGURE 6B DC LAMP DRIVE

Lamp Interface AC (Half Wave)

In Figures 7A and 7B are two potential configurations for operating the lamp driver from an AC source using only half wave rectification. Either of the two diode configurations is permissable and choice largely depends upon whether a common ground exists in the system.

FIGURE 7A AC/HALF WAVE

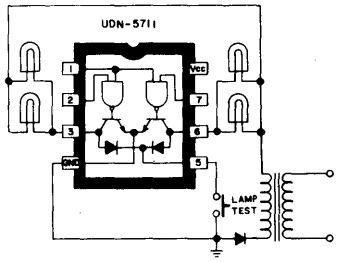
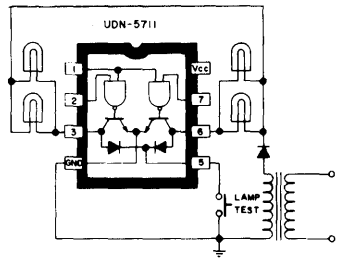


FIGURE 7B AC/HALF WAVE



Light output will be reduced somewhat when run in a half wave mode, as will the filament temperature. It should still suffice for most ambient conditions excepting, specifically, sunlight.

An element of caution should be observed when operating from an AC source rather than a DC supply: The inductance of the transformer may produce voltage transients beyond the device breakdown level, particularly when all lamps are switched OFF simultaneously. Use of such items as back-to-back zeners for clamping, a GE MOV™, or sufficient capacitance across the

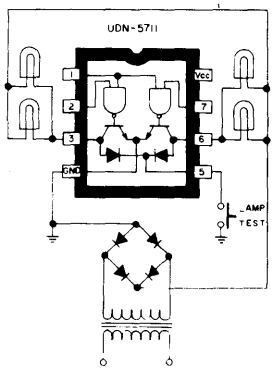
supply to prevent the transients will solve the problem. The capacitance required would be a function of the total lamp current, but it should not be necessary to achieve any appreciable degree of filtering.

Preventing these inductive transients through use of zero-crossing devices would be another approach, albiet a more complex solution.

Lamp Interface AC (Full Wave)

Inexpensive bridge rectifiers allow the approach shown in Figure 8, and will provide greater lamp brightness than the half wave counterpart. In either case the simplicity of lamp drive without the need for well filtered, regulated power supplies is apparent. Again the same cautions pertaining to the transformer inductance mentioned in the half wave AC section apply for full wave rectification.

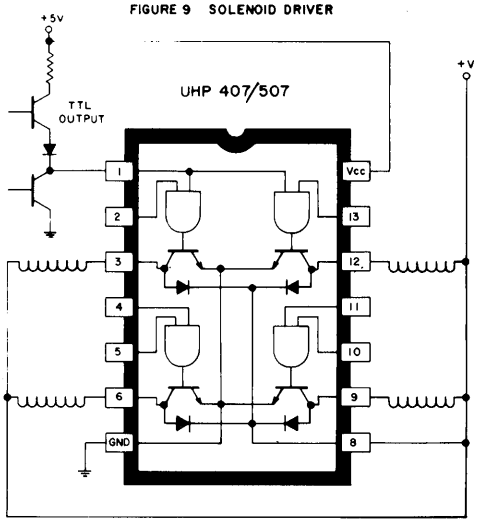
FIGURE 8 AC/FULL WAVE



Solenoid/Relay Interface

The use of the integral suppression diodes originated in the Sprague UHP 400/500 series of relay drivers is shown in Figure 9. The newer UDN 5700M dual and UDN 5700A quad peripheral/power drivers all include high current/high voltage transient (flyback) diodes for use with inductive loads. The obvious advantage is the reduction in component count, board space, and assembly costs that result when choosing a Sprague relay driver over a non-diode protected type.

FIGURE 9 SOLENOID DRIVER



$V_{CEX(sus)}$ Curve

Operating inductive loads requires that any semiconductor sustain a combination of output voltage and current as the load is switched OFF; a combination of voltage/current not required with non-inductive loads. The collapse of the inductive load produces a very non-linear, "looping" load line, and it is desirable that it not intersect the device output breakdown curve. The phantom of secondary breakdown creates device failures when the voltage/current (power) combines with an excessive interval (time). Secondary breakdown is a power/time related failure mechanism, but the intersecting of a breakdown curve by an inductive load line is at times permissable. It is, however, definitely advantageous to not have such an intersection (crossing) occur during turn off.

Figure 10 shows a typical $V_{CEX(sus)}$ curve obtained with the UDN 3600 or 5700 series of IC drivers. Per the V_{CC} notation this curve is obtained with the supply equal to, or greater than, +4.75 volts. Inductive load lines vary greatly with load current and voltage; and, hence, are difficult to include. The load line of a solenoid or relay may be obtained by properly connecting an oscilloscope to monitor load current and voltage while repetitively switching the coil. Use a current probe to monitor load current (vertical input) while applying the voltage waveform to a calibrated horizontal input; this will display the load line.

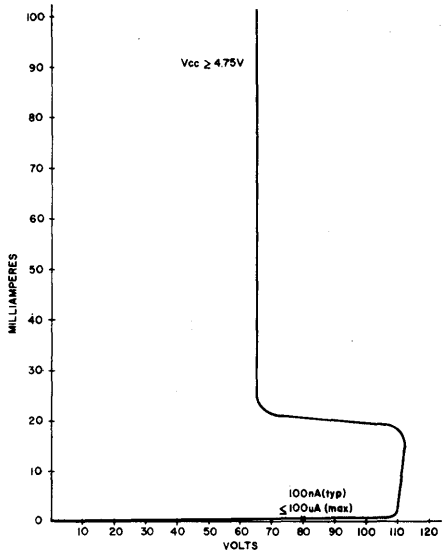


FIGURE 10 TYPICAL $V_{CEX(sus)}$ CURVE

Paralleling Outputs

Due to the excellent matching provided by monolithic IC processing and identical output transistor designs it is possible to parallel output/input combinations to allow higher current sinking as systems necessitate. All four outputs may be paralleled with results like those shown in Figure 11 for 2, 3, or 4 drivers in parallel. Current sharing is extremely well matched, and the best choice for pairing would be outputs on the quads across from one another (example: output pin #3 with output pin #11 with a UHP 400).

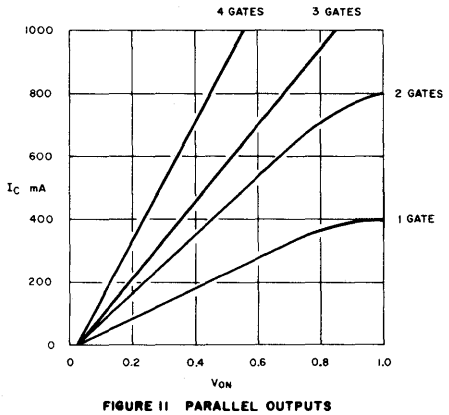
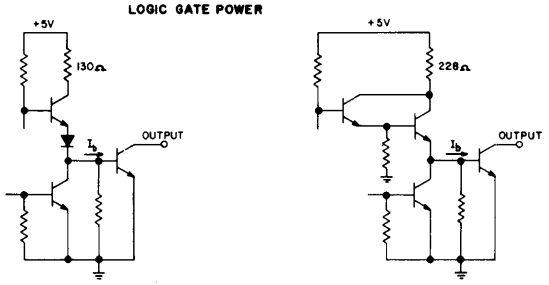


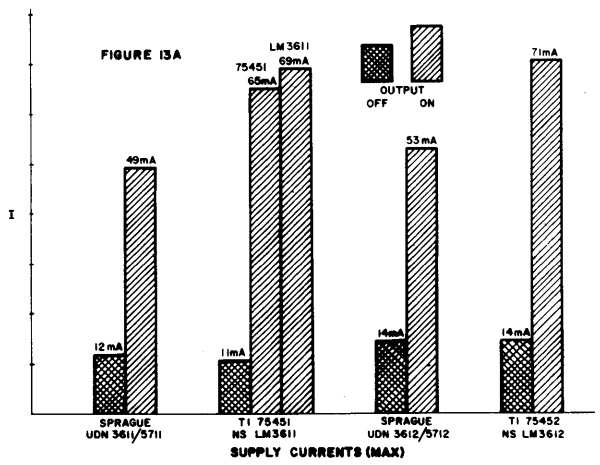
FIGURE 11 PARALLEL OUTPUTS

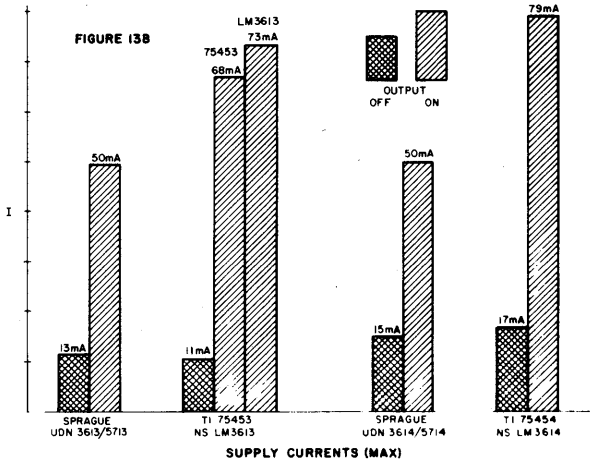
Logic Gate Power

The high gain transistor process used at Sprague has resulted in a considerable decrease in the power dissipated in the logic gate which sources base current for the output NPN. The collector resistor in the TTL totem pole output is the determining component for the high current output; and it is this resistor in each of the gates that has a principal influence on the power when the output is switched on (supply current/output low). Figures 12A compares the TI or National logic output (130 ohm resistor) to the Sprague types (228 ohm resistor), and it is this 228 ohm resistor value in Figure 12B that decreases package power substantially.



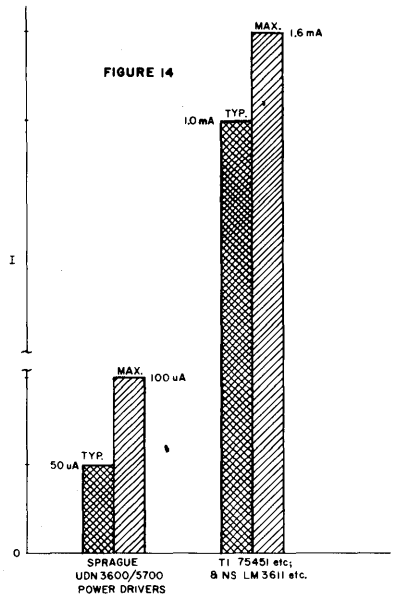
The histograms of Figures 13A and 13B compare the dual driver types supplied by Sprague, TI, and National. Note that the output ON currents of the TI and National circuits are either identical, or very similar; but that both are considerably higher than the Sprague equivalents. A small discrepancy exists between the output OFF currents, but these are of much less consideration than the output ON supply current. The Sprague units offer an obvious advantage of less heat and lower current supplies while providing the same functional capability for interface to peripheral loads. Reductions in power (heat) can appreciably affect some system considerations (largely power supplies) and/or improve reliability via lower operating temperatures.





Logic "0" Input Current - DTL Types

The reversion to a DTL logic gate with an allowable input logic level of up to 30 volts brings new dimensions to the applications for the newer dual and quad power drivers. TTL circuits are difficult to use in systems that have logic "1" input levels beyond the 5.5 volt TTL maximum specified; but even more difficult to handle with many MOS devices is the maximum logic "0" current of 1.6mA for 7400 TTL. In Figure 14 a comparison of the Sprague DTL types with a 100μA maximum is made to the much higher (1.6mA max) levels of the TI and National types.



PMOS Interface - DTL Types

The combination of an allowable input voltage of up to 30 volts and the 100μA input current for the logic "0" state opens up new areas of simple, minimum component interface with both PMOS and CMOS. The open drain PMOS depicted in Figure 15A shows the technique for employing the 5707 or 5712 with only an appropriate pulldown resistor for each MOS output and a series zener to obtain the +5 volt V_{CC} for the logic gate of the driver. The use of a high voltage input diode (collector/base diode) results in a vertical PNP to the substrate and accounts for the dramatic reduction in logic "0" input current. The input is actually a PNP, and the current ($\leq$ 100μA) being sunk in the

driving device or pulldown resistor is the base current of the PNP shown in Figure 15B. The main element of the logic "0" input current (I_E) is shunted to the substrate (ground) as PNP collector current (I_C); another Sprague first in peripheral/power drivers.

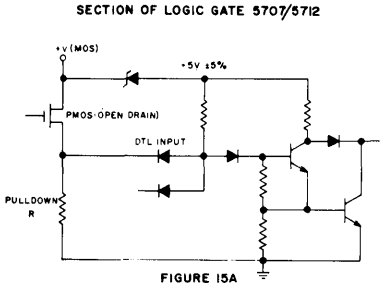


FIGURE 15A

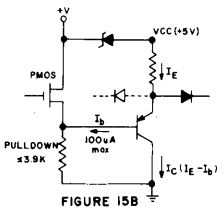


FIGURE 15B

Pulldown resistors for this type of application fall within reasonable values, although the same is not true of standard TTL inputs with a 1.6mA input current limit. It would be necessary to use a 250 ohm pulldown with standard TTL (0.4 ÷ 1.6mA = 250 ohm), but the 100μA of the Sprague DTL units will allow up to 4K ohms. Even if this need for a 250 ohm pulldown is overcome the maximum input voltage of 5.5 volts for TTL precludes its easy use with most MOS circuitry operating above 5 volts.

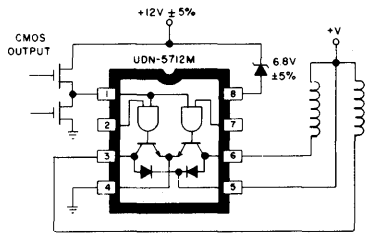
Additionally most PMOS circuits have a limited output source capability, and a problem presents itself when the PMOS output is turned on. To properly operate a TTL or DTL IC its input must swing higher than 2.4V; a serious difficulty if a 250 ohm pulldown resistor is used (TTL). However, when using the Sprague DTL types it is only necessary to source 615μA (2.4V ÷ 3900 ohms), and should present little, if any, problem for most PMOS interface.

With the exception of the probable need for a pulldown resistor (may not be required with some depletion load PMOS) the same basic considerations apply to CMOS interface.

CMOS Interface - DTL Types

The much lower input logic "0" current and the 30 volt minimum input breakdown afforded by a collector/base diode are a great asset for those wishing to interface from CMOS to a high current load. In Figure 16 a typical CMOS to relay/solenoid scheme employs the UDN 5712 dual driver; a configuration operating from the CMOS supply of +12 volts. Use of a simple series zener diode of an appropriate current rating will suffice for the V_{CC} line if the +12 volt CMOS supply is adequately regulated.

FIGURE 16



Systems with poor regulation may require a simple regulator such as those shown in Figure 17A (NPN power transistor) or Figure 17B (power Darlington) to obtain the +5 volt V_{CC} line. The choice is largely based upon the input current required for the series pass power device, and the effects upon the zener power rating under minimum load conditions (obvious advantage to Darlington). For CMOS logic systems requiring relatively few peripheral/power drivers for high current interface either of these schemes, or the use of an appropriate three terminal regulator, would simplify system design - particularly if system +5V currents are small and little heat sinking is required for the pass transistor or three terminal regulator.

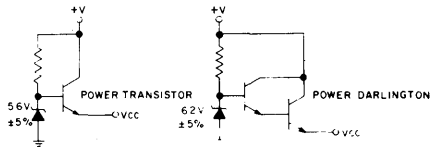


FIGURE 17A FIGURE 17B

Extending Output Sink Current - DTL Types

The high current output NPN of all the Sprague peripheral power drivers will operate beyond the 250mA guaranteed level (UHP 400/500's) or the 300mA rating of the newer DTL types (UDN 3600 and 5700's). The newer units are better suited to use above standard output current ratings; an improvement largely related to improvements in the output NPN design. Extending the 300mA capability requires additional base current, and consequently the V_{CC} line must be raised above the nominal 5 volts. In Figure 18A the increased output current (minimum) is plotted against the increased V_{CC}; alternatively it can be viewed as a maximum increase in V_{CC} to obtain additional sink current capability.

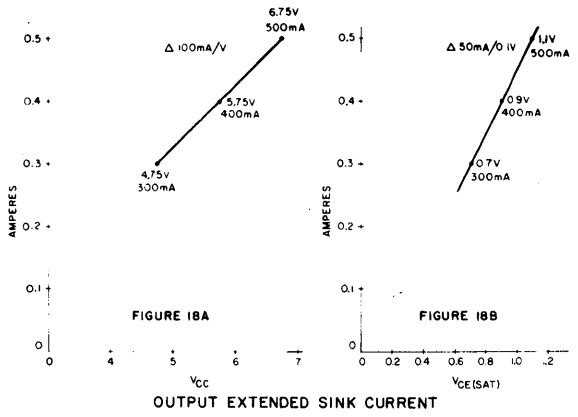
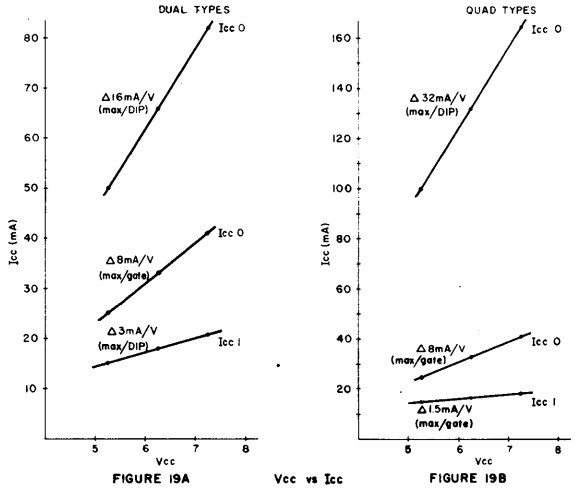


Figure 18B indicates the maximum $V_{CE(sat)}$ changes as the current is increased from 300mA to 500mA. This information is largely necessary for calculations of package power dissipation if the system duty cycle presents questions of average power.

Figure 19A and 19B present information relating to the increased logic gate supply currents as a function of the increased V_{CC} required for extending output currents. Figure 19A is for the dual types, and presents the information on both a per package change (Δ16mA/V) and a per gate maximum. The maximum increase is presented for both I_{CC} "1" and I_{CC} "0", and the most significant is the I_{CC} "0" which is indicated in each figure on both a per package and per gate basis. The change in I_{CC} "1" is small ($\leq$ 1.5mA/gate) and to prevent confusion in Figure 19B is shown per package for the dual only and per gate only in the quad graph. Increase in I_{CC} "1" in quad types is 4X the per gate maximum (4 X 1.5mA, or 6mA/package per volt).



Vcc vs Icc

Package Dissipation - Mini DIPs

The Øja rating of the Sprague dual types in a mini DIP is +80°C/W, while the National specifications list all dual minis at +110°C/W. Similar ratings apply to other suppliers, and the advantage is to any type using copper alloy lead frames.

Per the discussion included in package availability the allowable average power is definitely in favor of the Sprague DIPs; the allowable power at +70°C is 1.0W for the Sprague units. Contrast this 1.0W maximum from Figure 20 (+80°C/W curve intersects 1.0W at +70°C) with a 727mW limit calculated from the +110°C/W ratings listed by National (both limits are derived using max junction limit of +150°C and manufacturers Øja rating). A worst case analysis will reveal that the electrical specifications of competitive parts produce power levels for 100% duty cycle applications that may result in junction temperatures above the +150°C level if both outputs are simultaneously and continuously sinking 300mA with V_{CC} at 5.25V. The LM 3611 max power (per spec) could reach 782mW (LM 3612, etc, slightly higher) and the 75451 maximum is 761mW (75452, etc slightly higher).

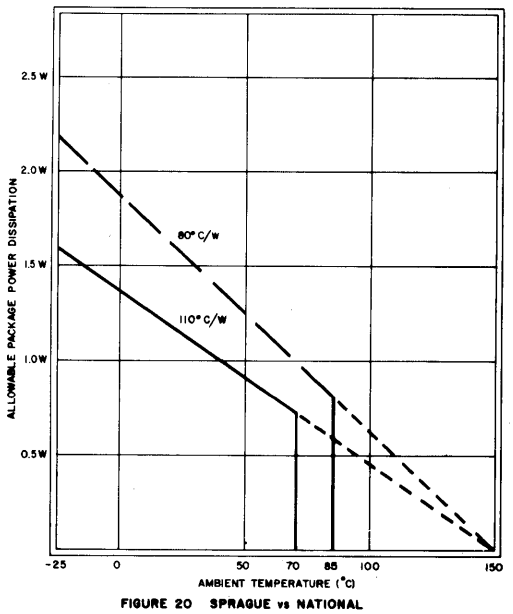


FIGURE 20 SPRAGUE vs NATIONAL

The Sprague UDN 3611 or 5711 will have a worst case power (same conditions of V_{CC} and I_C) of only 677mW, and is also manufactured in a DIP package capable of sustaining 1.0W rather than the

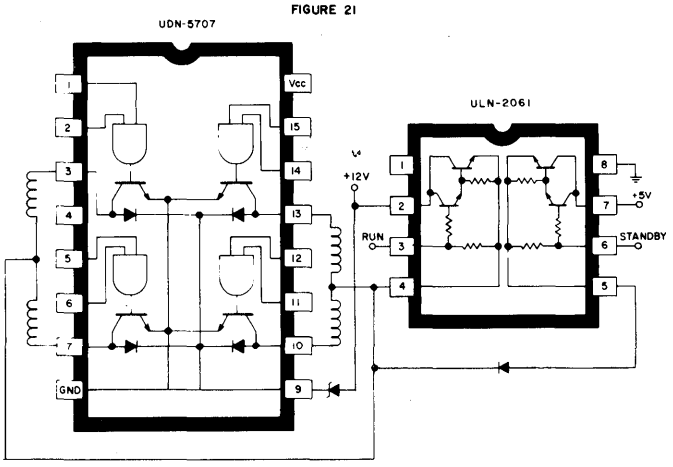
727mW limit. A clear advantage for lower junction temperatures, less heat, and/or greater output capability in a system. Maximum junction temperatures obtained would be: LM 3611 with 782mW $\simeq$ 156°C; 75451 with 761mW $\simeq$ 153°C; and the Sprague UDN 3611/5711 with 677mW $\simeq$ 124°C.

Stepping Motor Applications

Combining the use of a peripheral/power driver and a dual Darlington switch (type ULN 2061) provides a capability of driving a 4-phase bifilar stepping motor. Motors designed to operate with voltages and currents compatible with these ICs may be driven with a minimum of components. In Figure 21 the signals from appropriate logic/sequencing circuitry operate a Sprague UHP 407 or UDN 5707 (may also be done with two UDN 5712s) for switching the motor coils selected. The transient suppression diodes are utilized here as per solenoid/relay applications, although here they are connected through a zener diode to the supply voltage. The zener will improve the speed of the switching, but should be chosen such that the maximum voltage across the output (+V added to V_Z) is below the device breakdown. Permitting voltage excursions of this sort produces improved high speed motor operation, but must be clamped to a safe value.

STEPPING MOTOR DRIVE

FIGURE 21



Applications requiring a holding or detent current may employ a dual Darlington (ULN 2061) as shown. One Darlington switch is used for the RUN mode, while the second (lower) half of the 2061 is used to provide a lower, holding current to maintain the position of the motor. Use of two supplies is shown, and diode D1 decouples the power supplies and prevents unwanted reverse bias from reaching the STANDBY Darlington. Similar schemes may be employed to obtain bipolar drive schemes for

high speed stepping motor applications. Current is sourced from the positive supply (a PNP switched by a gate capable of sustaining the supply voltage is one example), while an appropriate peripheral/power driver is used to sink coil current per Figure 21.

Control 100 Watts With an IC

The quad peripheral power drivers are capable of controlling (switching) loads that total 100 watts per package (50 watts for duals). Load currents beyond the standard 250mA level (UHP 500s) or 300mA (UDN 5700A quads) would result in the capability of switching loads in excess of the 100 watt capability with standard specifications.

The UHP 500 series has a 100 volt/250mA capability for each of four outputs: 100V X .25A X 4 outputs = 100 watts of control.

The UDN quads offer an 80 volt/300mA combination for each of four outputs: 80 volts X .3A X 4 outputs = 96 watts.

All of these high current/high voltage peripheral/power drivers offer simple, inexpensive interface solutions to some tough load requirements. Those applications beyond the output voltage and/or current handling limits of lower current, TTL type devices are quite easily handled with these units. Additionally, the newer DTL types provide solutions to interface problems associated with PMOS or CMOS that are quite often impossible with TTL type units. The "giant step backward" actually results in a product "stride forward" with the newer DTL peripheral/power drivers.

4. MASTERSLICE LSI — THE COST EFFECTIVE ALTERNATIVE

DR. CHARLES A. ALLEN
Vice President of Engineering
International Microcircuits, Inc.
Santa Clara, CA 95051

I. What is Masterslice LSI?

Masterslice LSI is a cost-effective approach to the design and manufacture of custom integrated circuits. A Masterslice circuit is a predesigned and preprocessed array of logic elements, ready to be customized for each special requirement. Each Masterslice circuit also contains bonding pads, input and output buffering circuits, and cross-over elements that may be interconnected in a vast variety of patterns to form any sort of complex or specialized logic function. The physical arrangement of these elements is the same for each application. Only the interconnect wiring pattern is unique for each new design.

The use of Masterslice LSI is straight-forward. Even with many of the design trade-offs left in the hands of the system designer, the procedure is not complex and can usually be learned in a few days by an experienced circuit or logic designer. The basic steps are these:

1. After the system is specified, it is partitioned appropriately and the desired circuit function is defined.

2. The detailed logic design required to implement the function is fully designed. Breadboarding is often valuable at this point as a functional check.

3. The detailed logic design is converted to a complete circuit design showing every transistor and every electrical connection required. A count of Input/Output lines and logic elements will indicate which of the available Masterslice circuits can accommodate the required function.

4. Since the elements on the Masterslice circuit are already physically placed, the remaining design effort required is the physical placement of all the interconnect wiring. This is done by sketching the desired wires on a large drawing of the Masterslice chip. This process is similar to the layout of a complex printed-circuit board involving an element of trial-and-error and requiring both some intuitive feel for the grouping of the various parts of the circuit and a careful attention to detail to assure that every wire is correctly shown.

5. This interconnect sketch is used to produce precision artwork which will determine where the aluminum interconnect wiring on the Masterslice wafer is to be located. This artwork is also used to check against the logic design performed in Step 2, and the detailed circuit design performed in Step 3 to assure that no errors have been introduced in the process of translating the logic design into chip artwork.

6. A photomask produced from this artwork is used to perform the aluminum etching step on a preprocessed Masterslice wafer.

7. The resulting chips are inspected, diced, packaged, tested, and shipped to the customer.

The Masterslice approach to custom LSI has several obvious advantages over the conventional custom-design process.

1. The chip design cycle is very short since most of the engineering has been done before the custom design starts. Only one of the six or seven photomasks must be customized for each new custom circuit, and even that mask has over 80% of the geometries already specified. Only the non-critical geometries of the interconnect wiring need to be added to complete the chip design. Typical design cycle from logic drawing to finished artwork is three-to-four weeks rather than three-to-four months as in a conventional design.

2. The engineering costs associated with chip layout are correspondingly reduced since these costs are almost entirely manpower and overhead. Typical layout and artwork charges for a Masterslice chip are $500 to $2,000 rather than the $3,000 to $15,000 required for a conventional design.

3. Tooling costs are very low in the Masterslice approach since only one new mask needs to be manufactured for each new custom circuit in place of the usual six or seven masks. The rest of the tooling is common across a large number of Masterslice users, so that the costs can be amortized widely.

4. Prototype manufacturing schedules are very short since 80% of wafer processing has already been accomplished. Only the metal interconnect etching remains to be done. Prototype schedules of two-to-four weeks are easily attainable in place of two-to-three months with conventional custom designs.

5. Because the tooling costs are low and the turnaround time is short, it is viable to institute changes and updates to the circuit as market conditions change.

6. Production prices after the prototype stage are very low for low or moderate volumes since much of the processing can take advantage of the large volume associated with all the custom parts being produced on a given Masterslice rather than the very small volumes associated with a single custom circuit. A typical wafer lot will yield 10,000 to 40,000 finished circuits, which is frequently more than the total annual production of a single custom circuit. With a conventional approach, all the wafers in a lot must be personalized for a single-part type, and thus the manufacturer is faced with either feast or famine, depending on final test yield. The Masterslice approach allows the wafer lot to be split at the metal etch step

into sublots as small as one wafer, and thus production runs as small as 100 pieces are feasible.

7. Rapid changes in production volume can easily be accommodated using the Masterslice approach. With a total production cycle of four-to-eight weeks including overseas assembly and final testing, the production schedule can be increased or decreased on a monthly basis with little risk to either the supplier or the customer. Even if changing market conditions require that the program be canceled altogether, the pipeline is so short that usually only a one-month notice is required. The uncommitted wafers can readily be diverted to other custom circuits at almost no penalty. In a usual custom design, with a production pipeline of four-to-five months, the impact of changes in either demand or yield are much more severe and the cost associated with cancellation can be very substantial.

The advantages of Masterslice LSI over standard SSI and MSI design are many and well known:

1. Size. By reducing five-to-forty packages to a single package, the overall system can usually be made much smaller and lighter, often opening up entirely new markets.

2. Reliability. Integrated circuit reliability is usually related to the number of connections to the outside world rather than to the number of switching elements. Equally important, with a single package it is frequently attractive to spend a few more pennies per package for greater integrity or for active burn-in to further improve reliability.

3. Market Monopoly. A custom circuit which is available to you but not to your competition allows you to sell your creativity at a substantial profit. A design using standard parts, on the other hand, is an open invitation to any and all competitors to capitalize on your creativity.

4. Dependable Parts Supply. Standard parts appear and are discontinued or "unavailable" for reasons over which a single customer has no control or foreknowledge. With a custom part, however, production and delivery schedules can be directly monitored and controlled, and thus there are fewer surprises in your production schedule.

5. Cost. The major advantage of LSI is in reducing total system cost by reducing dramatically the cost of assembly, test, rework, printed-circuit boards, system-interconnect sockets and cables, power supplies, cooling elements, frames and covers, and all the miscellaneous costs of supporting the numerous circuit packages in a standard SSI-MSI design. Since these support costs increase year by year as labor and material costs increase while LSI costs per circuit function decrease with improvements in processing yield, the cost advantage of LSI has become more pronounced each year and this trend can be expected to continue for many years to come.

The advantages of Masterslice LSI over conventional custom designs are less well known:

1. Schedules. A Masterslice design cycle from conception to volume production is typically three-to-four months rather than 12-to-18 months with a fully custom design.

2. Engineering Costs. The nonrecurring costs associated with a Masterslice design typically amount to $4,000 to $6,000 rather than $50,000 to $100,000 for a fully custom design.

3. Availability. It is feasible to develop a Masterslice chip for production runs as low as 100 parts, while it is rarely possible to find a fully custom vendor willing to tie-up scarce engineering talent without guarantees of over 100,000 parts per year.

4. Ease of Change. Since the tooling investment and work-in-process are so low with a Masterslice design, it is possible to modify either the design or the production schedule to match a changing market. Contracts for fully custom parts, on the other hand, must be much more rigid to protect the supplier from enormous scrap charges.

5. Continuity of Supply. All Masterslices being presently offered are designed around standard wafer and assembly technology, with the expectation that a single design can be fabricated at any one of a number of semiconductor or packaging facilities. Thus, the risk of "losing the recipe" is minimized. In addition, since preprocessed wafers are stockpiled ready for personalization, short-term fluctuations in either demand or yield can be quickly compensated for with almost no impact to the customer.

6. Ease of Communications. The system designer's role in a Masterslice design is very similar to his accustomed role in designing and testing printed-circuit boards and sub-assemblies, so that there are very few new skills to learn and no mysteries that must be accepted on faith. With the design kits and manuals provided, an experienced designer can become conversant with the Masterslice details in a few days. And he can use his unique perspective on the market and system environment to optimize the LSI circuit function. With a fully custom design, the costs are so large, the schedules so long, and the risks so great that they normally involve numerous layers of purchasing, marketing, legal, production, quality, and shipping departments until the system designer is hopelessly lost in a maze of bureaucracy.

7. Cost. All costs in LSI production are strongly dependent on production volume. The Masterslice approach allows all customers to benefit from their combined production volume, whereas many of the costs of a fully custom design must be amortized over that single customer's production. Thus, the real costs of production are lower for Masterslice designs at any production volume with this difference being most pronounced for low-volume customers.

A comparison of nonrecurring engineering costs for a typical subsystem is shown in Fig. 1. For both the standard SSI/MSI design and the Masterslice design, the dominant costs are the in-house labor and overhead for design, drafting, printed-circuit layout and prototypes, test fixtures, and documentation. In addition, the Masterslice approach requires the expenditure of approximately $5,000 for chip layout and prototypes. For the full custom circuit, however, a huge additional expenditure of $50,000 is required for chip development plus some additional costs associated with the longer development cycle and the greater difficulty in monitoring vendor progress.

A comparison of manufacturing costs for these same three systems is shown in Fig. 2, with the nonrecurring costs amortized over a production volume of 10,000 units. Note that the fully custom and the Masterslice design are nearly equal in reproduction costs, but the very large development charge of the fully custom chip is an overwhelming disadvantage. The SSI/MSI design yields the lowest parts costs, but the assembly, test and rework, and system-support hardware result in an unattractive total cost. As labor costs escalate and as environmental constraints on printed circuits and plastics manufacturers become more severe, we can expect this system overhead to continue to grow, while yield improvements will allow even more circuitry to be compressed into a single LSI chip, with even lower costs per gate.

II. The MasterMOS-M Chip

A scale drawing of one of three CMOS Masterslice circuits developed by International Microcircuits, Inc., called the MasterMOS-M chip, is shown in Fig. 3. The entire central region of the chip consists of 254 P-channel and 254 N-channel MOS transistors and 322 interconnection underpasses arranged in a regular array of unit cells.

An enlarged drawing of one of the cells is shown in Fig. 4. The rectangles identified by a diagonal slask mark are contact openings through the protective silicon oxide to the bare silicon below. All circuit connections to source, drain, Vdd, Vss, or to underpasses are made by running a metal line to or across the appropriate contact opening.

The dashed lines on the drawing indicate thin-oxide regions. Note that all contact openings are enclosed by a thin-oxide region. This improves contact hole uniformity and alignment tolerance and reduces unwanted pinhole contacts, and thus improves circuit yield. Note also several large areas of thin oxide within the cell. These areas will form the gate regions for all the N-type and P-type transistors. The P side can be identified by the narrower channel stop diffusion which is the same width as the contact hole. The N side has a channel stop diffusion which is 50% wider than the contact hole. All P-channel transistors are located on the P side, while all N-channel transistors are located on the N side. Note that alternate rows of cells are mirror-imaged and thus the relative location of P side and N side varies from row to row.

Each cell contains two groups of complementary transistors. The first group contains two N-channel devices and two P-channel devices, while the second group contains three transistors of each type. Actual

transistor action takes place in the thin-oxide regions (dashed lines) of the cell. The source and drain-diffusion areas are the small irregularly-shaped areas each containing a single contact hole. This contact opening is used to perform the desired circuit interconnection. There are fourteen separate source/drain areas in each cell; seven on the P side and seven on the N side. Note that six of the fourteen source/drain areas are common to two transistors within the cell. This greatly simplifies the wiring in performing combinational logic.

The N+ bed surrounding the P-channel devices forms a channel stop for those devices, while the P+ bed surrounding the N-channel devices forms a channel stop for those devices.

The region between the N-type and P-type transistors contain eight contact openings to permit circuit connection to Vdd and Vss. The four contacts in the N-side channel stop are Vss contacts, and the four contacts in the P-side channel stop are Vdd contacts. Because of the very low quiescent current in CMOS logic, no other wiring is necessary for Vdd and Vss in the cell area.

The narrow horizontal region between the N+ and the P+ channel stops is the N- substrate. This region serves to raise the breakdown voltage between the N+ and P+ regions to permit circuit operation with a higher supply voltage.

The long vertical bars of N+ and P+ serve as underpasses or circuit cross-overs for signal interconnect wiring. Contacts are provided at both ends on all underpasses and at an intermediate point on four of the seven underpasses. Up to seven horizontal metal lines can be run between the end contacts. See the following section for details on allowable interconnect patterns.

The periphery of the chip contains:

(a) 20 medium-power buffers;

(b) 4 high-current N-channel transistors;

(c) 2 high-current P-channel transistors;

(d) 1 high-impedance N-channel transistor;

(e) 1 high-impedance P-channel transistor;

(f) 28 bonding pads.

The medium-power buffers are suitable for driving low-power TTL or standard CMOS circuitry. The high-current N-channel devices have adequate drive capability for one standard TTL load. The high-current P-channel devices are useful for active pull-up on low-impedance loads. The high-impedance devices are useful for special applications where either the input or output current must be very low, but not zero, during a standby condition, e.g., low-power pull-up. The 28 bonding pads may be assigned arbitrarily by the system designer as Input, Output, Voltage, Intermediate Test Points, or left blank. Input protection is provided by the underpasses available near each pad.

Fig. 5 shows two efficient realizations of a simple inverter circuit. Note that Vss is connected to the source of the N-channel device through the channel-stop contact and that Vdd is similarly connected. A simple two-input NOR is illustrated, showing two N-channel devices in parallel and two P-channel devices in series. A two-input NAND circuit would be formed by wiring the N-channel devices in series and the P-channel devices in parallel.

Fig. 6 shows a D-type Master-Slave flip-flop without Set or Reset. The dashed line shows the interconnect necessary for binary operation. An important feature of this circuit is the large amount of interconnect area needed when transmission gates and feedback are used in a cell.

III. MasterMOS Applications.

Fig. 7 shows a simple consumer product implemented on the small MasterMOS-S chip. This circuit displaces five packages of standard CMOS plus twelve discrete components for a size reduction of eight-to-one.

Fig. 8 shows a multiplex circuit implemented on the medium-sized MasterMOS-M chip which displaces ten standard CMOS chips in a hybrid application. Wire bonds and package real estate are both reduced by a factor of ten-to-one.

Fig. 9 shows a communication circuit implemented on the large-sized MasterMOS-L chip which displaces an entire board containing 32 packages of standard CMOS logic plus 18 discrete components.

These three typical applications illustrate the broad flexibility of the Masterslice approach to custom LSI.

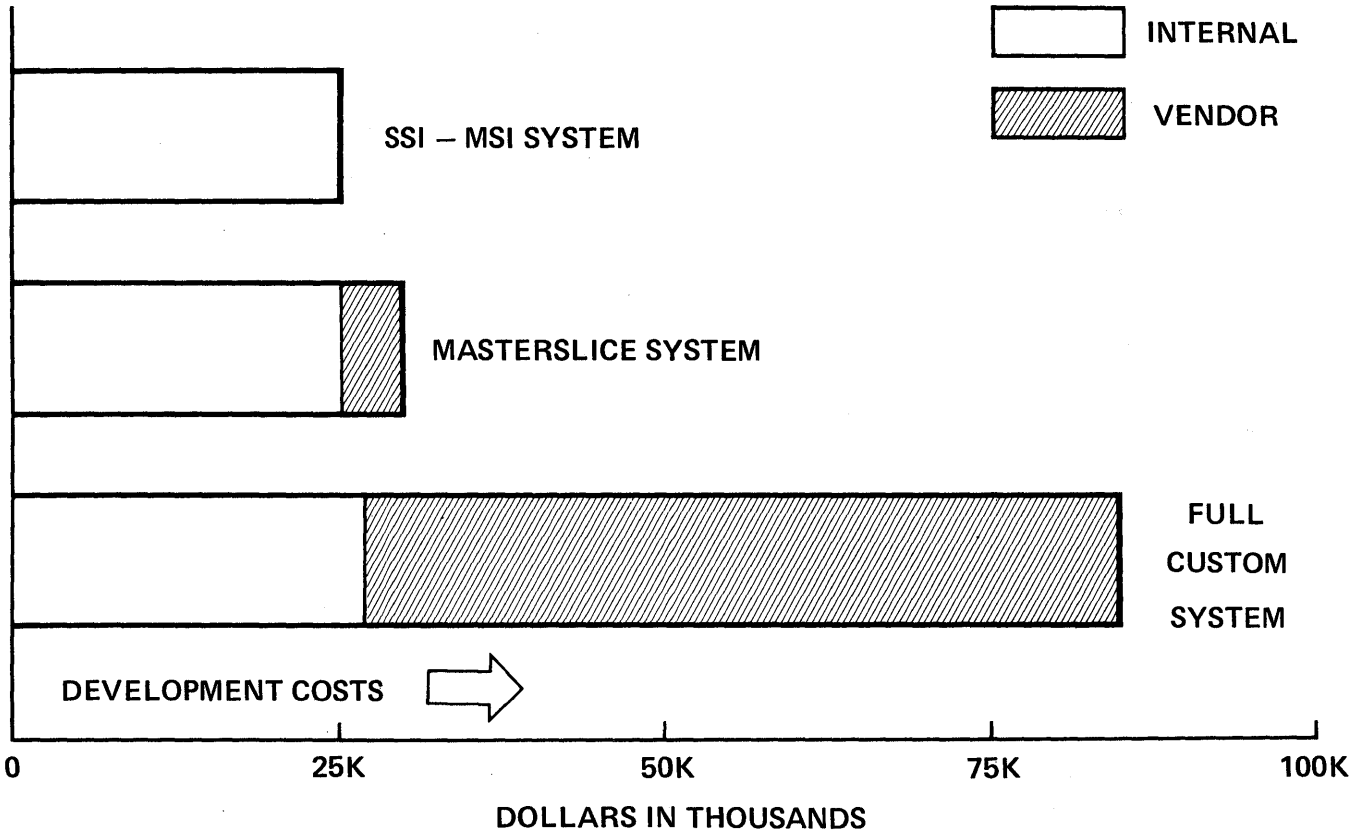


Figure 1. TYPICAL ENGINEERING COSTS

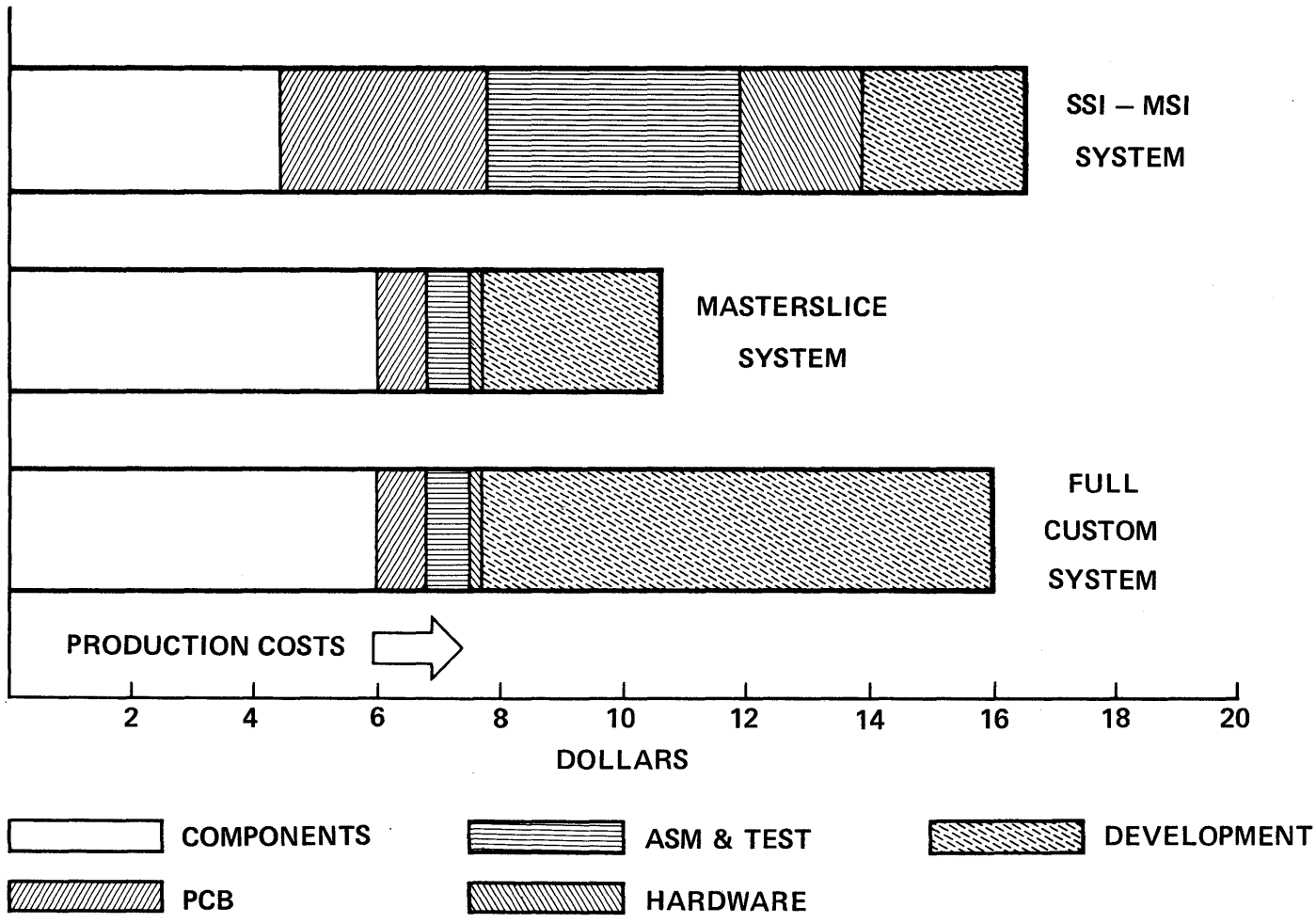


Figure 2. TYPICAL PRODUCTION COSTS

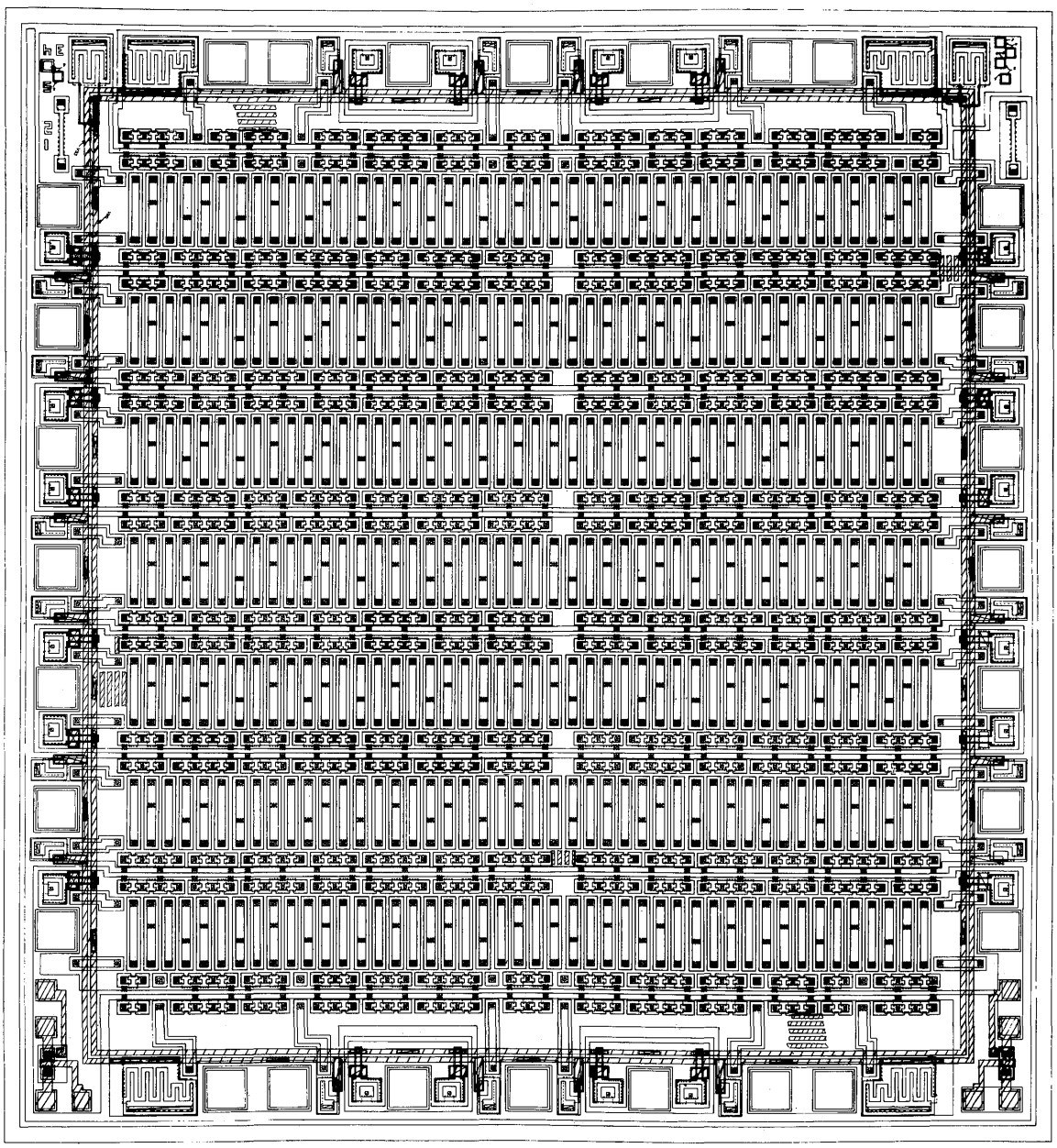


Figure 3. MASTERMOS CHIP – 60 times actual size

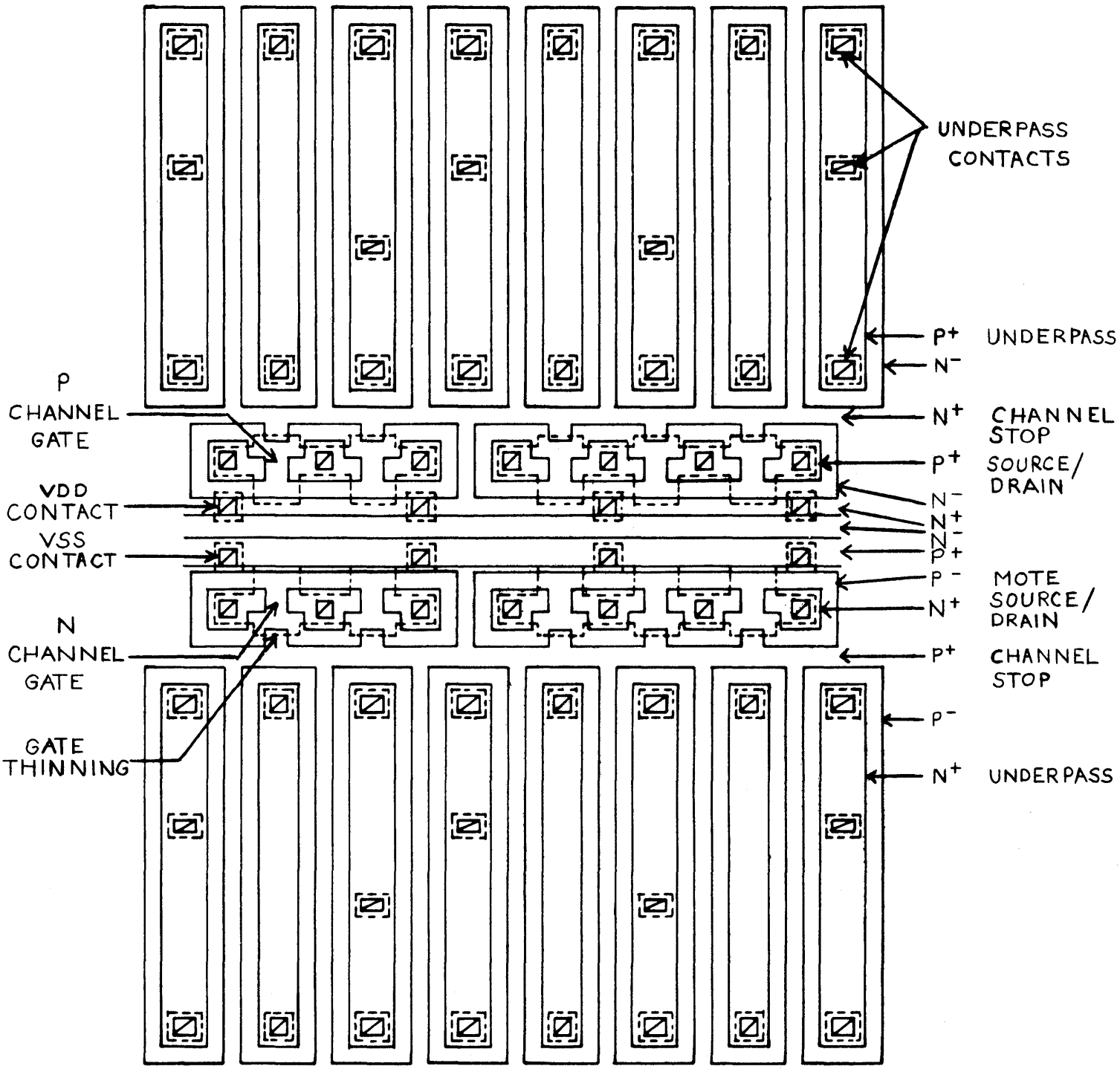
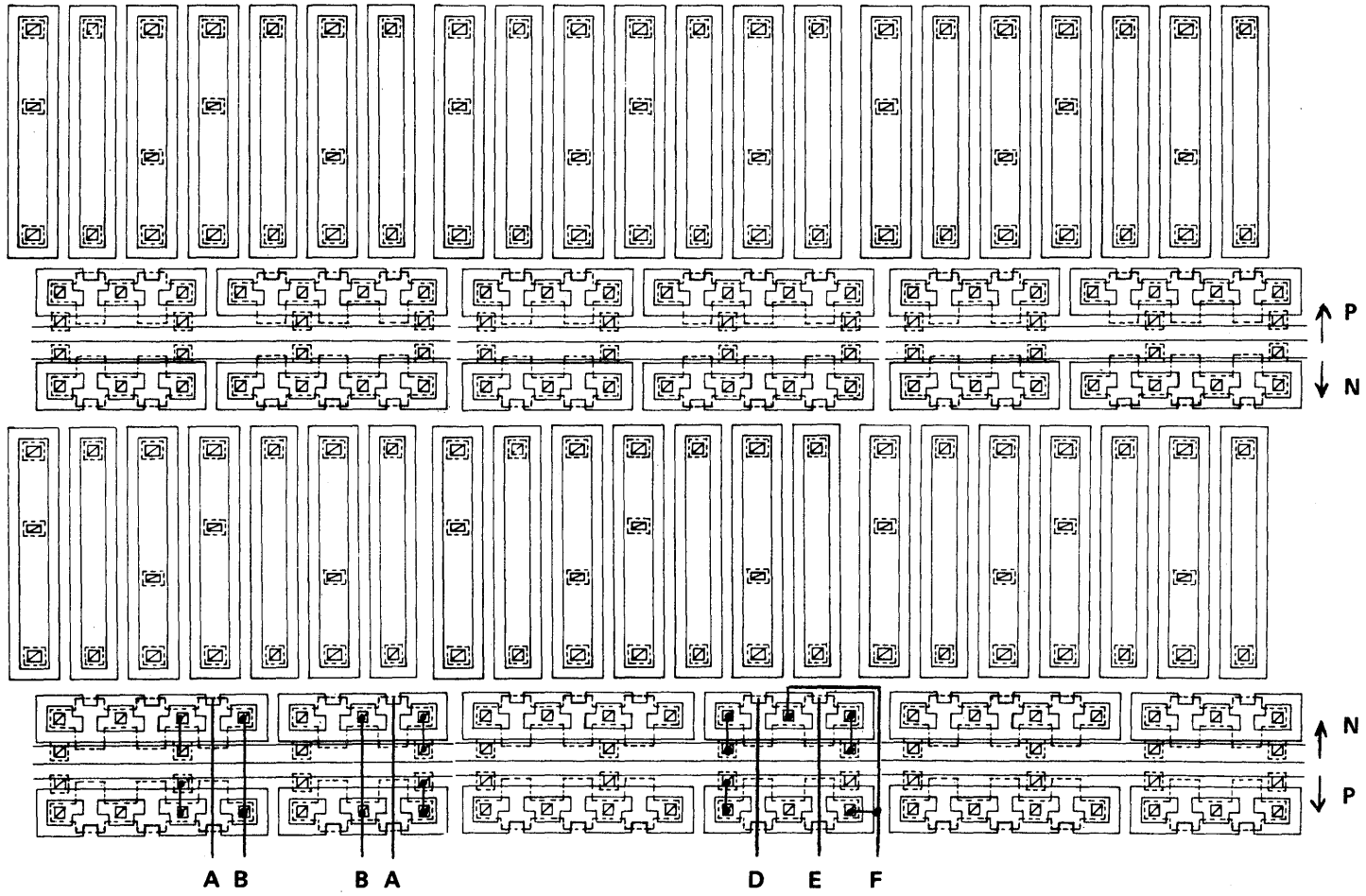
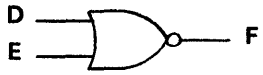


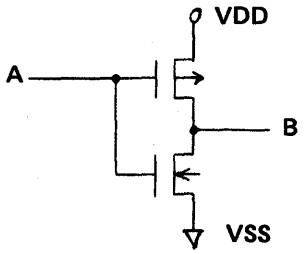
Figure 4. MASTERMOS CELL – 400 times actual size



INVERTER

2 INPUT NOR





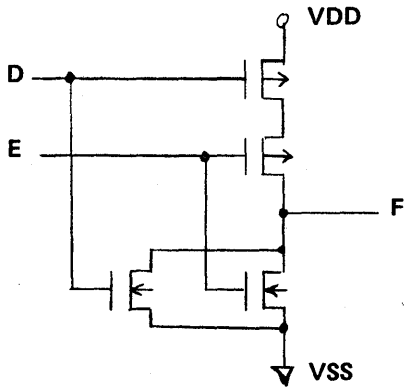
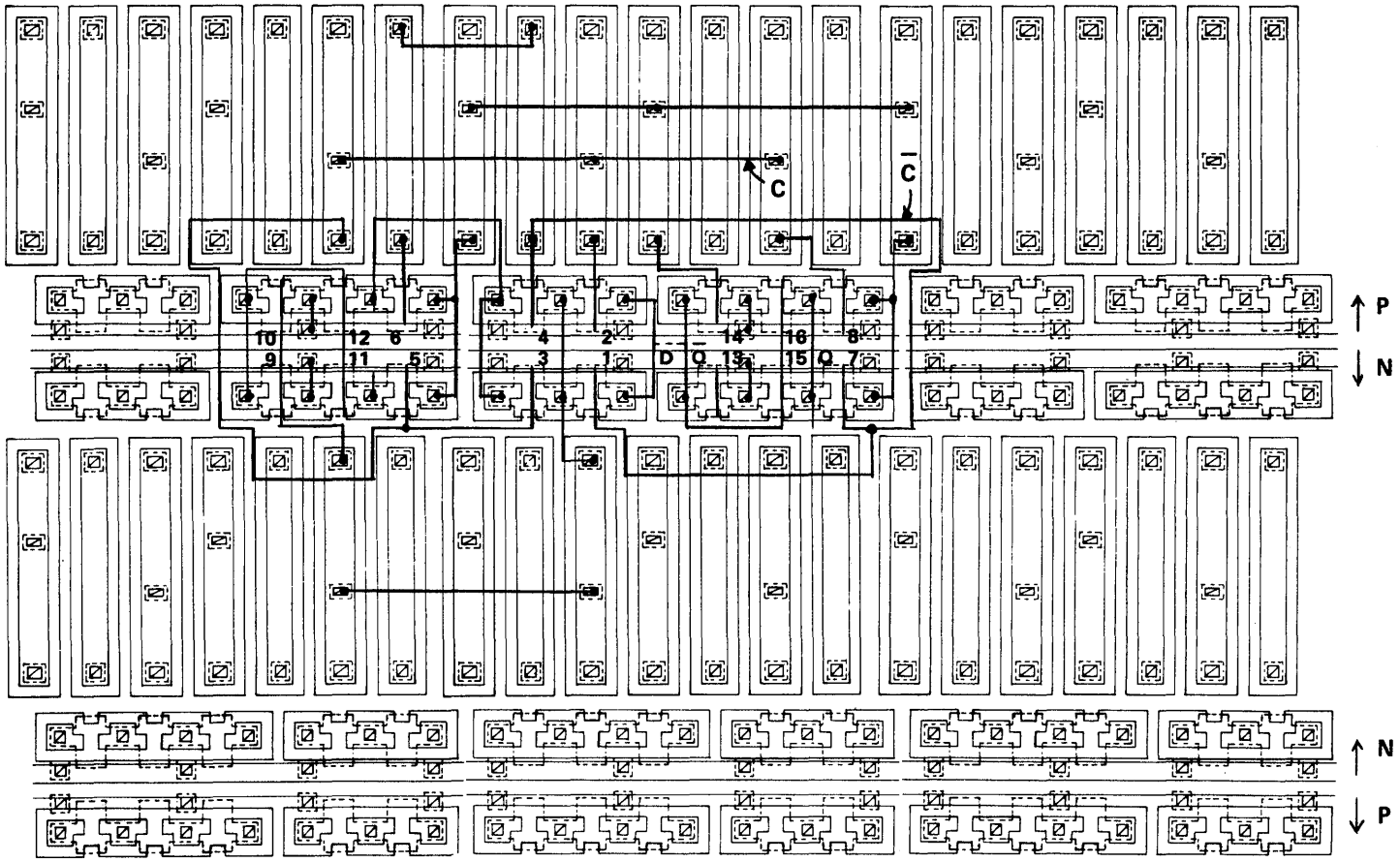


Figure 5.



"D" TYPE FLIPFLOP
WITHOUT RESET/SET

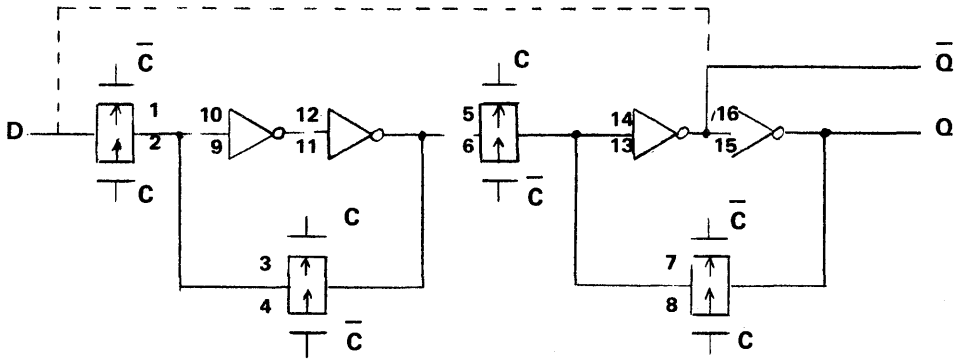


Figure 6.

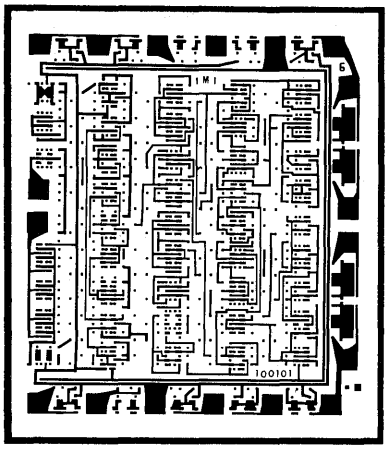


Figure 7. CONSUMER PRODUCT

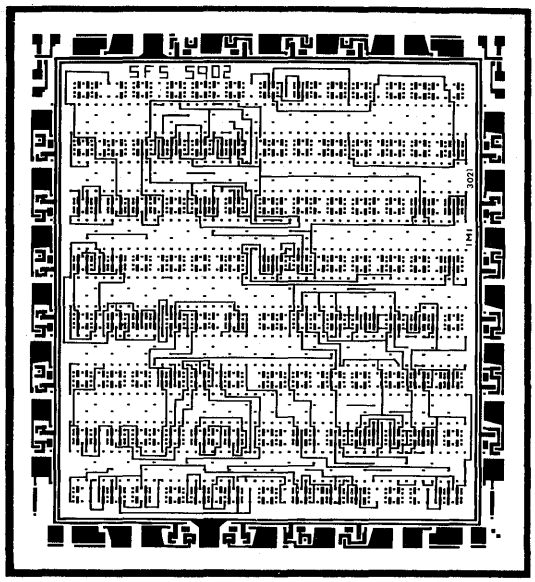


Figure 8. MULTIPLEX CIRCUIT

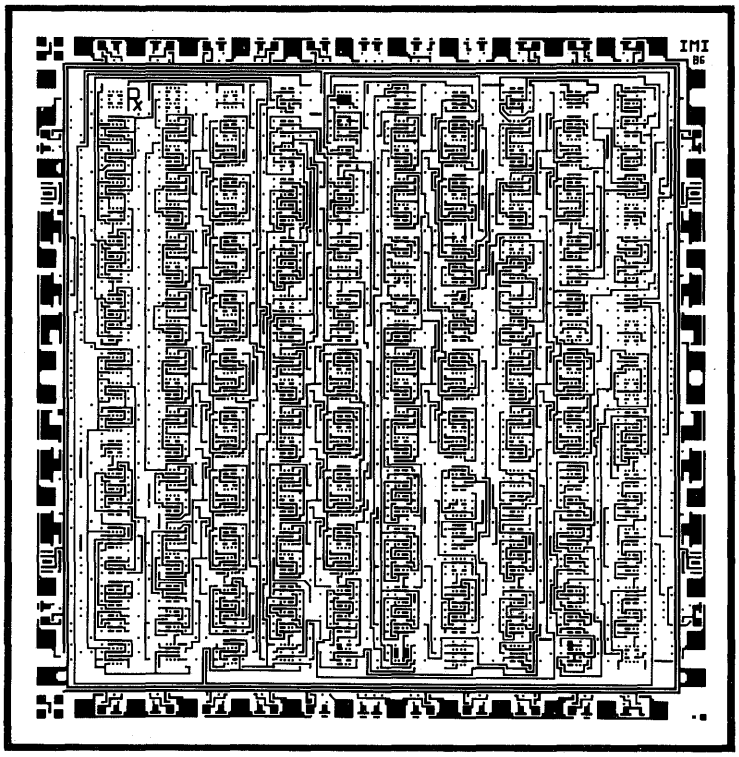


Figure 9. COMMUNICATIONS CIRCUIT

5. STRUCTURE AND APPLICATIONS OF FPLAs

NAPOLEONE CAVLAN
Signetics
Manager, Advanced Products Marketing
Sunnyvale, California

Since the practical introduction of microprogramming in the last decade or so, microcode has displaced random logic in step with the growing availability of low cost, user Programmable Read-Only Memories (PROMs). However, even with PROMs designers soon realized that their rigid addressing structure made them unsuitable in a wide variety of applications which could greatly benefit from a structure logic approach.
Recently, microprocessors have provided a quantum jump in design flexibility in applications requiring about 30 IC packages, and beyond. When fewer packages are required, the inherent speed limitation, software requirements, and support circuitry of microprocessors place them out of range of a broad spectrum of applications. These in general involve algorithms which require a high speed logic decision based on a large number of controlling variables. It is here that we step in the basic domain of Programmable Logic Arrays, encompassing applications in microprogramming, code conversion, random logic, high speed character generators, look-up and decision table, etc.
Although PLAs have been available for several years, their introduction to the market caused hardly a stir. The reason can be primarily attributed to the dampening of user enthusiasm following the realization that in most cases mask charges and weeks of turnaround time imparted by factory programming created as many problems as it solved, and quickly eroded initial usage gains. The inevitable programming changes resulting from either user inexperience with the device or system evolution implied an intolerably long design cycle and, most important, no custom design flexibility and slow recovery from marginally hidden faults detected after system release in the field.
These considerations rendered field programmability a mandatory feature for design economy, and provided impetus to the industry to develop FPLAs which could be easily programmed by the user. With the growing availability of these devices, now for the first time the user can utilize the associative and logic compression properties of FPLAs as cost effective design tools. Moreover, when accompanied by a few storage elements (flip-flops), the FPLA becomes a powerful logic machine of the Mealy or Moore form for the design of finite state sequential controllers for traffic, process, peripheral devices, and other similar applications.
These characteristics, in conjunction with the flexibility afforded by user programmability are the foundation for the emergence of FPLAs as a practical design alternative.

SIGNETICS' FPLAs

The logic structure of Signetics' FPLAs, excluding the internal fusing circuitry is shown in Fig.1. Both devices consist of an upper AND matrix containing 48 product term

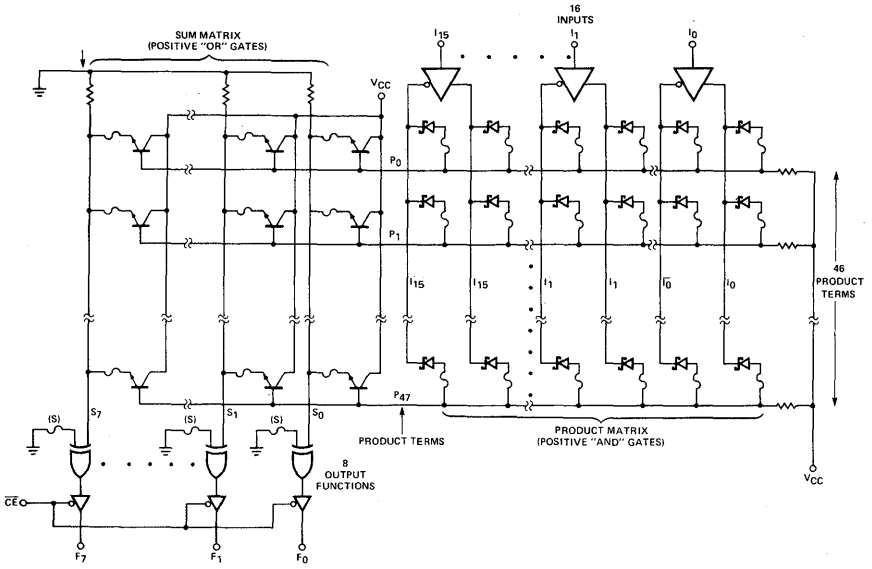


FIG. 1: Logic structure of Signetics' FPLA illustrating AND, OR, and EX-OR arrays. All fusible links are made of Nichrome. Both tri-state and open collector devices are available, with access time of 50 NS maximum (0°-70°)C.

rows (P-terms), and a lower OR matrix containing 8 sum term columns (S-terms), one for each output function. Each P-term in the AND matrix is initially coupled to each of 16 input variables via 2 Schottky diodes for programming the desired input state, and to each S-term in the OR matrix thru an emitter follower with an emitter fuse, for pulling the summing node to a HIGH level when the P-term is activated. Each S-term in turn is coupled to its respective output via an EX-OR gate which has programmable transmission polarity by means of an input to ground thru a fusible link.

In its initial unprogrammed state all Ni-Cr links are intact, such that:

A. *Each P-term contains both True and Complement values of every input I_m. Hence all P-terms are in the "NULL" state (always LOW).*

B. *Each S-term contains all 48 P-terms.*

C. *The polarity of each output is set to active-HIGH (F_p function). Since all P-terms are inactive, all outputs will be at a LOW level when the chip is enabled ($\overline{CE}$=LOW), regardless of input condition.*

Since in a blank device all P-terms are in a logic "NULL" state, unused P-terms require no programming at all, and can be skipped during a programming sequence. Each P-term is programmed with the desired logic state of each input variable by fusing the appropriate link in the pair which couples each P-term to each input variable. If P_n contains I_m, the $\overline{I_m}$ link is fused, and viceversa. If I_m is a Don't Care in P_n, both the I_m and $\overline{I_m}$ links must be fused. If fewer than 16 variables are used, the unused variables represent Don't Care conditions for all used P-terms, and their corresponding I_m

and $\overline{I_m}$ links must be fused.

The response of each output function to programmed P-terms is assigned in the OR matrix. If any product term P_n activates an output function, the link coupling that output function to the P-term(s) must be fused, and viceversa. No programming is required of OR matrix links coupling used or unused P-terms to S-terms servicing any *unused* output functions. Finally, to program an output function true active-LOW when logically selected by any P-term, the corresponding link (S) must be fused.

PROGRAMMING AND EDITING

The FPLA is programmed with the desired Program Table in three successive steps involving the AND matrix, OR matrix, and the transmission polarity of the output EX-OR gates. For this purpose, automatic programming equipment is currently available on the market.

The peripheral fusing internal to the FPLA, and the basic terminal requirements which must be provided to fuse the three sectors of the FPLA are shown in Figs. 2 and 3 respectively. Each P-term 0 thru 47 is individually addressed by applying a binary code

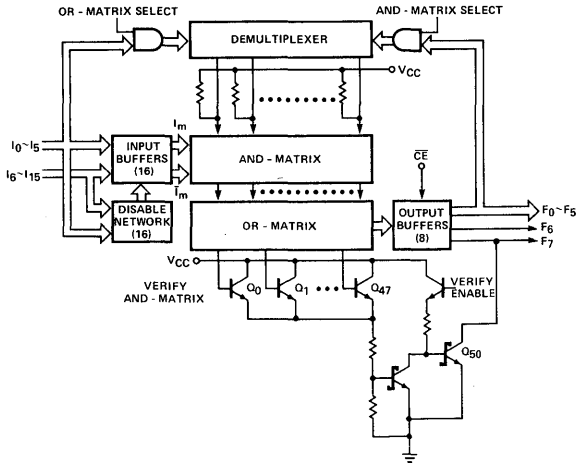


FIG. 2: Functional FPLA blocks activated during ARRAY Program/Verify sequence.

to outputs F_0 thru F_7, while all internal output buffers are disabled via the $\overline{CE}$ input. The address is steered by the AND-matrix select line thru AND gates to the demultiplexer, selecting the P-term to be programmed. All input variables are initially discon-

PROGRAM →	'AND' MATRIX	'OR' MATRIX	OUTPUT ACT LEVEL
V_{CC}	+5.0 V	+8.75 V	LOW
INPUT(S) (Program) I_m	HIGH	ADDRESS P-TERM WITH $I_0 \sim I_5$	HIGH
INPUT(S) (Program) $\overline{I_m}$	LOW		
OTHER INPUTS	+10.0V		
OUTPUT(S) Contains P_n	ADDRESS P-TERM WITH $F_0 \sim F_5$	LOW	HIGH / LOW
OUTPUT(S) Excludes P_n		+10.0 V	LOW (0V, 1MS, +17.0V)
Fuse Enable	+17.0 V	+17.0 V	LOW
$\overline{CE}$	+10 V HIGH 1MS	+10 V HIGH 1MS	HIGH

FIG. 3: Summary of FPLA terminal requirements for programming respective areas in the device.

ADD	P-Term To F_p/F_p^*	YES	Program desired logic function into any unused P-Term. Blow S-Term link(s) coupling P-Term to undesired output functions.
	$I_m/\overline{I_m}$ To P-Term	NO	Delete erroneous P-Term. Add new, corrected P-Term.
DELETE	P-Term From F_p/F_p^*	YES	Blow S-Term link coupling P-Term to F_p/F_p^*.
	$I_m/\overline{I_m}$ From P-Term	YES	Blow both links coupling the input variable to the P-Term.
CHANGE	$F_p \longrightarrow F_p^*$	YES	Blow EX-OR link of output to be inverted.
	$I_m/\overline{I_m} \longrightarrow X$	YES	Delete $\overline{I_m}/I_m$ from P-Term.
	$I_m \longleftrightarrow \overline{I_m}$	NO	Delete erroneous P-Term, and add a new P-Term.
	$F_p^* \longrightarrow F_p$	NO	Use spare active-HIGH output.

FIG. 4: Summary of EDITING features of SIGNETICS' FPLA.

nected from the P-term by applying +10V to all inputs. Links are set to blow one at a time by applying to each variable a TTL logic HIGH or LOW level, depending on whether a HIGH or LOW logic state is contained in the P-term. Actual fusing of the link is accomplished by raising the Fuse Enable input to +17V, and by pulsing the $\overline{CE}$ input from a HIGH level to +10V for about 1 MS. If an input variable is a Don't Care in the P-term, both fuses are successively blown. To Program the OR matrix, inputs I_0 thru I_5 are now used to address each P-term, also by means of a binary input code. The demultiplexer steering network is correspondingly reversed by enabling the OR-matrix select line. All input buffers are simultaneously disabled by raising V_{cc} to +8.5V, effectively disconnecting all inputs from the P-terms. For each P-term, the emitter links coupling the P-term to each S-term are set to be fused one at a time as required. If the P-term appears in the logic expression of an output function, the corresponding S-term link is left intact, *regardless of the chosen output polarity*. Conversely, if the P-term is not contained in the output function, the S-term link is set to be fused by applying +10V to that output. Again, actual fusing of the link occurs by applying +17V to the Fuse Enable and by pulsing the $\overline{CE}$ input from a HIGH level to +10V for 1 MS. Finally, to program any output to active-LOW (F_p^* function, whereby the output switches from HIGH to LOW logic level in response to an activated P-term), the link grounding the input to the EX-OR gate is fused by applying +17V to that output pin for about 1 MS, with no V_{cc} power to the device.

In contrast with PROMs, FPLAs have inherent program editing capabilities allowing the user to incorporate a number of program modifications, and benefit from an unprecedented degree of flexibility and economy comparable to the versatility afforded by erasable PROMs. The type of modifications possible in the program stored in Signetics' FPLAs are tabulated in Fig.4.

VERIFYING THE STORED PROGRAM

Unlike PROMs, verification of an FPLA after programming presents unique difficulties posed by the large number of inputs to be manipulated, and by the associative characteristics of FPLAs. From a black-box viewpoint, the logic function of the FPLA should match entry for entry the original truth-table. This level of verification should be obtained thru a *logic* verification procedure, in which the logic transfer characteristic of the FPLA os exhaustively examined. Logic verification, however, is a useless tool for determining the FPLA stored program. Since a non-ambiguous map of the status of every link in the device is an essential tool required to monitor and manipulate the stored program, the peripheral fusing circuitry in Signetics' FPLAs incorporates additional networks and dedicated paths for an ARRAY VERIFY test sequence to yield the internal map of the FPLA. This procedure (summarized in Fig.5), is similar to a pro-

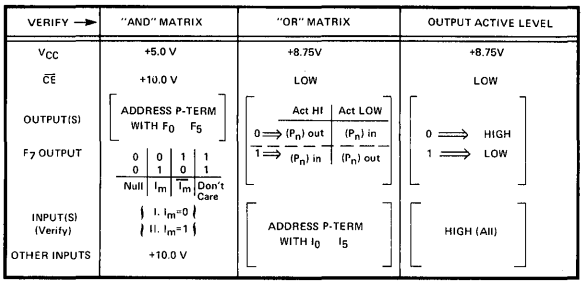


FIG. 5: Summary of FPLA terminal requirements for mapping the status of all internal links. The Output Active Level test must be performed before the OR Matrix test.

gramming sequence, but with Fuse Enable grounded.

LOGIC VERIFY

After an FPLA has been programmed and its contents checked by Array Verify against hard-copy reference of the Program Table, there should be in most cases little reason to suspect that the device will not exhibit the correct logic function in a system environment. However, in some cases device defects, programming equipment problems, user coding inexperience, as well as system logic races and other marginalities, may all contribute in creating a situation in which system failures are nevertheless traced to an FPLA which appears to contain the correct *Program Table*. Also, at the end of the design cycle, it may often be necessary to replace FPLAs with *mask-programmable* PLAs, for cost reduction. Since a PLA does not contain peripheral fusing circuitry, it is not possible to logically address each of its internal links to verify that the PLA contains the same Program Table as the master FPLA. In this case the *only* verification possible is a full logic verify of PLA vs. FPLA function.

Ultimate verification of FPLA logic performance entails an exhaustive check of its logic function to compare the *expected* Truth-Table with the *stored* Truth-Table, obtained by cycling the FPLA inputs thru all 2^{16} combinations with a Minterm generator. This, however, involves dealing with a hardcopy reference of a table containing about 64 thousand input entries, which is a totally impractical task in view of what may be required to generate and store such table. A more feasible alternative consists of constructing a "hardwired" Logic Verify system which may be conveniently incorporated within the FPLA Programming system. The Programmer would then function as an FPLA emulator with the ability to produce and display the full Truth-Table of the FPLA, viewed just as a logic box.

The block diagram of a logic subsystem which executes a suitable algorithm, outlining basic hardware, controls, and data paths is shown in Fig.6. In essence the Logic Verify system must be able to compare the actual FPLA logic output with that computed on-the-fly by composite overlay and manipulation of the Output Table stored in the Programmer, as activated by all concurrent and multiple address selections for each state of the input minterm generator. The algorithm manipulates Program Table data stored in Main Memory and Active Level Register, in the format tabulated in Fig.7.

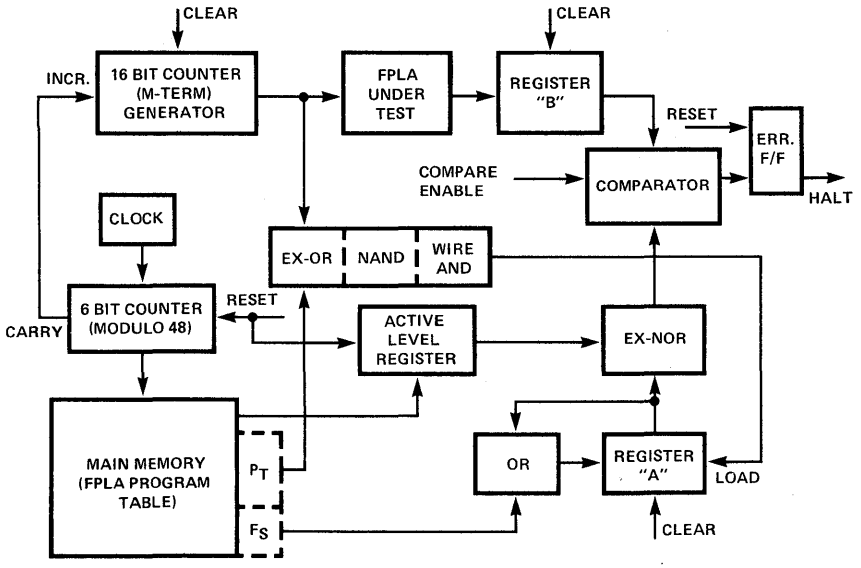


FIG. 6: Block Diagram of Logic Verify system

Before loading the Program Table, M/M and the ALR are reset to "0", to clear all previously stored fusing commands. A binary counter, conditionally incremented, functions as minterm (M_n) generator. The FPLA output for each M_n input is stored in Register B. All 48 P-terms are fetched one at a time from the Program Table in M/M, and examined to determine whether they *logically contain* each M_n. The criteria which logically include or exclude M_n from a P-term are tabulated in Fig.8 for all general programmed states.

	Address	Data									
	P-term #	P-term Field					F-Set Field				
		I_{15}	I_{14}	.....	I_0		F_7	F_6	.....	F_0	
Stored Format	Sequential	0	1	1	0	.....	1	1	0	1	.....	0
Typical Entry	27	H		L	.....	—	A	•	.....		A

a. M/M binary format and typical entry.

	F_7	F_6	.....	F_0
Stored Format	0	1	.....	0
Typical Entry	H	L	.....	H

b. ALR binary format and typical entry.

FIG. 7: Binary assignment of FPLA Program Table stored in Main Memory, and Active Level Register.

	I				II					III				
M_n	H	H	L	L	H	H	H	L	L	H	H	L	L	L
P-term	H	–	L	–	L	H	–	L	–	H	–	L	–	H
	M_n contained in P-term				M_n not contained in P-term									

FIG. 8: Criteria for the logical inclusion/exclusion of a minterm in a P-term. (H↔L) preclude logical inclusion.

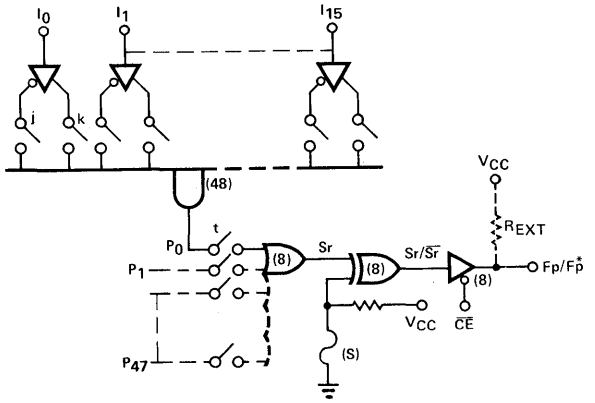
If the test fails, a new P-term is fetched and the test repeated until all 48 P-terms have been examined. On the other hand, if the test indicates that M_n is contained in the P-term, the F-set field associated with the addressed P-term is *overlaid* in Register A, while the M/M address of the P-term is stored in a stack containing the *concurrent* P-term list, and a presence flag set to indicate that the P-term address is a valid member of the list. Testing continues until all 48 P-terms have been compared to the M_n count. At this point Register A contains a *composite* FPLA Output Table

obtained when all concurrently selected P-terms are activated by M_n at the FPLA inputs. This table is merged thru an EX-NOR with the contents of the ALR to produce a composite binary F-set, which is in turn compared with the contents of Register B. If they are equal, the M_n generator is incremented, and the test sequence repeated with M_{n+1} until the last minterm. (Alternately, if in manual mode, before incrementing M_n one could observe the Logic Output of the FPLA with M_n as input by calling the contents of the display buffer). If the contents of Registers A and B differ, an error flag is set, and the M_n count halted.

All error conditions detected during Logic Verify will produce conflicting indications in the actual versus computed FPLA Output Tables. Knowing all concurrent P-terms and the logic input to the FPLA, we can analyze the Program Table for further diagnostics and isolation.

THE FPLA AS AN ASSOCIATIVE ARRAY

The logic transmission thru the FPLA can be traced along the equivalent logic path shown in Fig.9.



LET:

$P_n = \Pi_0^{15} (k_m I_m + j_m \overline{I_m})$; k = 0, 1, X (Don't Care)

n = 0, 1, 2, ... , 47

$S_r = .f (\Sigma_0^{47} t_n P_n)$; r ≡ p = 0, 1, 2, ... , 7

where:

Unprogrammed state : $j_m = k_m = 0$; $t_n = 1$

Programmed state : $j_m = \overline{k_m}$; $t_n = 0$

FIG. 9: Equivalent logic path of Signetics' FPLA, and a general set of equations describing the device logic transmission.

The view of FPLAs as two-level logic elements is adequate for all applications. However, a greater insight in their potential applications can be gained by viewing FPLAs as Conditionally Addressable Memories. Thus, Signetics' FPLA can be described as a 48X8 PROM, but a much more useful one, due to a fundamental difference in input structure. In a PROM (Fig.10), all internal words are reached via a fixed decoder inside the device. The size of this decoder, as well as the storage matrix, doubles for each additional address input. The presence of a fixed decoder renders PROM addressing exhaustive. This constraint is at the root of the inefficiency of PROMs in the type of application shown in Fig.11.

Notice that if we define logic "1" as the active-True state of all output functions, it is not possible to compress the truth-table by eliminating inactive minterms 2, 4, and 7. Moreover, with regard to minterms 0 and 1, it is necessary to allocate two distinct storage locations to activate output function O_3 with a single change in

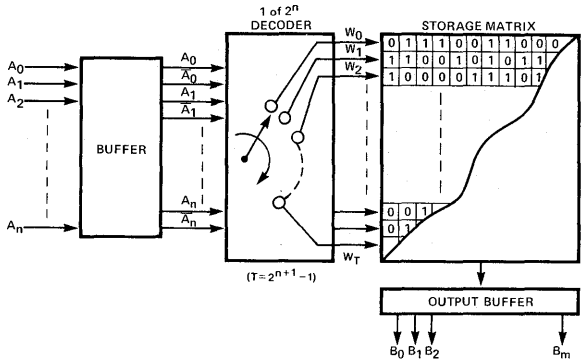


FIG.10: Simplified PROM organization.

M_n	ADDRESS A_2	A_1	A_0	OUTPUTS O_1	O_2	O_3	O_4	
0	0	0	0	1	1	1	1	
1	0	0	1	0	0	1	1	
2	0	1	0	0	0	0	0	
3	0	1	1	0	1	1	1	
4	1	0	0	0	0	0	0	Inactive minterms
5	1	0	1	1	0	1	0	
6	1	1	0	1	1	0	0	
7	1	1	1	0	0	0	0	

FIG.11: Typical truth-table stored in an 8X4 PROM.

input variable A_0. In this case, A_0 represents a logical Don't Care (X) which cannot be directly programmed in a PROM. Instead, separate minterms $\overline{A}_2\overline{A}_1\overline{A}_0$ and $\overline{A}_2\overline{A}_1A_0$ must be programmed.

With an FPLA both constraints are removed. As shown in Fig.12, the FPLA does away with a fixed decoder in favor of a Programmable Address Matrix which offers, in place of forced exhaustive addressing, the flexibility to choose by "linear select" any finite subset from a large number of input states. This is possible because each column of the address matrix functions essentially as a logic comparator programmed to recognize the simultaneous presence of (n) inputs, each either True, False, or both (Don't Care).

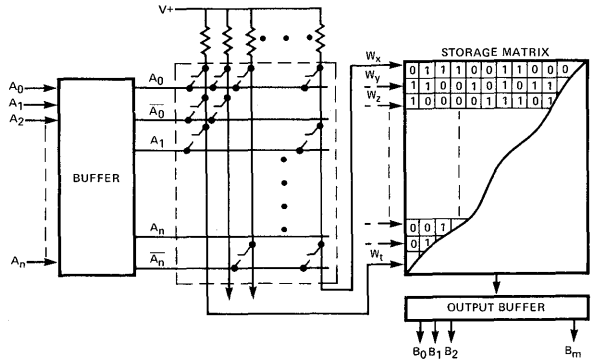


FIG.12: Typical FPLA organization. The input buffer drives a programmable address matrix in which anyone of 2^{n+1} input combinations can be programmed to select a stored word $W_{x,y....t}$.

As a result, storage for inactive minterms is no longer required. The necessary logic output for the inactive minterms occurs by *default*. And, Don't Care states of input variables can be directly programmed in the FPLA. This allows to program the FPLA with either *minterm* or the more general *product terms (P-terms)* of the input variables (addresses).

When any programmed logic combination is present at the FPLA inputs, the corresponding address matrix column (P-term) will be pulled HIGH (logically active), forcing all (B) outputs to their True logic state programmed in the storage matrix. Conversely, for all *unprogrammed* logic combinations present at the FPLA inputs, all columns will remain LOW (logically inactive) forcing all (B) outputs to their False logic state by default *(the complementary logic state of their programmed Active Level Polarity)*. Beacause it is programmable, the FPLA address matrix is not bound in size by the number

of inputs it examines. The matrix has to be only large enough to store the address of 48 words: the FPLA P-terms. The advantage comes about because here we have a choice to select a minimum of any 48 input words (or more, as determined by Don't Care input variables) from a total available pool of 65,536.

Due to the unique capability of FPLAs to store directly Don't Care states, each internal word (W) in the storage matrix can be addressed by several logic input combinations (minterms), given by:

$$(M_n)_T = 2^{m-r}$$

Where: m = total number of input variables

r = number of active inputs (True or Complement) contained in a programmed P-term column.

Thus, if $P_t = XXXI_0$, m = 4 and r = 1, for which $(M_n)_T = 8$.

LOGIC COMPRESSION WITH FPLAs

A concise illustration of logic compression with FPLAs is provided by the truth-table of the Squaring Matrix in Fig.13a. The Boolean form of each output function, including Don't Care states (for clarity), is shown in Fig.13b. Although this table can be directly programmed in a Signetics' FPLA as it is, for sake of illustration all functions have been *individually* minimized by means of Karnaugh maps, and expressed as a sum of product terms.

INPUTS				OUTPUTS							
I3	I2	I1	I0	F7	F6	F5	F4	F3	F2	F1	F0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0	0
0	0	1	1	0	0	0	0	1	0	0	1
0	1	0	0	0	0	0	1	0	0	0	0
0	1	0	1	0	0	0	1	1	0	0	1
0	1	1	0	0	0	1	0	0	1	0	0
0	1	1	1	0	0	1	1	0	0	0	1
1	0	0	0	0	1	0	0	0	0	0	0
1	0	0	1	0	1	0	1	0	0	0	1
1	0	1	0	0	1	1	0	0	1	0	0
1	0	1	1	0	1	1	1	1	0	0	1
1	1	0	0	1	0	0	1	0	0	0	0
1	1	0	1	1	0	1	0	1	0	0	1
1	1	1	0	1	1	0	0	0	1	0	0
1	1	1	1	1	1	1	0	0	0	0	1

FIG.13a: Squaring Matrix for which the output is the square of all 16 input minterms.

$$F_0 = XXXI_0$$
$$F_1 = 0$$
$$F_2 = XXI_1\bar{I}_0$$
$$F_3 = XI_2\bar{I}_1I_0 + X\bar{I}_2I_1I_0$$
$$F_4 = XI_2\bar{I}_1\bar{I}_0 + \bar{I}_3I_2XI_0 + I_3\bar{I}_2XI_0$$
$$F_5 = I_3\bar{I}_2I_1X + I_3I_2XI_0 + \bar{I}_3I_2I_1X$$
$$F_6 = I_3\bar{I}_2XX + I_3XI_1X$$
$$F_7 = I_3I_2XX$$

FIG.13b: Minimized output function set of Squaring Matrix in Boolean form.

To program an FPLA with the minimized function set we first generate an "ACTIVITY" map of the function set involving the ordered P-terms (Fig.14a), and then generate from it a Program Table as the direct source of programmable entries (Fig.14b).

By comparison it can be seen that the original Squaring Matrix has been compressed from 16 minterms to 13 P-terms. Compression has been obtained by means of the "Concurrent", "Selective", and "Multiple" addressing modes characteristic of FPLAs. These can be observed by listing thw FPLA output function set while executing an exhaustive logical scan at its inputs (Fig.15). By viewing each row of the Program

$F_0 = P_0 + / + / + / + / + / + / + / + / + / + / + / + /$
$F_1 = / + / + / + / + / + / + / + / + / + / + / + / + /$
$F_2 = / + P_1 + / + / + / + / + / + / + / + / + / + / + /$
$F_3 = / + / + P_2 + P_3 + / + / + / + / + / + / + / + /$
$F_4 = / + / + / + / + P_4 + P_5 + P_6 + / + / + / + / + /$
$F_5 = / + / + / + / + / + / + / + P_7 + P_8 + P_9 + / + /$
$F_6 = / + / + / + / + / + / + / + / + / + P_{10} + P_{11} + /$
$F_7 = / + / + / + / + / + / + / + / + / + / + / + P_{12}$

FIG.14a: "ACTIVITY" map of function set.
P_n = Function activated by P-term.
/ = Function ignores P-term.

P_n	INPUTS				OUTPUTS							
	I3	I2	I1	I0	F7	F6	F5	F4	F3	F2	F1	F0
0	X	X	X	1	0	0	0	0	0	0	0	1
1	X	X	1	0	0	0	0	0	0	1	0	0
2	X	1	0	1	0	0	0	0	1	0	0	0
3	X	0	1	1	0	0	0	0	1	0	0	0
4	X	1	0	0	0	0	0	1	0	0	0	0
5	0	1	X	1	0	0	0	1	0	0	0	0
6	1	0	X	1	0	0	0	1	0	0	0	0
7	1	0	1	X	0	0	1	0	0	0	0	0
8	1	1	X	1	0	0	1	0	0	0	0	0
9	0	1	1	X	0	0	1	0	0	0	0	0
10	1	0	X	X	0	1	0	0	0	0	0	0
11	1	X	1	X	0	1	0	0	0	0	0	0
12	1	1	X	X	1	0	0	0	0	0	0	0

FIG.14b: FPLA Program Table

Table as an FPLA word selected by the corresponding P-term address, Concurrent addressing is shown by the simultaneous selection of words P_0, P_3, P_6, P_7, P_{10} and P_{11} which occurs with binary 1011 input to the FPLA (Fig.16a). Similarly, Multiple addressing

M_n	INPUTS				CONCURRENT P-terms
	I3	I2	I1	I0	
0	0	0	0	0	None (Default state term)
1	0	0	0	1	P0
2	0	0	1	0	P1
3	0	0	1	1	P0, P3
4	0	1	0	0	P4
5	0	1	0	1	P0, P2, P5
6	0	1	1	0	P1, P9
7	0	1	1	1	P0, P5, P9
8	1	0	0	0	P10
9	1	0	0	1	P0, P6, P10
10	1	0	1	0	P1, P7, P10, P11
11	1	0	1	1	P0, P3, P6, P7, P10, P11
12	1	1	0	0	P4, P12
13	1	1	0	1	P0, P2, P8, P12
14	1	1	1	0	P1, P11, P12
15	1	1	1	1	P0, P8, P11, P12

FIG.16: Simultaneously selected P-terms obtained by exercising the FPLA with a Minterm generator.

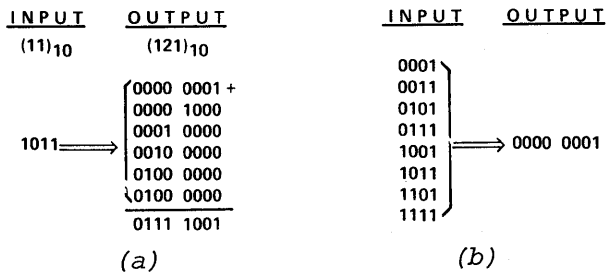


FIG.16: a)Concurrent address selection by $M_n = (11)_{10}$. Internal OR performed by the FPLA.
b)Multiple address selection of P_0.

is readily apparent by observing that word P_0 is selected by 8 different input combinations, in a manner reminiscent of virtual memory storage (Fig.16b).

Selective addressing occurs when minterm "0" is presented at the FPLA inputs, but does not activate any of the programmed P-terms P_0 thru P_{12}.

At this point it is worth noting that the above implementation is not unique, since the Program Table is not unique. This results from the individual, rather than *simultaneous* minimization of the output function set. For example:

$$F_4 = \overline{I}_3 I_2 X I_0 + I_3 \overline{I}_2 X I_0 + \overline{I}_3 I_2 \overline{I}_1 X + I_3 I_2 \overline{I}_1 \overline{I}_0$$

$$F_5 = \overline{I}_3 I_2 I_1 X + I_3 \overline{I}_2 I_1 X + I_3 I_2 X I_0$$

$$F_6 = I_3 \overline{I}_2 X X + I_3 I_2 I_1 X$$

is an equivalent form for $F_{4,5,6}$. This choice of expression, although it introduces an additional P-term in F_4, eliminates P_{12} for realizing F_7, since:

$F_7 = I_3 I_2 X X = (I_3 I_2 I_1 X + I_3 I_2 X I_0 + I_3 I_2 \overline{I}_1 \overline{I}_0$) ←--- contained in $F_{4,5,6}$.

In this case no net reduction in number of P-terms is obtained. However, the method

is at the root of the search for a minimum set of P-terms which will implement the desired logic function. Indeed, the reduction of a set of logic functions of several variables to a minimum set of prime implicants (P-terms) requires a simultaneous minimization process for which suitable algorithms have already been developed. Considerable efforts are currently underway at Signetics for translating such an algorithm in an efficient software program for execution at any of the major timeshare service organizations.

APPLICATIONS

The areas of application in which FPLAs provide a more efficient design alternative span the whole spectrum of logic design. Many applications based on mask-programmable devices have been well documented [1,2,3,4,5]. Since FPLAs can be readily programmed in the field by the user, they are more economical and easier to use, and should soon find their way in a wide variety of design situations specifically suited to FPLAs. The typical design applications described in the following pages emphasize the conceptual aspects of FPLA usage, in order to focus the reader on the basic roles of FPLAs in logic design, and ease the transfer of these basic ideas to a variety of other practical applications.

An estimate of the savings and design advantages obtainable by using FPLAs can be gleaned by examining the recent experience of a Signetics' customer who used FPLAs in the design of parts of an Automatic Landing System for aircrafts. By using a different design approach, he was able to replace 49 IC packages with one FPLA. The tradeoff in both design alternatives is shown in Fig.17. In the discrete approach, $1 is about what it takes today to place one IC on a PC board. The FPLA cost is based on the projected high volume price in 1976.

TYPE	Q
BCD/DEC Decoder(7442)	5
Hex Inverter(7404)	4
8-Input NAND(7430)	2
Quad D-FLOP(74175)	2
Triple 3-NOR(7427)	8
Quad 2-NAND(7400)	12
Quad 2-NOR(7402)	9
Quad 2-AND(7408)	7

COMPARISON		
	Random Logic	FPLA
ICs	49	1
Power	3.3 W	0.6 W
Speed	65 NS	50 NS
Cost	$49	$21
Pins	700	28
Space	50 in^2	2 in^2

FIG.17: The economics of logic replacement with FPLAs. One FPLA replaces 49 ICs at less than half the cost.

MEMORY OVERLAYS

The storage and software efficiency of a computer can be improved by overlaying Read/Write memory with (P)ROM memory in blocks of various sizes, including overlay on an individual word basis. This is also useful in incorporating special diagnostics, or for tailoring machine function to specific customer requirements while maintaining software compatibility.

A typical memory overlay system is shown in Fig.18, in which a flag is used to conditionally transfer (P)ROM or R/W data in the MDR. Since (P)ROMs are available in

discrete chunks confined to standard IC configurations, a lot of storage can be wasted when the application requires overlay of many blocks of few words each, scattered thru the address range of R/W main memory. All unused (P)ROM locations servicing sparsely overlaid sectors are forever inhibited access, and are therefore wasted.

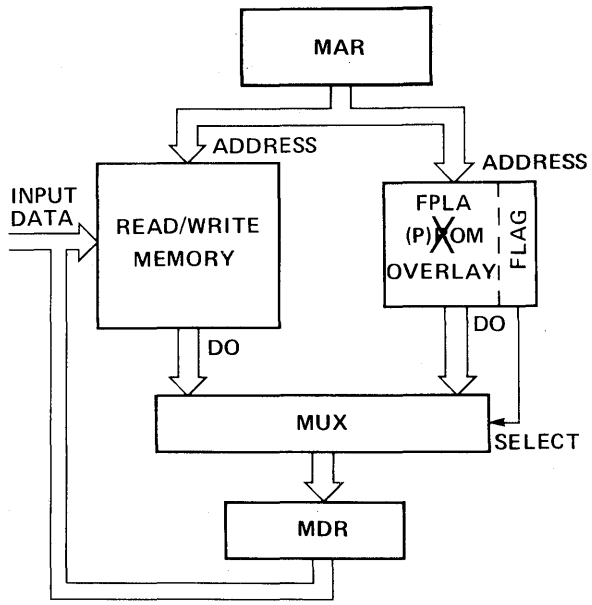


FIG.18: Typical Memory Overlay system. (P)ROM words are jammed in the MDR when the address "present" flag bit is True.

By using an FPLA instead of (P)ROM, the FPLA address matrix is programmed to recognize only the address of the RAM memory locations to be overlaid. The contents of these locations (the RAM modifier) are programmed in the FPLA storage matrix. This way, total read-only storage is compressed to the actual words used. Also, because of the large number of FPLA inputs, the overlaid locations can be scattered anywhere within a 64K address range. The chip enable feature readily extends this range to any practical size by allowing several FPLAs in parallel to examine a larger number of address inputs.

CORE MEMORY PATCH

Modern core planes are available in many sizes, up to 16K X 18 or 32K X 9. A 64K X 9 memory would require two planes, each containing about 300K cores, in which it is not unusual to find as many as 100 broken or improperly tested cores. Currently, defective cores are replaced by a hand "restringing" operation, at a cost of about $2/core. A better alternative to core replacement would be a dynamic repair routine, in which memory addresses containing bad bits are patched by an auxiliary memory. However, since bad cores can be scattered anywhere in the plane, this approach would in general be not cost effective because one would need an auxiliary memory as large as the Core Memory. The FPLA renders this technique economically feasible by providing an address "locator" function by virtue of its programmable address characteristics. The Core Memory address containing bad bits are mapped in the AND matrix of an FPLA, whose

output OR matrix is programmed in turn with sequential address pointers to a small auxiliary RAM containing correct data (Fig.19).

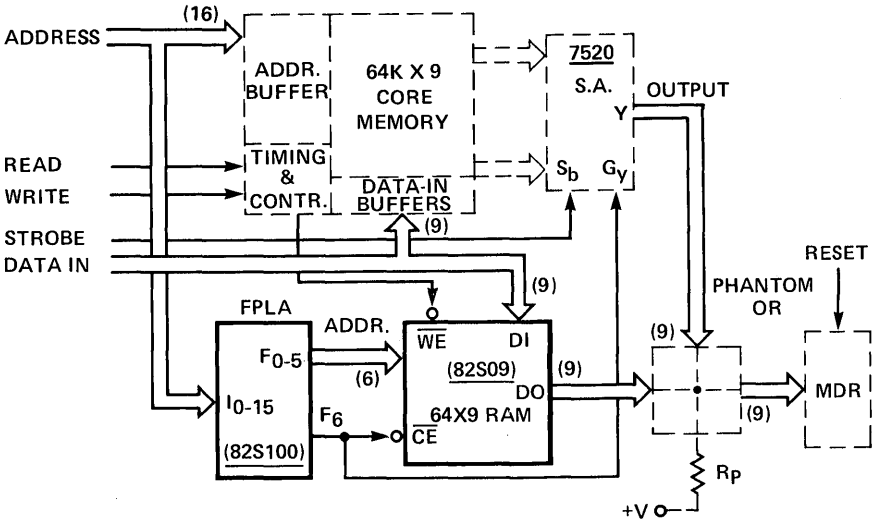


FIG.19: Core Memory "Patch" with an FPLA and auxiliary RAM.

A 16 input FPLA is used as an address map, and a 64X9 RAM (82S09) as the auxiliary memory. The 48 P-terms of the FPLA allow dynamic repair of 48 Core Memory addresses scattered anywhere. Correct data stored in the 82S09 is addressed by 6 FPLA outputs programmed as a binary table. Memory select control is provided by the F_6 output from the FPLA to jam the contents of auxiliary memory in the MDR when a faulty core location is addressed, and to enable writing in auxiliary memory only in the patched locations. The Core Memory system normally contains sockets and connections for both FPLA and auxiliary RAM. The sockets are filled only with partially functional planes. The FPLA Input Table is programmed immediately following final test with the addresses of core failures.

This technique could also be applied, with suitable modifications, to memory systems implemented with partially functional Bipolar or MOS memory devices. It could also be extended to patch modifications in ROM memory systems, or utilize spare locations in PROM memory systems to avoid replacing several packages because of random or frequent changes.

SUBROUTINE ADDRESS MAP AND BRANCH LOGIC

In the design of microprogrammed computers considerable design flexibility is gained by complete freedom in allocating microprogram subroutines throughout microcontrol store, and by the utilization of variable formats in the Instruction Register op-code field. FPLAs are ideally suited for this application as shown in a typical system in Fig. 20. The first FPLA translates the current op-code from a 16-bit Instruction Register into 48 subroutine-start addresses in microcontrol store. Variable op-code formats are easily handled by judicious programming of Don't Care states in the FPLA Input Table. The second FPLA is used to generate branch conditions based on the

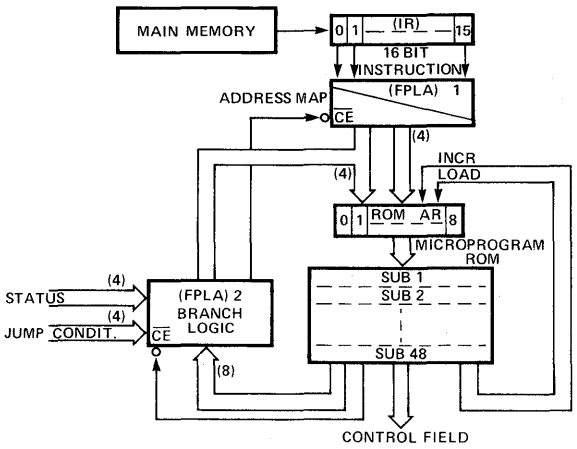


FIG.20: Subroutine Address Map and Branch Logic with tri-state FPLAs.

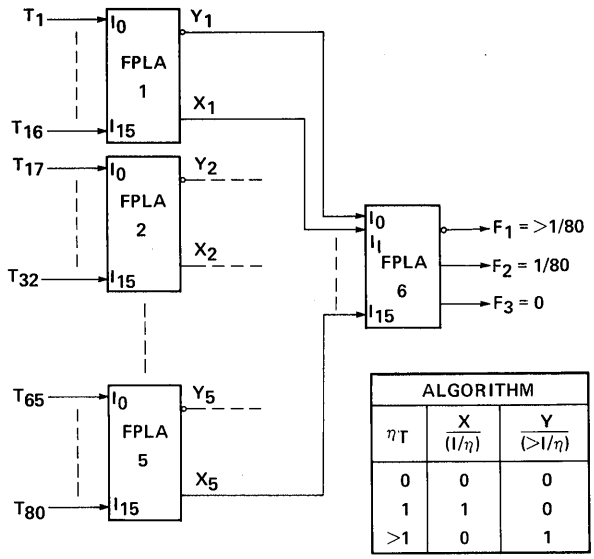
current microinstruction, as well as jump and status conditions in the machine. In particular, using tri-state FPLAs (82S100) saves a multiplexer in the address path of the ROM address register, while their 50 NS access time minimizes overhead time in the instruction execution loop.

FAULT MONITOR NETWORKS

1/N Detectors, as a special case of m/N decoder networks, are useful in a variety of applications in computers, data communications, and fault monitor systems. For example, in a Data Multiplexing system it is not uncommon to find 80 or more channels time-division multiplexed onto a single transmission line. If a fault occurs in the multiplexer control network, multiple or no connections on the line give rise to invalid transmission. These type of faults can be readily detected by using a 1/80 Detector to monitor the normal selection status of only one multiplexer channel at a time. A 1/N Detector could be implemented with logic gates. Excluding inversion and ORing of partial results, the number of gates required is given by the number of logic states to be detected. For 80 status-monitor terminals (one for each data channel):

$$\#\ \text{of Gates}_{(AND)} = N!/(N-1)! = 80$$

This approach, when complicated by the fact that each gate also requires 80 inputs, becomes too impractical. A more practical alternative involves partitioning the number of terminals in equal subsets which are applied to FPLAs whose truth-tables yield outputs x=1/n, and y=1/n. Each FPLA is used as a basic building block in a cascaded array to implement a general algorithm for detecting 0, 1 or more True states (logic "1" = channel selected) of n variables as shown in Fig.21.
Since each FPLA can examine 16 terminals, five are sufficient to service all 80 status-monitor lines. Each FPLA utilizes only 17 P-terms to detect the presence of 0, 1/16, or more connections via outputs (x) and (y) as defined in the Program Table of Fig.22. An additional FPLA is necessary to examine a total of 10 partial x and y



ALGORITHM		
ηT	$\frac{X}{(1/\eta)}$	$\frac{Y}{(>1/\eta)}$
0	0	0
1	1	0
>1	0	1

FIG.21: 1/80 Detector with FPLAs.

results from the first level devices, and to give final indication of the number of selected channels.

INPUT																OUTPUT	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	F_1 x	F_2* y
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

(a)

INPUT										OUTPUT		
−x=1/N−					−y>1/N−					>1/N	=1/N	0
10	9	8	7	6	5	4	3	2	1	F_1*	F_2	F_3
0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	0	0	1	0
0	0	1	0	0	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	0	0	0	1	0
1	0	0	0	0	0	0	0	0	0	0	1	0

(b)

FIG.22: Program Tables of FPLAs used in 1/80 Detector. Starred (*) outputs are programmed active-LOW. a) Program Table stored in FPLAs 1 thru 5. b) Program Table stored in FPLA 6.

PRIORITY RESOLVER AND LATCH

FPLAs can perform the dual function of detecting and latching tri-state bus data on a priority basis. By using only 24 P-terms in a single FPLA, three priority functions can be selected via inputs $S_{0,1,2}$ as shown in Fig.23.

The Reset pulse clears any previously latched priority, and must be at least 30 NS wide to compensate for FPLA delay. Sampling of the system bus begins with the trailing edge of Reset, and ends about 50 NS after detecting an input request (H-→L transition). This delay is provided by the feedback chain of spare gates in the DM8097 buffers, and is required to allow the FPLA to latch the incoming request before releasing the bus. It is also the circuit's resolving time of nearly simultaneous requests.

The FPLA Program Table is shown in Fig.24. The function selected by S_0 provides a 1

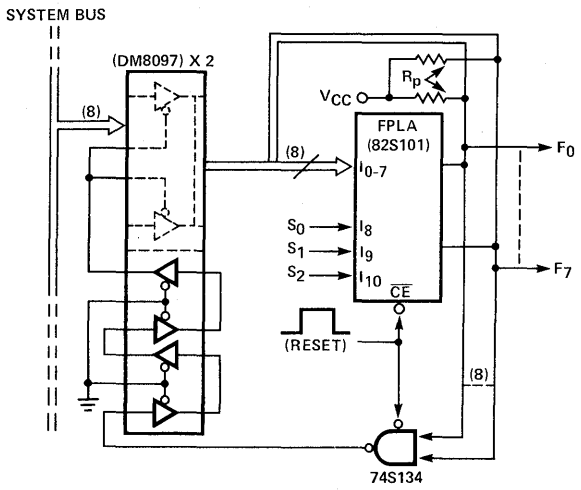


FIG.23: Priority Resolver and Latch with FPLA. The FPLA latched state must be reset prior to sampling new data.

INPUTS											OUTPUTS								
S2	S1	S0	I7	I6	I5	I4	I3	I2	I1	I0	F7	F6	F5	F4	F3	F2	F1	F0	
0	0	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	0	1st of 8 Priority
			1	1	1	1	1	1	0	1	1	1	1	1	1	1	0	1	
			1	1	1	1	1	0	1	1	1	1	1	1	1	0	1	1	
			1	1	1	1	0	1	1	1	1	1	1	1	0	1	1	1	
			1	1	1	0	1	1	1	1	1	1	1	0	1	1	1	1	
			1	1	0	1	1	1	1	1	1	1	0	1	1	1	1	1	
			1	0	1	1	1	1	1	1	1	0	1	1	1	1	1	1	
0	0	1	0	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	
0	1	0	X	X	X	X	X	X	X	0	1	1	1	1	1	1	1	0	1 of 8 Priority (Ascending rank)
			X	X	X	X	X	X	0	1	1	1	1	1	1	1	0	1	
			X	X	X	X	X	0	1	1	1	1	1	1	1	0	1	1	
			X	X	X	X	0	1	1	1	1	1	1	1	0	1	1	1	
			X	X	X	0	1	1	1	1	1	1	1	0	1	1	1	1	
			X	X	0	1	1	1	1	1	1	1	0	1	1	1	1	1	
			X	0	1	1	1	1	1	1	1	0	1	1	1	1	1	1	
0	1	0	0	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	
1	0	0	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	0	1 of 8 Priority (Descending rank)
			1	1	1	1	1	1	0	X	1	1	1	1	1	1	0	1	
			1	1	1	1	1	0	X	X	1	1	1	1	1	0	1	1	
			1	1	1	1	0	X	X	X	1	1	1	1	0	1	1	1	
			1	1	1	0	X	X	X	X	1	1	1	0	1	1	1	1	
			1	1	0	X	X	X	X	X	1	1	0	1	1	1	1	1	
			1	0	X	X	X	X	X	X	1	0	1	1	1	1	1	1	
1	0	0	0	X	X	X	X	X	X	X	0	1	1	1	1	1	1	1	

FIG.24: FPLA Program Table for Priority Resplver. F_{0-7} must be programmed active-LOW. Unused inputs are programmed as Don't Care.

of 8 priority in *time* by latching the first of eight signals occurring on the bus, and is useful in many polling applications in which a 50 NS resolution is adequate.
The functions selected by S_1 and S_2 provide 1 of 8 complementary priorities in *space* by latching the highest ranked signal on the bus.
Both functions are particularly useful in asynchronous multiport systems for transferring control of the main system bus. The concept illustrated is readily expanded with additional output circuitry to monitor up to 16 inputs with any preassigned rank, or to implement a clocked revolving priority of N signals. The primary advantage with the FPLA is that the reassignment of priority rank is facilitated by the ability to combine external selection with FPLA programmability, without resorting to system *wire changes*.

"VECTORED" PRIORITY INTERRUPT SYSTEM

Since FPLAs can store input Don't Care states directly, a simple ranked priority among N signals can be resolved with just N P-terms. With 16 inputs available most FPLA P-terms remain unused. In most applications of this type a more efficient utilization is possible by time-sharing FPLAs to perform separate functions as shown in the design of a "Vectored" priority interrupt system for the Signetics 2650 Microprocessor.
The circuit in Fig.25 is all that is required to service six I/O Devices via the conventional, single level, address vectoring interrupt mechanism of the 2650. When one, or more Devices request service, the CPU receives an $\overline{\text{INTREQ}}$ signal on its single

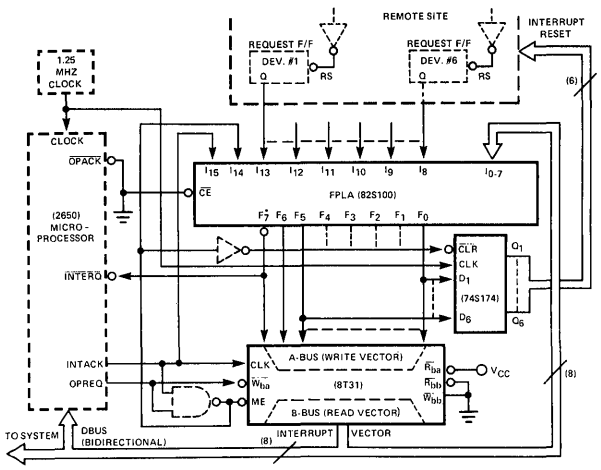


FIG.25: Vectored Priority Interrupt System for the 2650 µP requires 3 ICs, and 2 spare gates. Starred (*) FPLA outputs are programmed active LOW.

2's COMPLEMENT VECTOR	µP D-BUS							
	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
	D/I	+/-	32	16	8	4	2	1
+25 Direct	0	0	0	1	1	0	0	1
-39 "	0	1	0	1	1	0	0	1
+25 Indirect	1	0	0	1	1	0	0	1
-39 "	1	1	0	1	1	0	0	1
+55 Direct	0	0	1	1	0	1	1	1
+38 "	0	0	1	0	0	1	1	0

FIG.26: Vectors pointing to memory locations containing instructions for servicing interrupting Devices.

interrupt pin. Program control is transferred to any of 128 possible memory locations as determined by an 8-bit vector supplied by the FPLA on the CPU data bus, in accordance with a preprogrammed priority. Since memory locations are expressed in 2's complement, the vector can point anywhere within -63 to +64 bytes of page zero, byte zero of memory. Also, both Direct or Relative Indirect addressing modes can be specified by the vector (bit D_7 = 0/1), hence program execution can be directed anywhere within addressable memory. During the execution of the asynchronous CPU handshake the FPLA supplies at various times three distinct functions:

1: *INTACK = OPREQ = 0 --> I_{15} = 0; I_{14} = 1.*

Interrupt request to the CPU, triggered by one or more service requests from Devices 1 thru 6.

2. *INTACK = 1; OPREQ = 0 --> I_{15} = I_{14} = 1.*

Priority resolution of simultaneous requests by placing on the CPU data bus the vector of the highest ranked interrupting Device.

3. *INTACK = OPREQ = 1 --> I_{15} = 1; I_{14} = 0.*

Issue a request reset signal to 1 of 6 selected Devices to acknowledge servicing its interrupt.

The six I/O Devices have been assigned the arbitrary vectors tabulated in Fig.27. The Program Table of Fig.28 shows the FPLA P-terms necessary to execute the above functions, with inputs $I_{14,15}$ used as function selectors under CPU control. Note that it was necessary to program the FPLA with the complement of the vector to compensate for the inversion within the 8T31. The timing diagram of the CPU handshake and FPLA response is shown in Fig.29. For a detailed discussion of CPU response to interrupts refer to the 2650 µP manual.

Several variants of this basic approach have been investigated. In particular, in a case where one needs to service 12 I/O Devices and can tolerate to point the vector

FUNCTION	FUNCTION SELECTOR		PRIORITY/REQUEST GENERATOR						RESET GENERATOR								FPLA OUTPUT							
	I_{15}	I_{14}	I_{13}	I_{12}	I_{11}	I_{10}	I_9	I_8	I_7	I_6	I_5	I_4	I_3	I_2	I_1	I_0	F_7^*	F_6	F_5	F_4	F_3	F_2	F_1	F_0
INTERRUPT REQUEST TO μP	0	1	X	X	X	X	X	1	X	X	X	X	X	X	X	X	0	1	1	1	1	1	1	1
	0	1	X	X	X	X	1	X	X	X	X	X	X	X	X	X	0	1	1	1	1	1	1	1
	0	1	X	X	X	1	X	X	X	X	X	X	X	X	X	X	0	1	1	1	1	1	1	1
	0	1	X	X	1	X	X	X	X	X	X	X	X	X	X	X	0	1	1	1	1	1	1	1
	0	1	X	1	X	X	X	X	X	X	X	X	X	X	X	X	0	1	1	1	1	1	1	1
	0	1	1	X	X	X	X	X	X	X	X	X	X	X	X	X	0	1	1	1	1	1	1	1
PRIORITY RESOLVER	1	1	X	X	X	X	X	1	X	X	X	X	X	X	X	X	1	1	1	0	0	1	1	0
	1	1	X	X	X	X	1	0	X	X	X	X	X	X	X	X	1	0	1	0	0	1	1	0
	1	1	X	X	X	1	0	0	X	X	X	X	X	X	X	X	0	1	1	0	0	1	1	0
	1	1	X	X	1	0	0	0	X	X	X	X	X	X	X	X	0	0	1	0	0	1	1	0
	1	1	X	1	0	0	0	0	X	X	X	X	X	X	X	X	1	1	0	0	1	0	0	0
	1	1	1	0	0	0	0	0	X	X	X	X	X	X	X	X	1	1	0	1	1	0	0	1
RESET REQUEST	1	0	X	X	X	X	X	X	0	0	0	1	1	0	0	1	1	0	1	0	0	0	0	0
	1	0	X	X	X	X	X	X	0	1	0	1	1	0	0	1	1	0	0	1	0	0	0	0
	1	0	X	X	X	X	X	X	1	0	0	1	1	0	0	1	1	0	0	0	1	0	0	0
	1	0	X	X	X	X	X	X	1	1	0	1	1	0	0	1	1	0	0	0	0	1	0	0
	1	0	X	X	X	X	X	X	0	0	1	1	0	1	1	1	1	0	0	0	0	0	1	0
	1	0	X	X	X	X	X	X	0	0	1	0	0	1	1	0	1	0	0	0	0	0	0	1

FIG.28: FPLA Program Table. Only 18 P-terms are necessary to perform three time-shared functions.

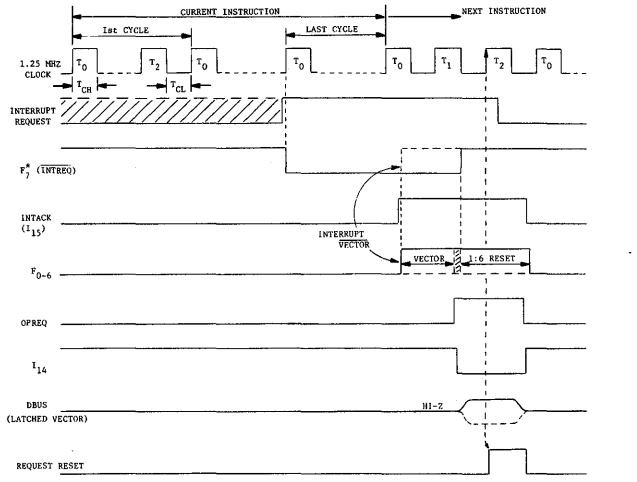


FIG.29: Timing diagram of I/O service request interrupting program execution.

within a narrower memory address range, it is possible to substitute the 8T31 with 4 tri-state buffers, and use the FPLA in a wrap-around connection to latch the vector. The generation of the $\overline{\text{INTREQ}}$ and Reset signals must however be reallocated outside the FPLA.

DEALING WITH DEVICE LIMITATIONS

In many applications a single FPLA cannot accomodate the full Program Table because it exceeds device limitations arising from the finite number of inputs, outputs, and P-terms available. In many cases this can only be overcome by resorting to design intuition and ingenuity in place of complex data manipulations which tend to obscure the problem on hand, and may render troubleshooting difficult.

To increase design flexibility in these situations Signetics' FPLAs are the only ones which feature a Chip Enable input which can be used for input and P-term expansions, preconditional decoding, and output inhibit.

PRODUCT TERM EXPANSION

Expansion of P-terms involving up to 16 input variables is easily accomplished with Open Collector devices, as shown in Fig.30.

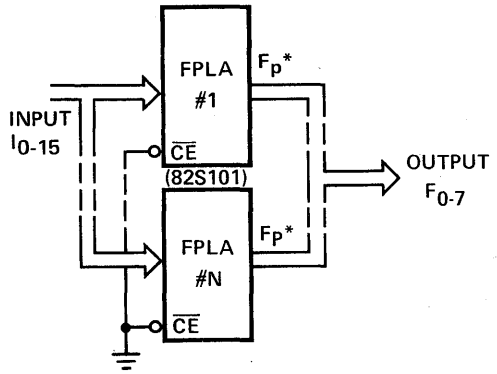


FIG.30; P-term expansion with Open Collector FPLAs. All outputs must be programmed active-LOW (F_p^*) to realize the wire AND function. The total number of P-terms available is 48N.

It is only necessary to parallel respectively all inputs and outputs of several devices, operated with $\overline{CE}$ at ground (unless needed as additional control function). The composite logic output of the network is determined by P-terms activated in one or several FPLAs simultaneously.

When using tri-state devices (82S100), P-term expansion cannot be readily achieved in the same way beacause of logic conflicts ensuing from the active pull-up outputs of FPLAs sharing the same output bus. To ensure enabling only one device at a time, P-term expansion *must* involve the $\overline{CE}$ input.

In most applications requiring more than 48 P-terms it should be a relatively simple task to partition the Program Table in 2 or more Subtables, each containing less than 48 P-terms which in turn can be fitted in separate FPLAs. This partitioning is achieved by *segmenting* the original Table about the "1's" and the "0's" of suitable input variables. Since all P-terms P_n which contain a segmenting variable as Don't Care give rise to 2 P-terms P_{na} and P_{nb}, it is best to segment a Program Table about variable with the fewest Don't Care states.

The logic sources of segmenting variables are removed from the FPLA input field and made to drive instead the $\overline{CE}$ input of the required FPLAs, after proper decoding.

As an example, if one were restricted to use tri-state FPLAs with only 10 P-terms each, to incorporate the Program Table of Fig.14b, a segmentation of this table about input I_2 yields the Subtables shown in Fig.31.

P-terms		INPUTS				OUTPUTS							
P_n	p_n	I_3	I_2	I_1	I_0	F_7	F_6	F_5	F_4	F_3	F_2	F_1	F_0
0a	0	X	0	X	1	0	0	0	0	0	0	0	1
1a	1	X	0	1	0	0	0	0	0	0	1	0	0
3	2	X	0	1	1	0	0	0	0	1	0	0	0
6	3	1	0	X	1	0	0	0	1	0	0	0	0
7	4	1	0	1	X	0	0	1	0	0	0	0	0
10	5	1	0	X	X	0	1	0	0	0	0	0	0
11a	6	1	0	1	X	0	1	0	0	0	0	0	0

(a)

P-terms		INPUTS				OUTPUTS							
P_n	p_n	I_3	I_2	I_1	I_0	F_7	F_6	F_5	F_4	F_3	F_2	F_1	F_0
0b	0	X	1	X	1	0	0	0	0	0	0	0	1
1b	1	X	1	1	0	0	0	0	0	0	1	0	0
2	2	X	1	1	0	1	0	0	0	1	0	0	0
4	3	X	1	0	0	0	0	0	1	0	0	0	0
5	4	0	1	X	1	0	0	0	1	0	0	0	0
8	5	1	1	X	1	0	0	1	0	0	0	0	0
9	6	0	1	1	X	0	0	1	0	0	0	0	0
11b	7	1	1	1	X	0	1	0	0	0	0	0	0
12	8	1	1	X	X	1	0	0	0	0	0	0	0

(b)

FIG.31: a)Subtable A to be stored in FPLA #1 with I_2 removed. P_{11a} can be eliminated since it is "covered" by P_{10}.
b)Subtable B to be stored in FPLA #2, with I_2 also removed.

Each Subtable contains less than 10 P-terms and will fit in separate FPLAs operated in parallel and controlled by I_2 via their $\overline{CE}$ input as shown in Fig.32.

The feasibility of this procedure is strongly dependent on the contents of the original Program Table, and in some degenerate cases it may not work. Also, note that in general the final number of P-terms used may increase due to expansion of input Don't Cares. However, this is preferable to no solution at all.

INPUT VARIABLE EXPANSION

This is the most difficult and cumbersome task with FPLAs. When the Program Table

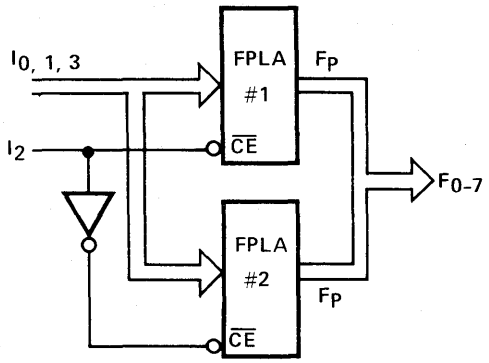


FIG.32: Squaring Matrix function resident in 2 tri-state FPLAs programmed respectively with Subtables A and B. Note that the inhibit function of $\overline{CE}$ renders unnecessary to program the outputs active-LOW.

involves more then 16 inputs, the above partitioning technique by Subtables segmented about any suitable variables will work as well with tri-state or open collector devices. This technique is shown applied to 18 input variables in Fig.33. In this case several devices are necessary, even though not all FPLA P-terms are used. *Note that the expansion capability provided by the $\overline{CE}$ input limits the total number of FPLAs required to 2^n, where (n) is the number of segmenting variables. Without $\overline{CE}$ (as with other competitive devices), the total number of FPLAs would increase as 2^{n+1}.*

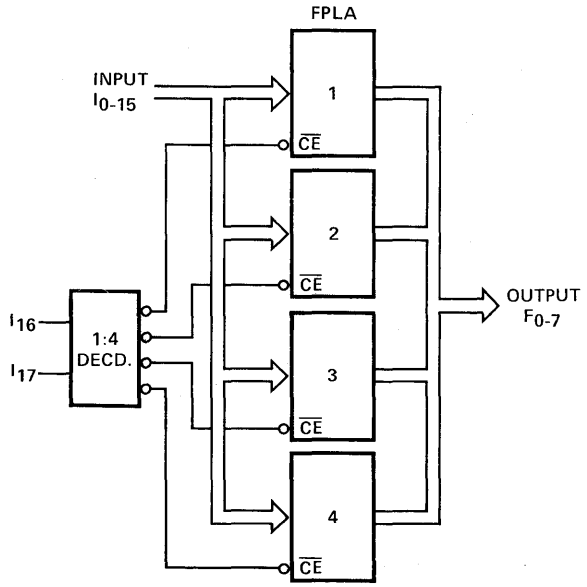


FIG.33: Direct manipulation of 18 input variables using $\overline{CE}$ with either 82S100 or 82S101 FPLAs. Note that here it is not necessary to program all output functions active-LOW (F_p^*) because of the disabling function of $\overline{CE}$.

With more than 20 or so inputs this approach may become too costly, and thus it may make more sense to review the Program Table in conjunction with the problem at hand for ways to multiplex FPLA inputs. This also involves a sort of segmentation of the Program Table for grouping P-terms about input variables which are mutually exclusive. The principle is illustrated in Fig.34a when dealing with only 17 inputs and 5 P-terms, for simplicity.

The original Program Table has been segmented about the "0's" and "1's" of variable I_n, and the P-terms regrouped as in Fig.34b. Note that it was necessary to create new P-terms 4a and 4b to expand the Don't Care for I_n in P-term 4. Also, it is apparent that when $I_n = 0$, the outputs are independent of I_{n-1}, and when $I_n = 1$ the outputs are independent of I_{n+1}. These inputs can be multiplexed in an FPLA with I_n as the

P_n	I_{16} ... I_{n+1}	I_n	I_{n-1} ... I_0	F_x	F_y
0	0 ... X	1	0 ... 1	1	0
1	1 ... 1	0	X ... 1	1	1
2	X ... 0	0	X ... 0	0	1
3	0 ... X	1	X ... X	1	0
4	1 ... X	X	X ... 0	0	1
5	1 ... X	1	1 ... 0	1	0

(a)

P_n	I_{16} ... I_{n+1}	I_n	I_{n-1} ... I_0	F_x	F_y	
1	1 ... 1	0→X	... 1	1	1	
2	X ... 0	0→X	... 0	0	1	UPPER SECTOR
4a	1 ... X	0→X	... 0	0	1	
0	0 ... X←1	0	... 1	1	0	
3	0 ... X←1	X	... X	1	0	LOWER SECTOR
4b	1 ... X←1	X	... 0	0	1	
5	1 ... X←1	1	... 0	1	0	

SELECTOR

(b)

FIG.34: a) Initial Program Table involving 17 input variables, which cannot be directly examined by a single FPLA.
b) Variables I_{n-1} and I_{n+1} are mutually exclusive "about" I_n (selector).

steering condition, as shown in Fig.35. The FPLA Program Table contains Upper P-terms with I_{n-1} variable removed, and Lower P-terms with I_{n+1} removed.

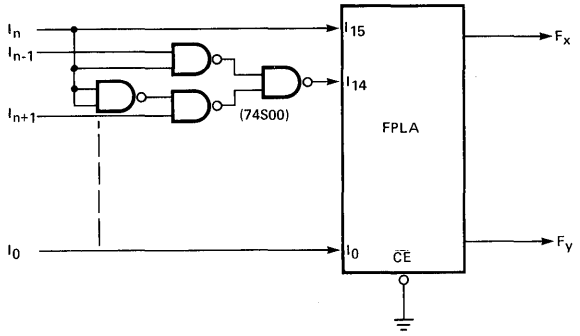


FIG.35: Multiplexing of inputs I_{n-1} and I_{n+1} with selector input I_n allows 17 inputs to be handled with one 16-input FPLA.

When this technique fails too, it may still be possible to factor out of the logic equation of each FPLA output common expressions involving the variables in excess. These can be externally combined with simple gating, or another FPLA, into first level P-terms generating dummy variables to be applied to a second level FPLA.

OUTPUT EXPANSION

If an application requires more than 8 outputs, several FPLAs can be used with parallel inputs and separate outputs. In other cases, it may be more cost effective to encode the Output Table stored in a single device, and then unscramble the desired output states via a 32X8 PROM, or 1/N decoder as required. Both methods are shown in Fig.36, and Fig.37.

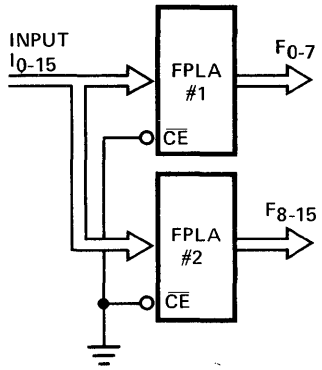


FIG.36: Output expansion by utilizing additional FPLAs.

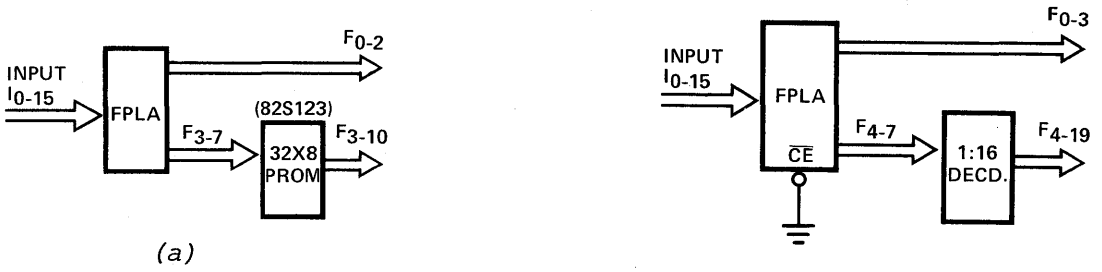


FIG.37: Output expansion by decoding outputs previously encoded in the FPLA Program Table.

REFERENCES

1. *D. Mrazek, and M. Morris, "How to design with Programmable Logic Arrays", National Semiconductor Corp., app. note AN-89, 1973.*
2. *G. Reyling, "PLAs enhance digital processor speed and cut component count", Electronics, August 1974.*
3. *J. Maggiore, "PLA-- A universal logic element", Electronic Products Magazine, April 1974.*
4. *W.N. Carr, and J.P. Mize, "MOS/LSI Design and Application", pp.229-258, T.I. Electronics Series, McGraw-Hill Co., 1972.*
5. *J.C. Logue et al, "Hardware implementation of a small system in PLAs", IBM J. Res. Develop., March 1975.*

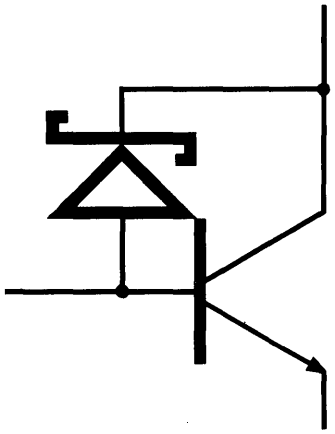
6.

LOW POWER SCHOTTKY TTL

by Peter Alfke and Charles Alford

Fairchild Semiconductor Components Group, Mountain View, California 94042

January 1976



For many years TTL has been the most popular digital integrated circuit technology, offering a good compromise between cost, speed, power consumption and ease of use. As the price of TTL circuits decreased and the average IC complexity increased to MSI (medium scale integration), the cost and size of the power supply and the difficulty of removing the heat dissipated in the TTL circuits became increasingly important factors. Recent improvements in semiconductor processing have made it possible to not only reduce TTL power consumption significantly, but also to improve the speed over that of standard TTL.

The 9LS low power Schottky TTL family combines a current and power reduction by a factor of 5 (compared to 7400 TTL) with anti-saturation Schottky diode clamping and advanced processing. Shallower diffusions and higher sheet resistivity are used to achieve circuit performance better than conventional TTL. With a full complement of popular TTL functions available in 9LS and in the new, more complex and powerful LSI MACROLOGIC™ TTL circuits, low power Schottky is destined to become the dominating TTL logic family.

9LS represents more than just a conventional speed versus power trade-off. This is best illustrated by *Figure 1* which compares 9LS to other TTL technologies. Note that 9LS dissipates eleven times less power than 9S or 74S, suffering a delay increase of only 1.7 times. The 9L low power non-Schottky family by comparison also dissipates eleven times less power than 74H, and 74L dissipates ten times less power than 74N, but both suffer a delay increase of 3.4 times.

The performance of 9LS is not just the result of Schottky clamping. 9LS is four times faster than 9L at the same power dissipation, while 9S and 74S are only two times faster than 74H at the same power. The new and higher level of efficiency exhibited by 9LS is the result of advanced processing, which provides better switching transistors with no sacrifice in manufacturability.

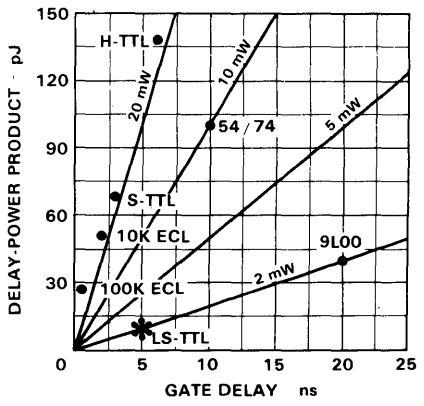


Fig. 1. Delay-Power Product for Popular Logic Families

To the system designer the advantages of this new TTL family are many:

- Less supply current allows smaller, cheaper power supplies, reducing equipment cost, size and weight.
- Lower power consumption means less heat is generated, which simplifies thermal design. Packing density can be increased or cooling requirements reduced, or perhaps both. The number of cooling fans can be reduced, or slower, quieter ones substituted.
- Reliability is enhanced, since lower dissipation causes less chip temperature rise above ambient; lower junction temperature increases MTBF. Also, lower chip-current densities minimize metal related failure mechanisms.
- Less noise is generated, since the improved transistors and lower operating currents lead to much smaller current spikes than standard TTL; consequently, fewer or smaller power supply decoupling capacitors are needed. In addition, load currents are only 25% of standard TTL and 20% of HTTL; therefore, when a logic transition occurs, the current changes along signal lines are proportionately smaller, as are the changes in ground current. Rise and fall times, and thus wiring rules, are the same as for standard TTL and more relaxed than for HTTL or STTL.
- Simplified MOS to TTL interfacing is provided, since the input load current of LSTTL is only 25% of a standard TTL load.
- Ideally suited for CMOS to TTL interfacing. All Fairchild CMOS and most other 4000 or 74C CMOS are designed to drive one 9LS input load at 5.0 V. The 9LS can also interface directly with CMOS operating up to 15 V due to the high voltage Schottky input diodes.
- Best TTL to MOS or CMOS driver. With the modest input current of MOS or CMOS as a load, any 9LS output will rise up to within 1 V of V_{CC}, and can be pulled up to 10 V with an external resistor.
- Interfaces directly with other TTL types, as indicated in the input and output loading tables.
- The functions and pinouts are the same as the familiar 7400/9300 series, which means that no extensive learning period is required to become adept in their use.

CIRCUIT CHARACTERISTICS

The 9LS circuit features are easiest explained by using the 9LS00 2-input NAND gate as an example. The input/output circuits of all 9LS TTL, including, SSI, MSI and MACROLOGIC TTL are almost identical. While the logic function and the base structure of 9LS circuits are the same as conventional TTL, there are also significant differences.

Input Configuration

LSTTL is considered part of the TTL family, but it does not use the multi-emitter input structure that originally gave TTL its name. All 9LS TTL, with the exception of some early designs*, employ a DTL-type input circuit with Schottky diodes to perform the AND function. Compared to the classical multi-emitter structure, this circuit is faster and it increases the input breakdown voltage to 15 V. Each input has a Schottky clamping diode that conducts when an input signal goes negative, as indicated by the input characteristic of *Figure 3*.

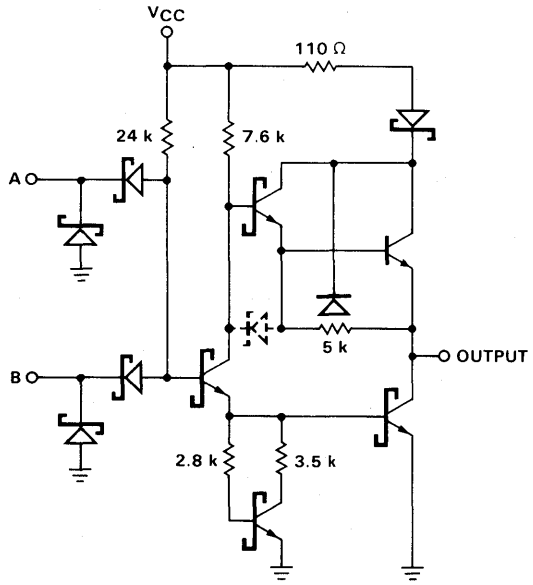


Fig. 2. 2-Input NAND Gate

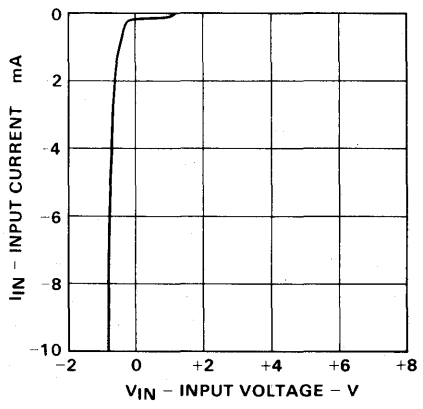


Fig. 3. Typical Input Current-Voltage Characteristic

This helps to simplify interfacing with those MOS circuits whose output signal tends to go negative. For a long TTL interconnection, which acts like a transmission line, the clamp diode acts as a termination for a negative-going signal and thus minimizes ringing. Otherwise, ringing could become significant when the finite delay along an interconnection is greater than one-fourth the fall time of the driving signal.

The effective capacitance of an LSTTL input is approximately 3.3 pF. For an input that serves more than one internal function, each additional function adds 1.5 pF.

Output Configuration

The output circuits of 9LS low power Schottky TTL have several features not found in conventional TTL. A few of these features are discussed below.

- The base of the pull-down output transistor is returned to ground through a resistor-transistor network instead of through a simple resistor. This squares up the transfer characteristics since it prevents conduction in the phase-splitter until base current is supplied to the pull-down output transistor. This also improves the propagation delay and transition time. (See *Figure 4*)

- The output pull-up circuit is a 2-transistor Darlington circuit with the base of the output transistor returned through a 5 kΩ resistor to the output terminals, unlike 74H and 74S where it is returned to ground which is a more power consuming configuration. This configuration allows the output to pull up to one V_{BE} below V_{CC} for low values of output current.

- As a unique feature, the 9LS outputs have a Schottky diode in series with the Darlington collector resistor. This diode allows the output to be pulled substantially higher than V_{CC}, e.g., to +10 V, convenient for interfacing with CMOS. For the same reason, the parasitic diode of the base-return resistor is connected to the Darlington common collector, not to V_{CC}. Some early 9LS designs – the 9LS00, 02, 04, 10, 11, 20, 32, 74, 109, 112, 113 and 114 – do not have the diode in series with the Darlington collector resistor. These outputs are, therefore, clamped one diode drop above the positive supply voltage V_{CC}. These older circuits also contain a "speed-up" diode that supplies additional phase-splitter current while the output goes from HIGH to LOW, and also limits the maximum output voltage to one diode drop above V_{CC}. Since this is the fastest transition even without additional speed-up, this diode is omitted in all new designs.

Output Characteristics

Figure 5 shows the LOW state output characteristics. For low I_{OL} values, the pull-down transistor is clamped out of deep saturation to shorten the turn-off delay. The curves also show the clamping effect when I_{OL} tends to go negative, as it often does due to reflections on a long interconnection after a negative-going transition. This clamping effect helps to minimize ringing.

The waveform of a rising output signal resembles an exponential, except that the signal is slightly rounded at the beginning of the rise. Once past this initial rounded portion, the starting-edge rate is approximately 0.5 V/ns with a 15 pF load and 0.25 V/ns with a 50 pF load. For analytical purposes, the rising waveform can be approximated by the following expression.

$$v(t) = V_{OL} + 3.7\,[1 - \exp(-t/T)]$$

where

$T = 8$ ns for $C_L = 15$ pF and 16 ns for $C_L = 50$ pF

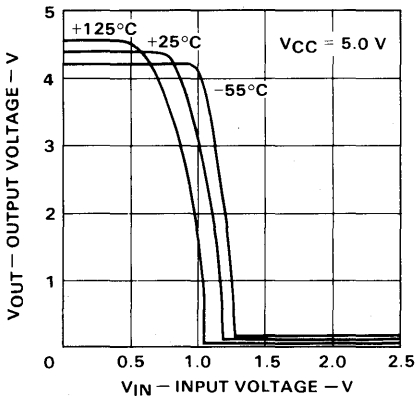


Fig. 4. Typical Output vs Input Voltage Characteristic

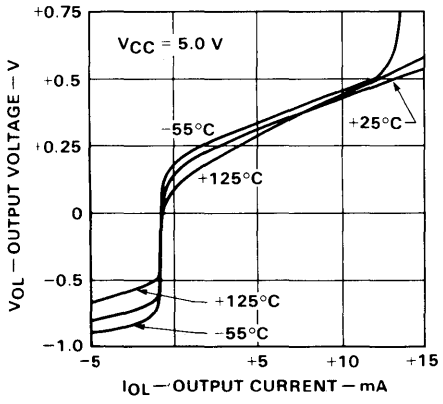


Fig. 5. Typical Output Current-Voltage Characteristic

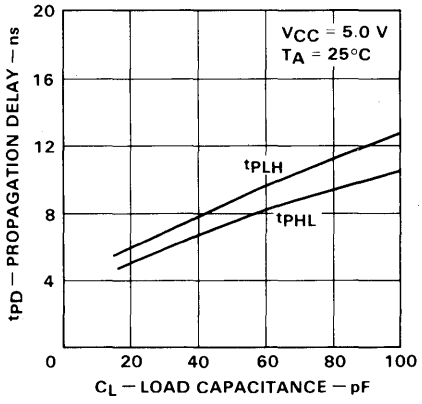


Fig. 6. Typical Propagation Delay vs Load Capacitance

The waveform of a falling output signal resembles that part of a cosine wave between angles of 0° and 180°. Fall times from 90% to 10% are approximately 4.5 ns with a 15 pF load and 8.5 ns with a 50 pF load. Equivalent edge rates are approximately 0.8 and 0.4 V/ns respectively. For analytical purposes, the falling waveform can be approximated by the following.

$$v(t) = V_{OL} + 1.9\,u(t)\,[1 + \cos \omega t] - 1.9\,u(t-a)\,[1 + \cos \omega(t-a)]$$

where

$u(t) = 0$ for $t < 0$
$= 1$ for $t > 0$

and

$u(-a) = 0$ for $t < a$
$= 1$ for $t > a$

For t in nanoseconds and C_L = 15 pF, a = 7.5 ns, ω = 0.42

For C_L = 50 pF, a = 14 ns, ω = 0.23

AC Switching Characteristics

Low power Schottky TTL gates have an average propagation delay of 5 ns measured under the traditional 15 pF load. At higher capacitive loads, the delay increases at a rate of less than 0.1 ns/pF, as shown in *Figure 6*. Although some drive capability is lost by using high value resistors and small transistor geometries in LSTTL, actually more drive is gained by using non-gold doped transistors with much higher current gain than those in conventional TTL. Even at 200 pF load, TTL circuits are still faster than "full power" TTL such as 9000, 7400, 5400.

Figure 7 shows the power dissipation of various logic families as a function of the input frequency. Under static conditions, only CMOS uses less power than LSTTL, but CMOS loses this advantage when operating at more than a few 100 kHz. At speeds over 1 MHz, LSTTL is the most efficient logic.

The delay times of LSTTL are rather insensitive to variations in temperature and supply voltage, as shown in *Figures 8* and *9*. The average propagation delay changes less than 2 ns over the full military temperature range, less than 1 ns over the commercial temperature range, less than 1 ns over the military supply voltage range, and less than 0.5 ns over the commercial voltage range. Compare this to standard TTL where changes of 6 ns over temperature and several ns over supply voltage are typical. As a result, the designer can use the guaranteed maximum delay values with much more confidence and less additional worst-case derating.

Another advantage of LSTTL is its higher output impedance during a positive-going transition. Whereas the low output impedance of STTL and HTTL allows these circuits to force a larger initial swing into a low impedance interconnection, the low output impedance also has a disadvantage. It makes the reflection coefficient negative at the driven end of the interconnection whenever a transmission line is terminated by an impedance lower than the characteristic impedance. Therefore, when the reflection from the essentially open end of the interconnection arrives back at the driver, it will be "re-reflected" with the opposite polarity. The result is a sequence of reflected signals which alternate in sign and decrease in magnitude, commonly known as ringing. The lower the driver output impedance, the greater the amplitude of the ringing and the longer it takes to damp out.

On the other hand, the output impedance of LSTTL is closer to the characteristic impedance of the interconnections commonly used with TTL, and ringing is practically non-existent. Thus no special packaging is required. This advantage, combined with excellent speed, modest edge rates and very low transient currents, are some of the reasons that designers have found LSTTL extremely easy to work with and very cost effective.

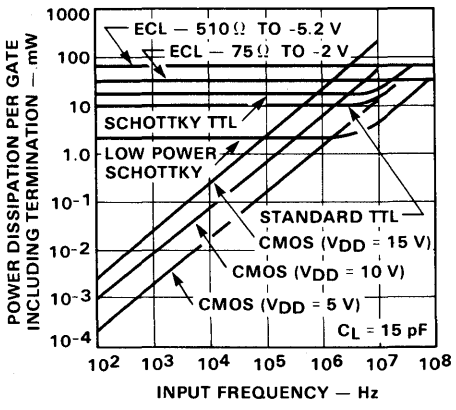


Fig. 7. Typical Power Dissipation vs Input Frequency for Several Popular Logic Families

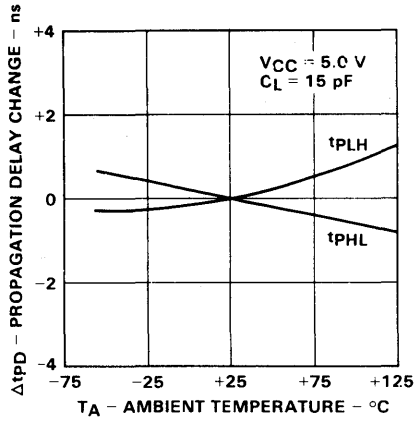


Fig. 8. Propagation Delay Change with Temperature

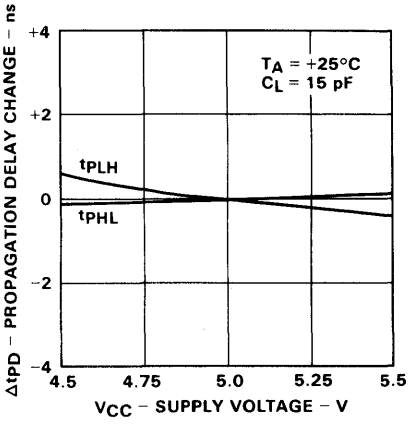


Fig. 9. Propagation Delay Change with Supply Voltage

ANSWERS TO SOME POINTED QUESTIONS

Why is There Such a Difference Between Manufacturers?

When LSTTL was introduced in 1972 by the leading manufacturer of TTL with over six years of design and fabrication experience and several hundred million dollars worth of total TTL sales, (hereafter referred to as TI) there was reason to assume that LSTTL would start off as a mature concept, skillful circuit design and a refined, stable technology. Unfortunatley, this was not the case.

The original 74LS series was introduced prematurely with an overriding emphasis only on low power consumption. Its rather conventional processing and pedestrian circuit design resulted in unnecessary slow components that could not be used as a general substitute for normal TTL. That's why the original 74LS circuits earned the nickname "Low Speed TTL".

More than a year later (early '74) Fairchild, AMD, and Raytheon introduced their LSTTL families, using more advanced processing (shallower diffusions, and higher sheet resistivity resulting in lower parasitic capacitance) and a better choice of resistor values to achieve twice the speed of the original TI circuits. (Fairchild's 9LS has a t_{pd} of 5 ns typ, 10 ns worst case; the original 74LS has a t_{pd} of 10 ns typ, 20 ns worst case). In 1975, TI tried to meet the California challenge by re-specifying the t_{pd} max from 20 to 15 ns. When this was not sufficient, they introduced a brute force circuit change, almost deleting the short circuit current limiting output collector resistor, thus increasing I_{OS} from a range of 5 to 42 mA to a range of 30 to 130 mA. This improves the propagation delay when driving very large capacitive loads, but the large parasitic capacitances on the chip still dominate the delay in all normal applications and make even the re-designed 74LS circuits slower than the Fairchild 9LS circuits for loads up to 150 pF.

9LS started out with a lower collector resistor and a specified I_{OS} of 15 to 42 mA. In an attempt to unify specifications, the newer 9LS circuits are designed with a lower short circuit current limiting collector resistor and typically have an output short circuit current of 60 mA. To accommodate both the older and the newer designs, the I_{OS} range is specified as 15 to 100 mA.

The I_{OS} specification serves two entirely different purposes:

- The maximum value of I_{OS} indicates the potential hazard (current, heat) that exists when a HIGH output is accidentally shorted to ground (by test leads, solder bridges, etc.).
- The minimum value of I_{OS} gives a superficial and often misleading idea of the drive capability on the LOW-to-HIGH output transition. In spite of lower I_{OS}, the original 9LS circuits are actually superior to the re-designed 74LS circuit.

The input threshold voltage is another circuit design parameter that causes some confusion. Most manufacturers use Schottky diode AND gating driving the phase splitter. This results in an input threshold voltage of $2\ V_{BE} - V_{Schottky}$, *i.e.*, roughly 1.0 V at 25°C with a negative temperature coefficient of 3 mV/°C. This input threshold voltage is about 0.3 V lower than for standard TTL. Many alternate circuit configurations with higher

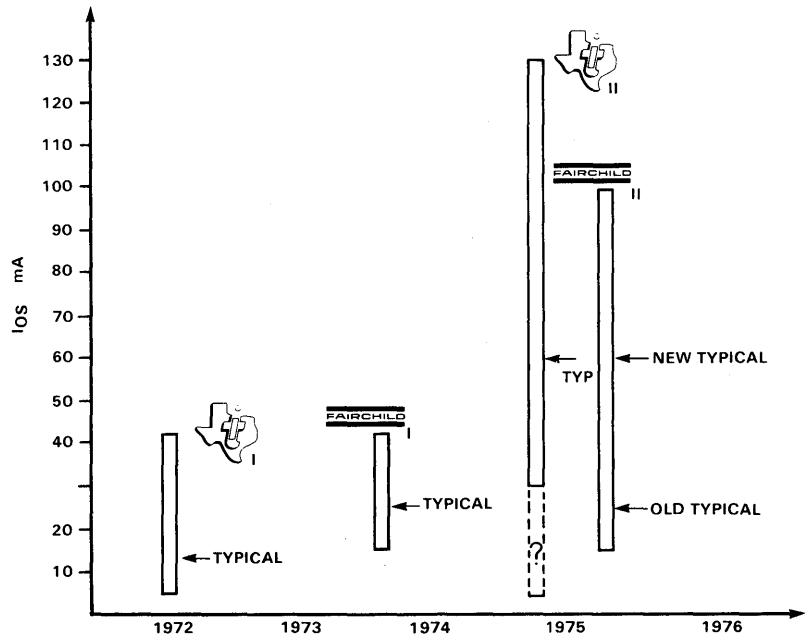


Fig. 10. Evolution of the 74LS/9LS Output Short Circuit Current

input thresholds were evaluated in an effort to improve the LOW state noise immunity. After examining the trade-offs, most manufacturers realized that additional level shifting components would either slow down the circuit or their additional gain would adversely affect both the output fall time and the sensitivity to narrow pulses, and thus actually reduce the system noise immunity.

Today only Signetics uses such a non-standard input structure. All other manufacturers have settled on the same input circuit design as a good compromise between dc and ac noise immunity and speed.

What Can Be Done with the Outputs?

All 9LS circuits (except the open collector types) use the same "totem pole" output structure with a Darlington pull-up transistor, the base of the output transistor returned through a resistor to the output terminal, and the base of the pull-down transistor returned to ground through an active squaring circuit. A Schottky diode in series with the collector resistor allows the output to be pulled substantially higher than V_{CC}, *e.g.,* to +10 V for interfacing with CMOS. In power-down applications (V_{CC} = 0 V), the output can be pulled to +10 V. This is of interest in memory applications where LS circuits perform the address and clock driving under normal circumstances, but 3-state CMOS buffers maintain the memory refresh in a stand-by mode when the TTL power is removed.

For open collector devices the output HIGH state is characterized in terms of leakage current I_{CEX}, since there is no pull-up circuit to establish a V_{OH}. Note that the worst-case limit (100 μA) assumes that the input level is at the edge of threshold (0.8 V for inverting gates). At the more realistic input levels of 0.2 to 0.5 V, *i.e.,* well below threshold, I_{CEX} is only a few microamps. For devices with the "totem pole" pull-up, I_{CEX} in the pull-down transistor is not specified since the output HIGH voltage V_{OH} is a more meaningful specification for all normal applications. I_{CEX} is, however, of interest in special cases like CMOS or MOS interfacing or in powered-down operation. For these cases, it is reasonable to assume that I_{CEX} for the "totem pole" output is the same as for the open collector output under the same conditions.

What is the Current Spike?

It is well known that during each TTL output transition, there is a brief period of time when both the pull-up and pull-down transistors are conducting. This conduction overlap allows current to flow from V_{CC} to ground, with a duration that depends on how well the turn-on and turn-off delays of the two transistors are balanced. In LSTTL these delays are so well balanced that the current spike through the totem pole has an amplitude of only 3 mA and a duration of only 5 ns.

There are usually much larger transient currents in the V_{CC} and ground leads as a result of the rapid charging and discharging of load and interconnection capacitances. For example, a positive-going LSTTL output charging a 50 pF load results in a 12.5 mA spike in the V_{CC} lead. Similarly, a negative-going output discharging 50 pF results in a 25 mA spike in the ground lead.

These spike current values are probably greater than will actually occur in a system because they assume that all of the capacitance is lumped at the driver output pin. A total load capacitance of 50 pF implies about ten driven inputs, considering that wiring capacitance is about 1 pF/in. and each LSTTL input represents about 3.3 pF. Given the physical constraints of layout in an actual system, the driven inputs are necessarily located at varying distances from the driver, which means that not all elements of load capacitance can charge or discharge simultaneously. Since wiring propagation delays range from about 0.12 to 0.18 ns/inch, an element of load capacitance located, for example, three inches from the driver would receive a signal transition about 0.4 ns later than a load located right at the driver. Furthermore, it would take another 0.4 ns for the effects of the charging or discharging to travel back to the driver and alter the output current. Thus it is reasonable to assume that the current spikes in an actual system will have less amplitude and longer duration than those measured in a test jig with a lumped capacitor load.

What Happens on Long Interconnections?

Discussions of TTL line drive capability and, more specifically, what kinds of TTL to use in different situations, tend to generate more heat than light. It is generally agreed that communications between cabinets are best left to specialized drivers and receivers that provide complementary drive, impedance matching and common mode noise rejection. For data bus applications, where two or more drivers are connected to a line, a popular technique is to use open-collector drivers that can sink 80 mA or so in the quiescent LOW state. These drivers can handle the currents dictated by impedance-matching resistive terminations at both ends of the bus.

However, most interconnections in a system are relatively short and do not require such special circuits. Nevertheless, there are usually some areas wherein timing is critical and the interconnections cannot be regarded as lumped capacitances but must be treated as transmission lines. The characteristic impedance and propagation delay factor of a signal trace (or wire) are determined by its distributed inductance and capacitance. These, in turn, are partly dependent on the proximity of the ground return. Since most logic boards use some type of ground grid, as opposed to a continuous ground plane, it follows that signal trace characteristics vary. Actual measured values would be preferred, of course, but in their absence an impedance of 150 Ω and a delay factor of 0.15 ns per inch are workable averages.

The characteristic impedance of a line is the ratio of a voltage change to the current change that accompanies it as it moves along the line. When the output of a circuit starts to change state, the line impedance looks like a load resistor returned to the voltage that existed just before the change started. Thus, in *Figure 11*, when the output of gate 1 tries to pull down from a quiescent V_{OH} of +4.4 V, its dynamic load is a 150 Ω resistance returned to +4.4 V. This kind of load presents a problem since a worst-case LSTTL output can, under nominal conditions, only sink 10 mA. Thus, if the delay of the signal line BC is long compared to the output fall time, the voltage at B will initially drop by only 1.5 V (10 mA times 150 Ω), to 2.9 V. Since this is above the switching threshold (1.3 V) of gate 2, a delay is incurred at B while waiting for a reflection to arrive from C.

The input impedance of an LSTTL element is so much greater than 150 Ω that the signal line can be considered open-ended. Thus when the initial −1.5 V transition arrives at C it starts to double and simultaneously generates a −1.5 V reflection. When this reflection arrives at B it also sees a high impedance, since the output of gate 1 is not in clamp, and thus it starts doubling and pulls the driver output voltage down below the threshold of gate 2. In the process, a reflection is generated at B which returns to C and pulls the input of gate 3 down below threshold. Summarizing, the added delay at B is twice the delay of the signal line and the added delay at C is three times the line delay. This simplistic approach gives pessimistic results for short lines because the rate of change at the beginning of an output transition is rather slow. To compensate for this, 1 ns can be subtracted from the calculated delays, except of course that the corrected result cannot be negative.

If the LSTTL driver in the foregoing example is replaced by a 9N/74 or an STTL or HTTL circuit, there is no added delay at B because these circuits can sink at least twice as much current as the equivalent LSTTL circuit and also because their quiescent V_{OH} is only 3.8 V compared to 4.4 V. The initial transition is thus sufficient to pass through the switching threshold of gate 2. These more powerful circuits thus provide the designer with alternatives when the added delays incurred with LSTTL are incompatible with system timing requirements. It is also possible to use the LSTTL buffer, which can sink at least 24 mA.

In the arrangement of *Figure 12*, the driving output sees a 75 Ω dynamic load and a 10 mA current sinking capability gives an initial transition of only −0.75 V. To bring the voltage at B down through the switching threshold of gate 2 requires two reflections from the end of each line, which means that the added delay is twice the sum of the line delays. Once again, a correction factor of up to 1 ns must be subtracted for each line delay. The added delay at point C is the sum of the added delay at B (as corrected) plus the delay of line BC. Similarly, the added delay at D is the additional delay at B plus the delay of line BD.

For a positive going output the pull-up capability, which is proportional to short-circuit output current, is a determining factor. In *Figure 11*, an LSTTL circuit having the 15 mA minimum I_{OS} can force enough current into the 150 Ω dynamic impedance to pull the voltage up through the switching threshold of gate 2 without waiting for a reflection from C. The added delay at B is zero if the line delay is 0.5 ns or less. For a line between 0.5 ns and 1.75 ns the added delay increases linearly from zero to 1.5 ns. Line delays longer than 1.75 ns have no further effect. The added delay at point C is no more than just the line delay, and can be ignored if the line delay is 1 ns or less.

In *Figure 11*, a positive going output will pull the voltage at B up to about 1.0 V initially, and the voltage will rise through the switching threshold of gate 2 when a reflection arrives from the end of the shortest line. The added delay at B is twice the delay of the shortest line. The added delay at the end of either line is the delay from B to the end in question plus, at most, 0.8 ns.

The worst case pull-up capability of STTL and HTTL circuits is almost three times that of LSTTL. When one of these circuits serves as the driver in *Figure 11*, the initial positive going voltage at B exceeds the switching threshold of gate 2. Thus the added delay at B is insignificant. The added delay at the end of a line is just the

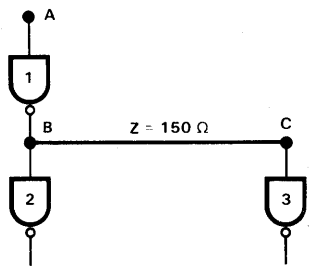


Fig. 11. Signal Line Driven from One End

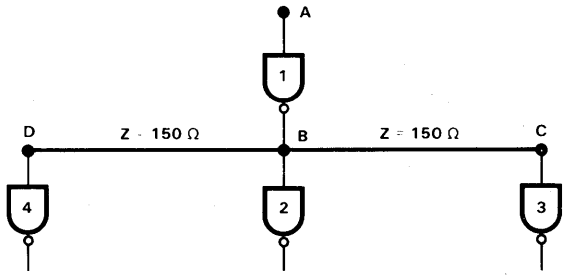


Fig. 12. Signal Line Driven in the Middle

line delay. Thus these more powerful circuits also offer a faster alternative for positive going signals into low impedance loads.

Propagation delays of LSTTL, measured in the traditional manner with lumped capacitance loading, are shorter than those of standard TTL and, with modest interconnection lengths, permit higher system operating rates. With long interconnections, however, the smaller current sinking capability of LSTTL leads to added delays for negative going signals that diminish the speed advantage of LSTTL over standard 9N/74 TTL. In the absence of direct measurements, the designer can use the approximation methods outlined above to estimate the added delays and can, in marginal situations, use an LSTTL buffer where the logic permits, or can select a circuit from the more powerful TTL families.

What Happens to Unused Inputs?

For best noise immunity and switching speed, unused AND or NAND-gate inputs should not be left floating, but should be held between 2.4 V and the absolute maximum input voltage. Two possible ways of handling these unused inputs are:

Connect an unused input to V_{CC}. Most 9LS inputs have a breakdown voltage $>$ 15 V and require, therefore, no series resistor. For all multi-emitter conventional TTL inputs, a 1 to 10 kΩ current-limiting series resistor is recommended to protect against V_{CC} transients that exceed 5.5 V.

Connect an unused input to the output of an unused gate that is forced HIGH.

Note, do not connect an unused LSTTL input to another input of the same NAND or AND function. This method, although recommended for normal TTL, increases the input coupling capacitance and therefore reduces the ac noise immunity.

What Happened to 9300 MSI?

All 9LS and 74LS circuits follow the well-established 7400 series numbering scheme. For the sake of uniformity in nomenclature Fairchild has given up the traditional 9300 numbers on LS parts even for those MSI functions that were originally invented and introduced by Fairchild many years before they were copied and re-named by the 7400 proponents. The following circuits are, therefore, functionally and pin-out identical:

9024	9LS109	DM8093	9LS125
9300	9LS195	DM8094	9LS126
9310	9LS160	DM8095-8	9LS365-8
9316	9LS161	DM8214	9LS253
9321	9LS139	S8280/90	9LS196
9322	9LS157	S8281/91	9LS197
9334	9LS259	DM8560	9LS192
9341	9LS181	DM8563	9LS193
9350	9LS290	DM8570	9LS164
9356	9LS293		

The following circuits are functionally identical, but have different pin-outs:

9301	9LS42
9312	9LS151

All 9LSXXX and 74LSXXX parts are functionally and pin-out identical with the 74XX part, with the following exceptions:

The 9LS160 through 163 and the 74LS160 through 163 counters are fully edge triggered devices, (like their Schottky counterparts) without any of the mode control timing constraints of the 9310, 9316, 74160 through 74163.

Many of the newer LS circuits have no equivalent 7400 counterpart.

What Does it Cost?

Presently, LSTTL circuits are slightly (25 to 30%) higher priced than their normal TTL counterparts. Even at this higher component price it can be shown that they offer a lower systems cost, taking into account the saving in power consumption and heat removal. Each LS gate package saves approximately 30 mW and each LS MSI package saves 150 to 300 mW.

In commercial, line-operated equipment, a Watt of power consumption usually costs between $1 and $2 for power supply, power distribution, and heat removal. In portable, airborn, or space applications the cost per Watt is significantly higher.

A switch to LSTTL therefore is economical even at today's prices. Over the next years the price of LSTTL will come down further and may even become lower than that of conventional TTL.

WHAT IS A SCHOTTKY DIODE?

A Schottky diode, also called a "hot carrier diode", offers two big advantages over the conventional pn-junction diode – very high speed due to extremely short recovery time and a substantially lower forward-voltage drop for a given current, or an order-of-magnitude higher current for the same voltage.

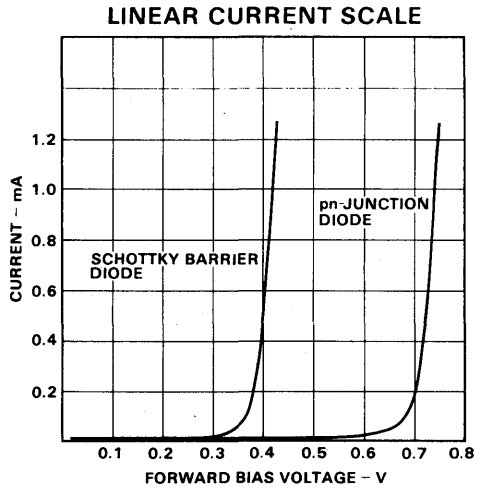
The more familiar pn-junction diode that exists at the boundary of two differently doped sections inside a semiconductor crystal relies on *minority* carriers for current transport. In contrast, a Schottky diode is formed by the metal-to-semiconductor contact at the surface of the semiconductor crystal and relies on *majority* carriers for current transport (electrons in the case of n-type semiconductor). Charge storage is negligible and forward-to-reverse recovery is extremely fast.

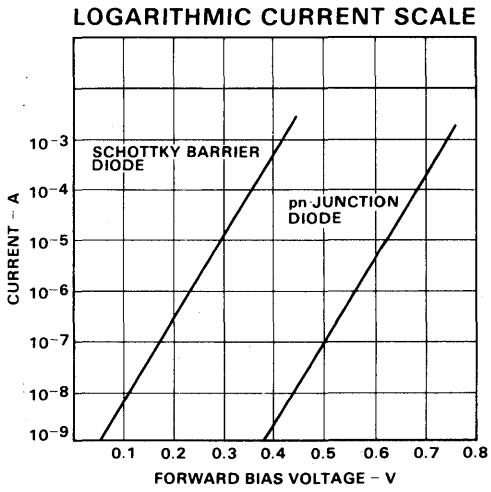
Metal-semiconductor contacts can be classifed into two groups according to their current/voltage characteristics. Those contacts with a *linear* characteristic are called ohmic and are used extensively in monolithic integrated circuits for interconnecting the various components on a chip. Those with a *non-linear* rectifying characteristic are called Schottky barrier diodes. Whether a metal-semiconductor contact is ohmic or rectifying, *i.e.,* has linear or non-linear characteristics, depends on the properties of the metal and on the doping level and the type of the semiconductor.

The rectifying non-linearity of a Schottky diode results from the presence of a potential barrier at the metal-semiconductor interface, which the carriers must surmount by thermionic emission before they can flow through the junction. The barrier potential can be reduced by forward bias (metal more positive than the n-type semiconductor) to increase the carrier flow from the semiconductor into the metal. Under reverse bias, the Schottky diode behaves similarly to a pn-junction diode; the reverse current is small and almost voltage independent unless the breakdown voltage is exceeded.

The Schottky barrier height is always less than the energy gap of the semiconductor. Thus, for a given voltage, the current flowing in a Schottky barrier diode is orders-of-magnitude larger than in a pn diode of the same area; but, the forward current follows the same exponential law, doubling for every increase of 18 mV in forward voltage, *i.e.,* increasing tenfold for every voltage increase of 60 mV (See Figure).

Rectifying metal-semiconductor contacts were discovered and investigated by Ferdinand Braun in 1874. Despite many attempts to understand their current-flow mechanism, the correct physical model was not discovered until half a century later by Walter Schottky. Researchers at Bell Labs in the late forties were investigating metal-semiconductor interfaces when they accidentally discovered the transistor. From then on, most efforts in the semiconductor industry have been directed toward pn-junction devices. Only in the past six or seven years have manufacturers gained the understanding of surface phenomena and developed the metallization techniques required to produce reliable Schottky barrier diodes.





WHY SCHOTTKY TTL?

With the use of Schottky diodes, the saturation delay normally encountered in saturated logic (TTL, DTL and RTL) can be avoided. These logic families operate by turning their transistors either fully on or fully off. The amount of base current applied to turn on a transistor is critical. Too little current will not turn the transistor on sufficiently. Too much current will turn the transistor on quickly; however, when the base current is interrupted, the transistor continues to conduct until the excess charge in the base disappears, usually through thermal recombination.

The designer of saturated logic circuits therefore faces a problem. He must design the circuit parameters so that each transistor receives sufficient base current even under the worst-case combination of manufacturing tolerances—positive resistor tolerances and low transistor current gain(beta)—and environmental conditions —low supply voltage, low temperature that reduces beta and increases V_{BE}, and high fan-out that increases the collector current of the output stage. On the other hand, he must be concerned about overdriving the transistor and causing excessive saturation delays under the opposite worst-case conditions—negative resistor tolerances, high beta, low V_{BE}, high temperature, high supply voltage and low fan-out—where the transistor may receive ten times more base current than required.

Conventional TTL circuits use gold doping to increase the probability of thermal base-charge recombination, thus decreasing saturation delay; but, this also lowers beta and makes the circuit less efficient. As early as 1955, an elegant circuit trick, the Baker clamp, was developed to overcome this problem. A diode is connected between the base and the collector; originally a germanium diode was used. If this diode has a very low forward-voltage drop, it starts conducting when the collector becomes slightly forward biased with respect to the base. The excess current applied to the base terminal of the transistor then flows through this diode into the collector. The transistor only receives the base current necessary to pull the collector into the "soft saturation" region. There is no excess charge storage and the saturation delay is non-existent.

At first, monolithic integrated circuits could not use this trick, since no pn-junction diode was available with a voltage drop significantly lower than that of the base-emitter diode. The Schottky barrier diode, however, has this desirable characteristic. By 1970, a great deal of progress had been made in the understanding and manufacturing of these diodes. Metal-silicide and refractory-metal contacts assured high temperature stability and the surface effects of silicon were better understood and controlled. All the major TTL manufacturers introduced a line of Schottky TTL circuits where all the transistors that normally would be saturated are equipped with anti-saturation Schottky barrier clamp diodes. These Schottky TTL circuits are very fast; but, since the emphasis is on speed, they consume more power than normal TTL and their short rise and fall times cause interconnection problems.

Low power Schottky TTL consumes one-quarter the current and power of conventional TTL and uses Schottky diode clamping and advanced processing to regain the speed that is lost because of the lower internal charging currents. The 9LS family offers performance superior to conventional TTL while saving 75% of the power dissipation.

WHO IS MR. SCHOTTKY?

"Schottky" has become a semiconductor household word, yet how many engineers know the man behind the name? Is he

1. a famous soviet scientist, inventor of **ТТЛ ЛОГІК**?
2. one of the three Nobel prize winners from Bell Labs who invented the transistor in 1948?
3. a German physicist who invented the screen-grid tube in 1915?
4. a research scientist with one of the leading U.S. semiconductor manufacturers?

If you checked 1, 2 or 4, you are wrong, but this would not be surprising for we researched several libraries before we found only the briefest biographical information on Walter Schottky.

He was born in June 1886 in Zürich, Switzerland, where his father, a well known German mathematician, had a teaching assignment. Walter Schottky received doctorates in engineering and natural sciences from the University of Berlin and spent several years as a professor at the universities of Würzburg and Rostock. He also worked in the research department of Siemens, the German telecommunications giant.

Most of his early research dealt with electrons and ions in vacuum tubes. He invented the screen-grid tube in 1915 and later discovered an irregularity in the emission of thermions, known as the "Schottky effect"—the reduction in the minimum energy required for electron emission under the influence of an electrical field. During the thirties, Walter Schottky worked mainly on the theory of semiconductor physics which, at that time, had the bad reputation of being the "physics of dirt effects" or the study of "order-of-magnitude effects". Semiconductors such as selenium and copper-oxide rectifiers, overvoltage protectors and photovoltaic cells were used commercially but there was no clear understanding of their theory of operation. Walter Schottky established the boundary-layer theory for crystal rectifiers that explains how special concentration and potential conditions exist in the boundary layer of the semiconductor and how these conditions depend on the current through the rectifying junction.

Walter Schottky remained active in semiconductor research for several decades until his death in 1956. He wrote several books, few if any translated into English, and published many articles in scientific journals, *e.g.*, Zeitschrift für Physik and Annalen der Physik.

7. THE AD 7550 - A 13-BIT
"QUAD SLOPE" ANALOG TO DIGITAL CONVERTER

WILL RITMANICH
Applications Engineer
Analog Devices Microsystems
Santa Clara, CA.

INTRODUCTION:

In applications where conversion speed is of secondary importance to accuracy requirements, the integrating type analog to digital converter has long been a popular choice of designers because of its cost-effectiveness. Presently, there are a number of monolithic IC's on the market which will provide 10 bit or greater accuracy. All these earlier designs utilizing the traditional dual slope technique or derivatives of it, however, suffer one or more of the following drawbacks, thus limiting their usefulness in many applications:

1. Lack of temperature stability - Amplifier and comparator offset drift affects zero and gain TC's. This imposes restrictive operating temperature ranges in order to maintain 25°C accuracy.

2. Poor flexibility or a requirement for many additional external components. - Bipolar input capability, ease of interfacing with control logic and flexible power supply voltage requirements with low power consumption are desirablee features which are not available in many of today's monolithic IC sets.

This paper will describe a revolutionary new device, the AD7550, which alleviates all these shortcomings and employs a recently patented* conversion technique called "Quad-Slope" that consists of four slopes of integration as compared to the conventional dual slope. This unique conversion method, with it's digital correction scheme, makes the conversion virtually insensitive to amplifier and comparator offset drift, and comparator hysteresis effects. Performance features include 1 ppm°/C zero offset and gain drift and nearly infinite power supply rejection. The AD7550 includes the CMOS amplifier, comparator, digital and switching logic on it's 118 X 125 mil chip. It needs only an external RC for integration and a single positive reference supply (which allows ratiometric operation). It furnishes 13 bits of parallel binary data in microprocessor compatible 2's complement code that can be outputted through it's three-state buffers onto an 8 bit data busline.

* PATENT #3872466

THE "QUAD SLOPE" INTEGRATION PROCESS

The technique that allows this superior performance is best explained through Figures 1 and 2.

The inputs AGND. VREF, AIN and VREF are applied in sequence to an integrator (time constant τ = RC) creating four slopes at the output of the integrator. The integrator has a voltage equal to half the reference voltage applied to its + input at all times. Hence the equivalent input voltages to the integrator are as follows:

$$\text{Phase 1: } V_{IN} = AGND - \frac{VREF}{2} = -\frac{VREF}{2}$$

$$\text{Phase 2: } V_{IN} = VREF - \frac{VREF}{2} = +\frac{VREF}{2}$$

$$\text{Phase 3: } V_{IN} = AIN - \frac{VREF}{2} = -\frac{VREF}{2} + AIN$$

$$\text{Phase 4: } V_{IN} = VREF - \frac{VREF}{2} = +\frac{VREF}{2}$$

However, the offset voltage e and drift effects of the op amp distort these equivalent input voltages to:

$$\text{Phase 1: } V_{IN} = -\frac{VREF}{2} - e$$

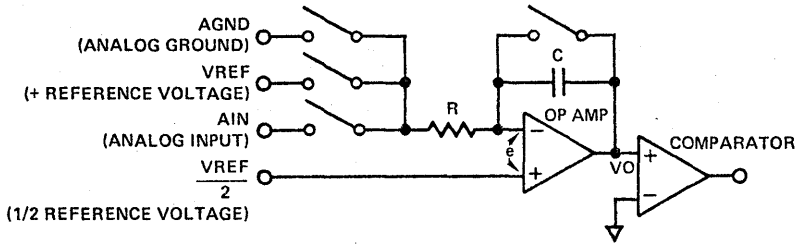
$$\text{Phase 2: } V_{IN} = +\frac{VREF}{2} - e$$

$$\text{Phase 3: } V_{IN} = -\frac{VREF}{2} + AIN - e$$

$$\text{Phase 4: } V_{IN} = +\frac{VREF}{2} - e$$

(The reference voltage VREF must be positive for proper operation.)

AD7550



Phase 1: $VIN = -\frac{VREF}{2} - e$

Phase 2: $VIN = +\frac{VREF}{2} - e$

Phase 3: $VIN = -\frac{VREF}{2} + AIN - e$

Phase 4: $VIN = +\frac{VREF}{2} - e$

Fig. 1

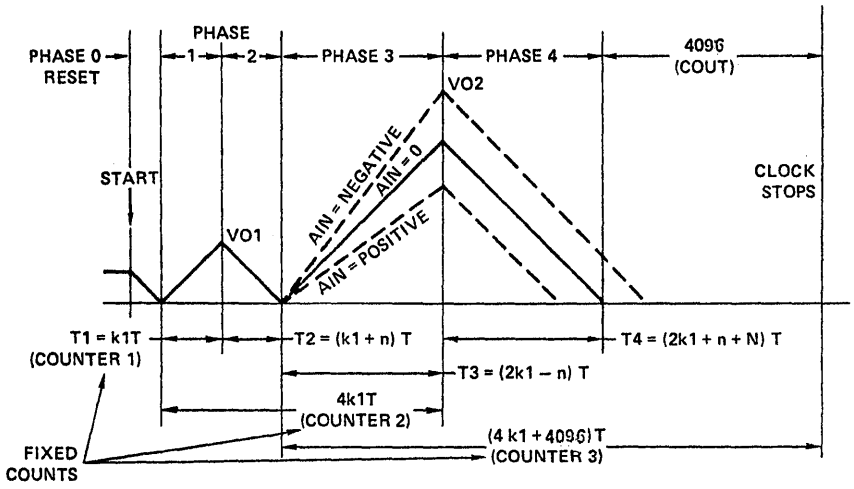


Fig. 2

Initially, during Phase 0, the integrator is ramped to the zero crossing of the comparator in order to start the quad slope process. Phase 0 can be considered as the reset phase of the converter. It is important to note that the comparator will always reach its zero crossing-point from the same direction throughout all the phases, exhibiting, therefore, constant propagation delay and no hysteresis effect.

Phase 1 will integrate $(- \frac{VREF}{2} - e)$ for a fixed period of time equal to $T_1 = k_1 T$; k_1 being a fixed number of pulses of duration T (clock frequency = $\frac{1}{T}$)1 The output of the integrator reaches a value V_{01} at the end of T_1 and is equal to:

$$V_{01} = -(- \frac{VREF}{2} - e).\frac{1}{\tau}$$

$$= (\frac{VREF}{2} + e).\frac{k_1 T}{\tau} \qquad (1)$$

The input is switched to $(+ \frac{VREF}{2} - e)$ and phase 2 ramps down for a period of time $T_2 = (k_1 + n)\ T$ such that:

$$-V_{01} = - (+ \frac{VREF}{2} - e\ \frac{(k_1 + n)\ T}{\tau}$$

This combined with equation (1) gives:

$$n = \frac{2k_1}{\frac{VREF}{2e} - 1}$$

or $$e = \frac{VREF}{2} . \frac{n}{2k_1 + n} \qquad (2)$$

Phase 3 starts now, integrating $(-\frac{VREF}{2} + AIN - e)$ to a final value of V_{02}. However, a second counter started counting pulses from the beginning of phase 1 and counts to a fixed value of $4k_1 T$, determining the end of the integration of phase 3. This means that T_3 will be equal to:

$$T_3 = 4k_1T - T_2 - T_1$$

$$T_3 = 4k_1T - (k_1 + n)T - k_1T$$
$$T_3 = (2k_1 - n)\ T$$

and we have:

$$V_{02} = -\ (\ -\frac{VREF}{2} + AIN - e).\ \frac{(2k_1 - n)_T}{6} \quad (3)$$

Phase 4 ramps down from this value with an input of $(\ +\frac{VREF}{2} - e)$ and hits the zero-crossing after a time T_4 which we consider equal to:

$$T_4 = (2k_1 + n + N)\ T$$

We can now show that N is a number of counts proportional to the AIN and essentially free of errors (due to e)

Indeed, the phase 4 ramp gives us:

$$-V_{02} = -\ (+\frac{VREF}{2} - e).\ \ (\frac{2k_1 + n + N)\ \ T}{6}$$

and combined with equations (2) and (3) we obtain:

$$N = \frac{-AIN}{VREF} \cdot k_1 \cdot [\ 1 - (\frac{n}{2k_1})^2\ ]\ - \frac{n^2}{k_1} \quad (4)$$

Therefore the basic equation (e = o, n = o)

$N = \frac{-AIN}{VREF} \cdot k_1$ has only second order effects from the offset voltage (and drift effects) on its scale factor and its zero value.

Due to practical limitations, equation (4) could not be implemented in its simple form in the AD7550. Indeed the "count out"

(C_{OUT}) of the device and hence its transfer function is:

$$N = (\frac{AIN}{VREF}\ \ 2.125 + 1)\ 4096$$

This means that N runs from zero to $(2^{13} - 1)$ counts for AIN from $(\frac{-VREF}{2.125})$ to $\frac{VREF}{2.125}$ (1 - LSB)), LSB meaning least significant bit.

N is derived from a third counter which was started at the end of phase 2 which counts to a time period

$$(4k_1 + 4096)\ T$$

and establishes the negative overrange. It is the count output which is gated out at the final comparator crossing that corresponds to the corrected count N of the transfer function and stops at the end of this period. Should the final comparator crossing not be attained by the end of this period, a negative overrange is indicated while a positive overrange is indicated by a zero crossing prior to the start of T_4.

AD7550 PERFORMANCE CHARACTERISTICS

A principle advantage of the "Quad-Slope" technique is that it can be implemented very readily in a monolithic CMOS IC. The high circuit desnity of digital CMOS can be used to compensate for the relatively mediocre performance of the CMOS linear devices. "Quad Slope" performance would be much more difficult (and costly) to attain in a discrete design.

Capable of operating over wide voltage ranges, the AD7550 operates from V_{DD} supply of +12 to +15V, a V_{SS} of 0 to -15V, and a logic supply, V_{CC}, of +5V to V_{DD} depending upon whether TTL or CMOS logic level interface is needed. Power dissipation is extremely low due to it's CMOS construction (8mw typical). Conversion can be externally initiated by a positive pulse or can be self-started for continuous conversion by adding a capacitor to it's "start" pin. To accomodate various design requirements, the free-running intermal clock may be used, or the AD7550 may be driven from an external clock of 100 KHz to 1 MHz maximum resulting in a conversion time of 40 ms over it's bipolar input range. In addition to it's parallel binary data outputs controllable through its "high byte enable", "low byte enable", and "status enable" commands, it furnishes a serial output pulse stream which is equal to 8191 pulses for a full scale input with overrange and polarity indicators.

DESIGN EQUATIONS

Provisions are made for user selection of conversion speed, full scale voltage and data-read time. The following steps, in conjunction with Figure 3, explain the calculations of the component values required to satisfy these user selected requirements.

1. Determination of VREF

   When the full sclae voltage requirement (VFS) has been ascertained, the reference voltage can be calculated by

$$VREF = 2.125\ (VFS)$$

2. Determination of Clock Frequency

After establishing the full scale voltage (VFS) and conversion time ($t_{convert}$) requirements, the clock frequency (f_{clk}) requirement is given by

$$f_{clk} = \frac{3.5 \times 10^4}{t\ convert} + \frac{VFS\ (1.8 \times 10^4}{t\ convert\ (VDD-4V)}$$

f_{clk} must be limited to frequencies less than 1.3 MHz for proper operation of the AD7550. If the internal clock oscillator is used, Figure 4 can be used to determine the required value of C_3 for a given f_{clk} requirement.

If the AD7550 is driven from an external clock, the conversion time ($t_{convert}$) is given by

$$t_{convert} = \frac{3.53 \times 10^4}{f_{clk}} + R_1\ C_1$$

where $R_1\ C_1$ is the integrator time constant (see next paragraph.

3. Selection of Integrator Components (R_1 and C_1)

To ensure that the integrator doesn't exceed it's bound during the phase 3 integration cycle, the integrator time constant ($R_1\ C_1$) is given by

$$\tau = R_1\ C_1 \frac{VREF\ (9 \times 10^3)}{f_{clk}\ (V_{DD}\ -4V)}$$

The integrator components R_1 and C_1 can be selected by referring to Figure 5 and/or Figure 6. Figure 5 plots the time constant ($R_1\ C_1$) versus clock frequency for different reference voltages. Figure 6 is a direct plot of the required C_1 versus f_{clk} for R_1 values of 1MΩ and 10MΩ at different reference voltages.

R_1 can be a standard 5% resistor, but must be selected to fall within the 1MΩ to 10MΩ ranges such that

$$R_1 = \frac{\tau}{C_1}$$

The integrating capacitor "C_1" must be a low leakage, low dielectric absorption type such as teflon, polystyrene or polypropylene. To minimize noise, the outside foil of C_1 must be connected to IROUT.

4. Selection of C_2 (Auto Start Operation Only)

The size of C_2 determines the length of time which data is available at the DB0 through DB12 outputs (assuming LBEN and HBEN = 1) after conversion is completed. This is the "read time" (t_{read}) of the AD7550, and is given by $t_{read} = C_2(1.7 \times 10^6 \Omega) + 20\mu s$

When first applying power to the AD7550, a positive pulse (power-up restart) may be required at the STRT terminal for the Auto Start Operation to begin.

AD7550 APPLICATIONS

Basic Operation

Figure 3 shows the basic circuit connection for binary operation. With all the data output commands held high as shown, parallel data will be present on all the outputs. By selectively excercising the various commands, HBEN (4MSB's plus polarity bit), LBEN (8 LSB's) and STEN (overrrange, $\overline{BUSY}$, BUSY) the desired data can be placed out on an 8 bit databus.

Single Supply Operation

To insure linearity of the AD7550's amplifier, a potential of +4V minimum should exist between AGND and V_{SS}. However, because AGND is not a supply path, it can be biased upward to allow single supply operation as shown in Figure 7. This configuration would be especially useful for battery pack operation but it is not without it's limitations. The value of the MSB is this circuit is directly dependent upon the T.C. of the resistor used, so to assure accuracy over a wide temperature range, a metal film or wire-wound resistor should be selected.

Digital Linearization Technique

Figure 8 shows a block diagram of how the serial count output of the AD7550 can be modified before being clocked back into it's three-state buffers, providing both a linearized BCD value of a thermocouple output for digital panel meter display plus the same value represented in binary format which can be outputted on a microprocessor data busline. For a data acquisition system, addition of a line driver/receiver will allow the AD7550 output to be supplied over a two wire line to the linearizing circuitry, allowing digital transmission of the analog value for visual display.

In this configuration the analog input to the AD7550 is derived from an instrumentation type amplifier such as an AD521 to which the thermocouple and cold junction inputs are applied. A conversion is started and an output pulse stream is generated by the AD7550 which is determined by the thermocouple voltage and the scale factor of the amplifier compared to the reference voltage. This pulse stream is clocked through a set of CMOS rate multipliers connected in cascade which perform the linearizing function by modifying the pulse count output as determined by the binary value applied to it's inputs. This binary value is selected by determining the scale factor used, the temperature range over which the readings will be taken and the accuracy required. By dividing the temperature range into several segments and calculating the BCD value needed to linearize each segment, a digital approximation of the piecewise linear correction technique can be realized. Clocking the pulses out of the CMOS rate multipliers into a set of BCD counters, and monitoring the outputs of the counters to successively change ranges as the pulse count continues, allows the BCD value to the rate multiplier to be changed for each segment.

In this circuit, accuracy is primarily a function of the number of segments used and the range desired. The digital linearization technique insures drift free performance over temperature without the usual expense and problems of using programmed ROM's. By applying an analog voltage to the input of the op amp which corresponds to a desired temperature center point, very accurate readings can be obtained for any particular temperature range.

SUMMARY

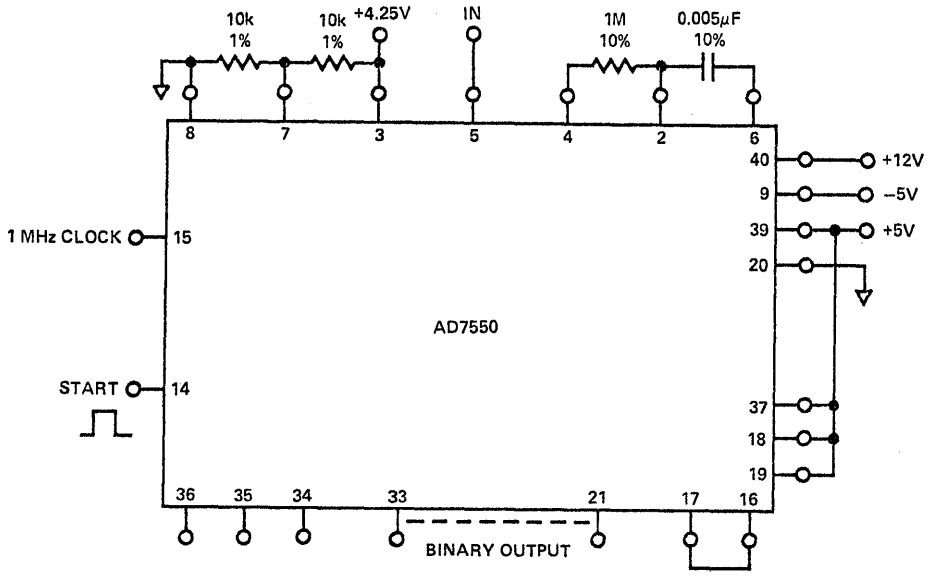
A state-of-the art advancement in analog to digital conversion techniques has been devised which is incorporated in the design of the AD7550, promising performance levels previously thought unattainable in monolithic devices. In addition to superior performance, the AD7550 requires minimal external components and is capable of operating over wide power supply ranges with very low power consumption. It can form the basis of an accurate data acquisition system with an ability to interface directly to an 8 bit databus, providing binary outputs in 2's complement code through it's three-state buffers.

BIBLIOGRAPHY

A-D Conversion Handbook, Edited by D. H. Sheingold, Analog Devices, Inc. Norwood, Mass 02062

Nonlinear Circuits Handbook, Edited by D. H. Sheingold, Analog Devices, Inc. Norwood, Mass 02062

AD7550
BASIC OPERATION



Notes: 1. The integrating capacitor has to be polystyrene or polypropylene type.
2. Conversion time is about 40 ms.
3. All digital inputs/outputs are TTL compatible (VCC = +5V).

Fig. 3

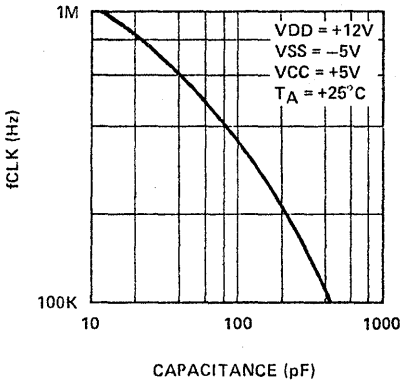


Fig. 4

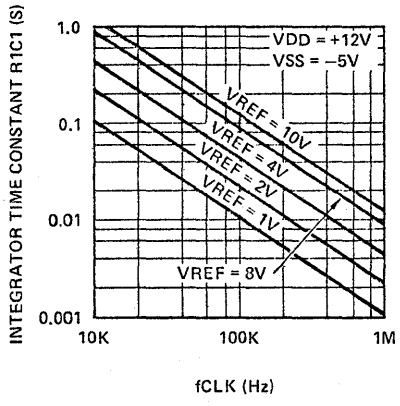


Fig. 5

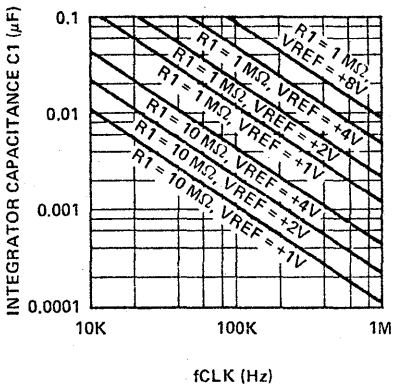


Fig. 6

AD7550
SINGLE SUPPLY OPERATION

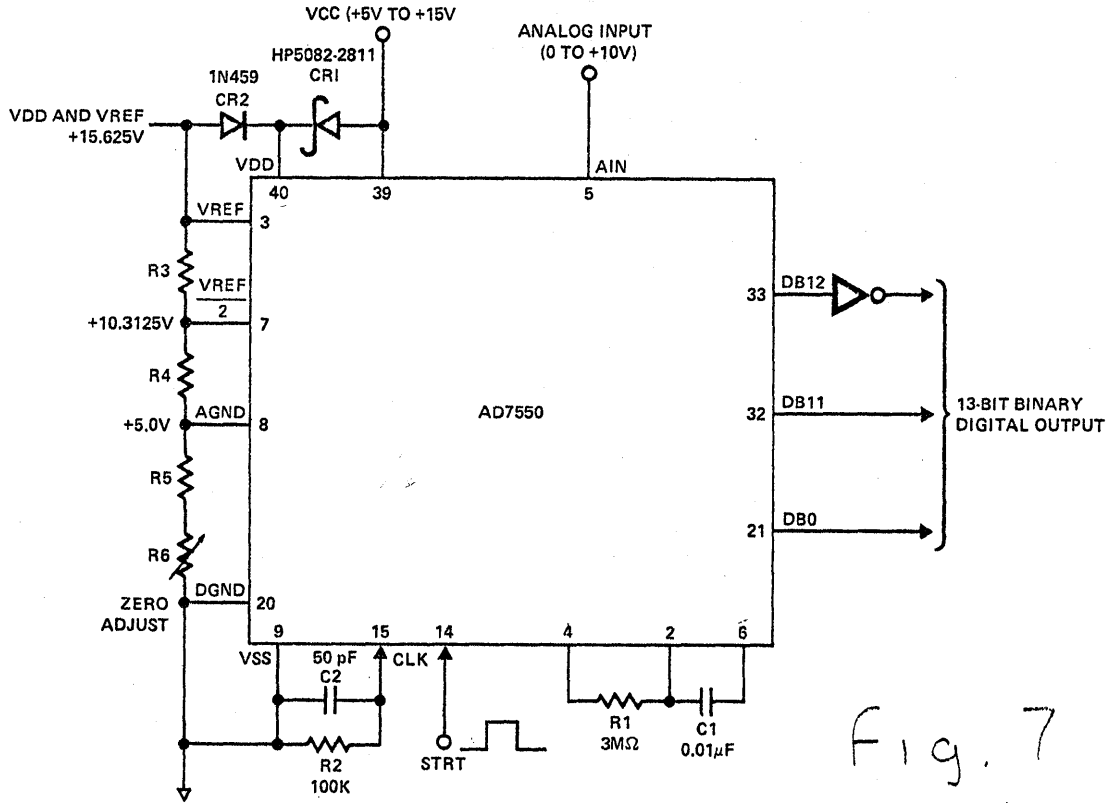


Fig. 7

AD7550
DIGITAL TRANSDUCER LINEARIZATION

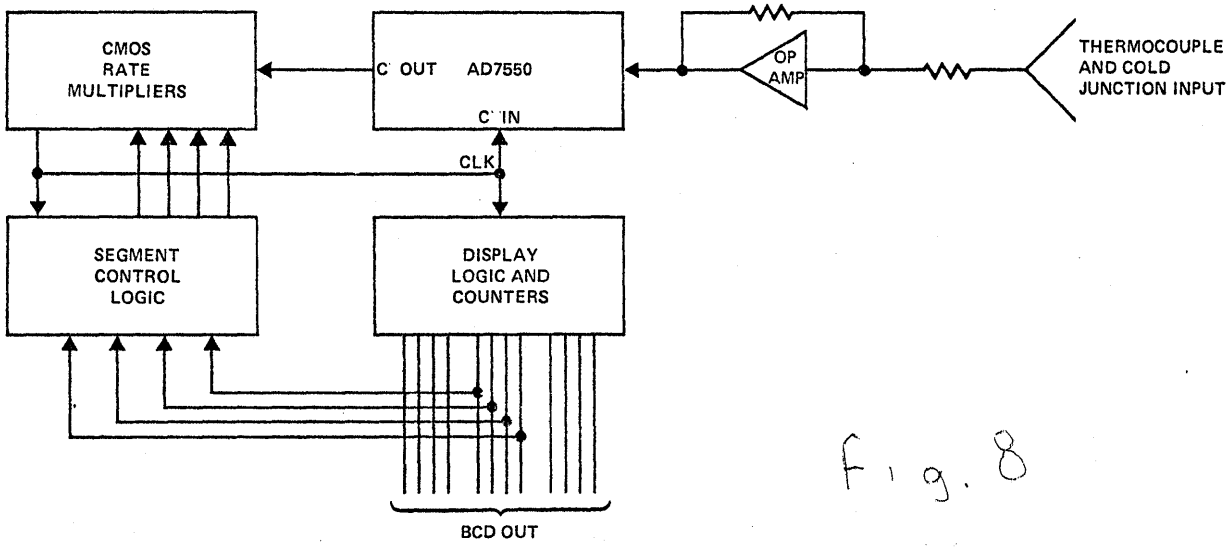


Fig. 8

8. CUSTOM IC DESIGN USING I^2L TECHNOLOGY

Alan B. Grebene
Vice President
Exar Integrated Systems, Inc.
Sunnyvale, California

Integrated Injection Logic (I^2L) is one of the most significant recent advances in the area of monolithic LSI technology. Compared to other monolithic LSI technologies, I^2L offers the following unique advantages:

- o High Packing Density
- o Bipolar Compatible Processing
- o Lower Power and Low Voltage Operation
- o Low (Power x Delay) Product ($\leqslant$ 1pJ)
- o Higher Speed than MOS

Figure 1 shows a comparative listing of some of the cost and performance advantages of I^2L technology, in comparison with other monolithic LSI technologies. Figure 2 gives a comparison of the speed and power capabilities of various logic families, including I^2L.

Since I^2L technology is a direct extension of the conventional bipolar IC technology, it readily lends itself to combining high-density digital functions and complex analog functions on the same chip.

		I^2L (N+ Isolated)	PMOS	NMOS (Si Gate)	CMOS	Schottky T^2L	ECL
Performance	Speed (Fastest = 1)	3	6	5	4	2	1
	Speed Power Product (Lowest = 1)	1	4	2	3	5	6
	Linear/Digital Compatable	Yes	No	No	No	Yes	Yes
Cost	Functional Density (Highest = 1)	1	2	1	3	4	5
	Mask Steps (Lowest = 1)	2	1	3	4	5	5
	Diffusions (Lowest = 1)	2	1	2	3	4	4

Figure 1

Cost and Performance Comparison of Basic Logic Families

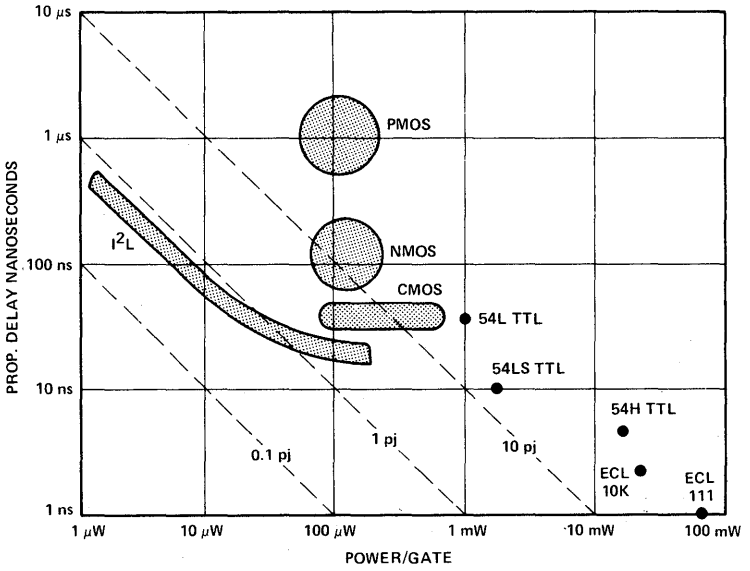


Figure 2

Speed and power capabilities of Various Logic Families

BASICS OF I^2L

The I^2L logic technology is designed around the basic I^2L gate circuit shown in Figure 3. The basic I^2L gate is a single-input, multiple-output inverter. The outputs of this gate can be directly cascaded with successive stages. Figure 4 shows a recommended symbol for the basic I^2L gate.

Most terminals of the I^2L gate share the same semiconductor region (for example, the collector of the PNP is the same as the base of the NPN and the emitter of the NPN is the same as the base of the PNP). This leads to a very compact device structure, and results in very high packing density in monolithic device fabrication. Figure 5 illustrates the basic device cross-section for a two-output I^2L gate. This basic structure can be made compatible with basic bipolar IC technology, by using a P-type silicon substrate as the semiconductor starting material. The bipolar compatible I^2L device structure is illustrated in Figure 6. Figure 7 gives a size comparison of the basic 4-output I^2L gate, with a 4-input T^2L gate, each fabricated with the same masking tolerances. Note that I^2L offers a 5:1 reduction in gate area.

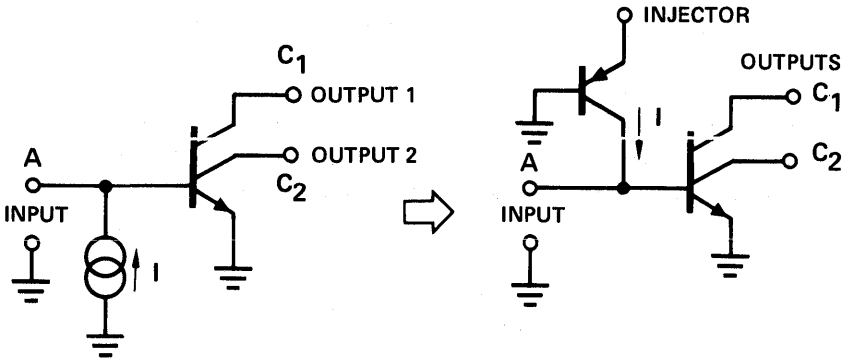


Figure 3
The Basic I^2L Gate Configuration

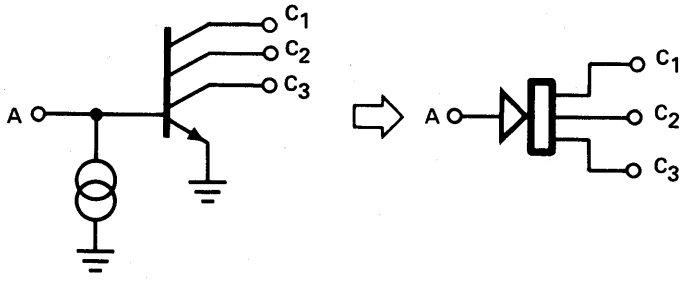
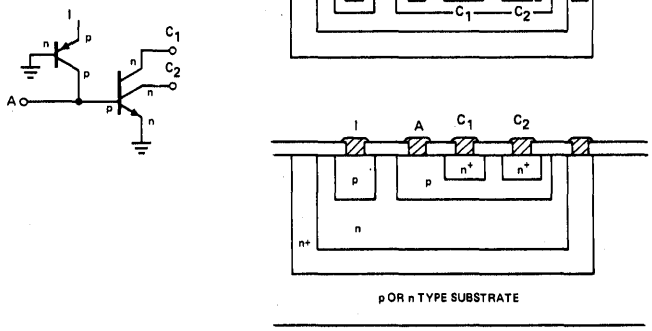


Figure 4
A Recommended Circuit Symbol for I^2L Gate

Figure 5

Structural Diagram
of Basic I^2L Gate



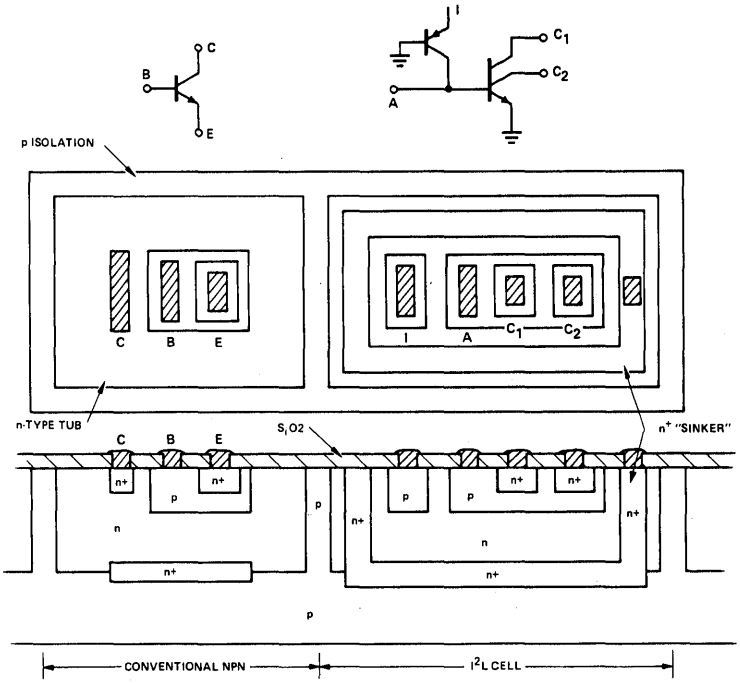


Figure 6

Bipolar Compatible
I^2L Gate Structure

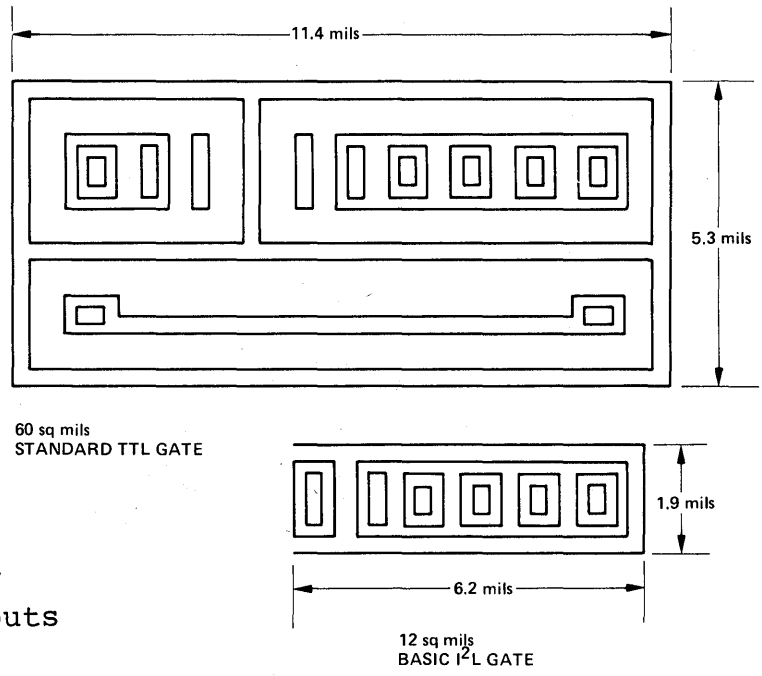


Figure 7

Area Comparison of
TTL and I^2L Gate Layouts

DESIGNING WITH I^2L

The biggest immediate application of I^2L is in the area of custom LSI, particularly for the system applications which require combination of complex analog and digital functions on the same chip. Since I^2L is a relatively new technology, it often requires the user to be familiar with the key features and capabilities of I^2L, as well as be aware of its limitations. To serve this purpose, Exar has developed a unique approach to the design of custom I^2L circuits. This design approach is based on the use of the Exar I^2L Design Kit, and the XR-400 I^2L Master Chip; and it greatly reduces the cost and the complexity of a custom I^2L LSI design. This is made possible by stocking I^2L silicon wafers that are completely built except for the final process step of device interconnection. These prefabricated wafers are then customized by application of two additional "masking" steps, one to open the contact windows on the chip, and one to form the desired metal interconnection pattern.

I^2L Design Kit:

The heart of Exar's custom I^2L program is the XR-400K I^2L Design Kit. This Design Kit is made up of 30 monolithic I^2L logic blocks and bipolar transistor arrays. These monolithic IC's or "kit parts" comprise the basic building blocks for complex analog or digital LSI systems. The XR-400K I^2L Design Kit also contains a comprehensive "Design and Applications Manual" which enables the kit user to breadboard his sytem with I^2L logic, and develop his own custom LSI circuit, at a fraction of the cost of a conventional full-custom IC development program.

The XR-400K I^2L Design Kit contains all the basic building blocks used on the digital systems design, such as I^2L gate arrays, flip-flops, counters and latches, along with T^2L/I^2L input and output interface circuits. The thirty monolithic IC's forming the I^2L Design Kit are packaged in 16-pin DIP packages. The kit is made up of seven different types of "kit parts", as listed in Table I, below:

Table I

Components Included in Exar's I^2L Kit

Part No.	Description	Quantity
XR-401	I^2L Inverter Array	10
XR-402	I^2L NOR Gate Array	4
XR-403	Dual R-S Latch	2
XR-404	Dual D-Type Flip-Flop	4
XR-405	TTL/I^2L Input Buffers	2
XR-406	I^2L/TTL Output Buffers	2
XR-407	I^2L Compatible NPN Array	4
XR-408	I^2L Compatible PNP Array	2
	Total:	30

The XR-400 I^2L Master Chip:

The XR-400 Master Chip contains a logic matrix made up of over 1,000 I^2L transistors and more than 200 conventional bipolar transistors and resistors. All of the logic building blocks, or kit parts, which comprise the Exar I^2L Design Kit are derived from the XR-400 Master Chip, with different metal interconnection patterns. Thus, the performance of the circuit components in the Exar I^2L Design Kit are virtually identical to those on the XR-400 Master Chip.

The XR-400 I^2L Master Chip is made up of three separate regions: (a) I^2L Gate Matrix; (b) Bipolar Input-Output Interface Section; (c) Linear Bipolar Section. Table II gives a list of total number of components available on the XR-400 I^2L Master Chip.

Figure 8 shows the basic layout of the XR-400 Chip. The three regions or sections of the chip are outlined and identified in the figure. A brief description of each of these three sections is given below.

(a) I^2L Gate Matrix

This region of XR-400 chip is made up of an array of 256 quad-output I^2L gates. It occupies the middle section of the XR-400 chip layout. These gates are arranged in 32 "cells" of 8-gates each, with low-resistivity diffused cross-unders between each "cell".

(b) I/O Interface Section

This region of XR-400 chip is made up of bipolar NPN and PNP transistor arrays and resistors specifically intended to serve as the input/output interface circuitry between the I^2L gate matrix and the circuit terminals (i.e., package pins or the chip bonding pads).

The bipolar transistors and resistors in this section of the chip can be interconnected to serve as either T^2L-to-I^2L or I^2L-to-T^2L interface buffers. In addition, these transistors and resistors can be utilized as the bias circuitry for the "injector" terminals of the I^2L gates in the "Gate Matrix" section.

(c) Linear Bipolar Section

This section of the XR-400 chip is located along the upper portion of the layout, shown in Figure 8. It is comprised of bipolar NPN and PNP transistors and resistors which are laid out in an "uncommitted" array form. They can be interconnected to serve as either linear or digital circuit blocks such as amplifiers, oscillators, comparators or clock generators.

TABLE II

Table of Components on
XR-400 I^2L Master Chip

Component Type	Quantity
Quad Output I^2L Gates	256
High-Freq. I^2L Gates	16
Input/Output Buffers	23
Bipolar Transistors	103
Resistors	120
Under-passes	210
Bonding Pads	40
Chip Size	110 x 110 mils

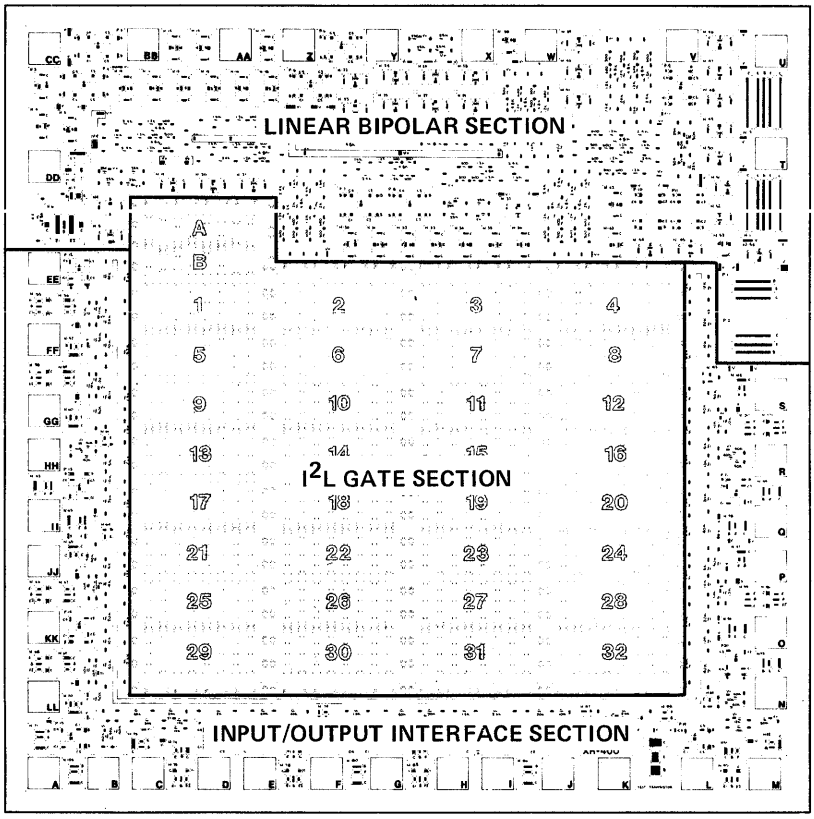


Figure 8

Basic Layout of XR-400 Master Chip

EXAR'S CUSTOM I^2L PROGRAM

Exar's custom I^2L program is based on the XR-400 Master Chip, and the XR-400K Design Kit. Figure 9 gives a simple flow chart of the basic development program. The I^2L Design Kit is used to breadboard and evaluate the circuit or system performance, prior to integration. After breadboarding with the kit, a decision is made to proceed with either a "semi-custom" or a "full-custom" design approach, depending on system complexity and production volume requirements.

Semi-Custom Design

In the semi-custom design approach, the monolithic custom IC is designed and fabricated by applying two custom "masks" to the prefabricated wafers of XR-400 Master Chip. These two masks customize the wafer by opening the specified "contact windows", and forming a custom metal interconnection pattern between the prefabricated components on the chip. The basic semi-custom development program involves 7 sequential steps. The first four of these steps is done by the customer, using the I^2L Design Kit, in consultation with Exar.

Step 1: Customer designs and breadboards his system using I^2L Design Kit.

Step 2: Customer, in consultation with Exar, evaluates design feasibility.

Step 3: Customer prepares circuit layout of his system on XR-400 Master Chip following basic instructions given in I^2L Design Kit manual. Layout is done simply by interconnecting appropriate device terminals with pencil lines on oversize drawings of XR-400 chips.

NOTE: As an option, Exar offers layout service at nominal charge.

Step 4: Customer submits his layout to Exar for final review.

NOTE: Steps 1, 2, 3, and 4 are done with no cost from Exar to customer. The formal part of the program is initiated at the completion of Step 4.

Step 5: Exar generates custom "contact" and "interconnection" patterns to be applied to prefabricated XR-400 wafers.

Step 6: Exar fabricates customized I^2L wafers from XR-400 chip, using tooling generated in Step 5.

Step 7: Exar assembles and delivers monolithic prototypes of custom IC to complete the program.

Some of the important facts about Exar's semi-custom I^2L development program are listed below.

Development Cost: Typical development costs for initial prototypes are in the range of $3,500 to $5,000.

Development Time: Typical development time to complete steps 5, 6, and 7 of development program is 8 weeks.

Additional Design Cycles: If additional design cycles are needed, the typical cost is $1,500 to $2,000 per design iteration, which includes the cost of additional prototypes.

Layout Services: Exar can perform the circuit layout (step 3) at a nominal charge of $1,000 to $2,000 depending on complexity.

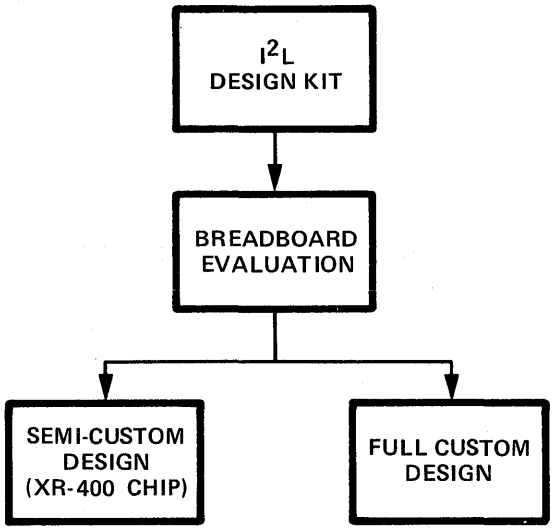


Figure 9

Flow Chart of Exar's Custom I^2L Program

Full Custom I^2L Design:

If the circuit complexity is in excess of the capability of XR-400 Master Chip, Exar can tool-up for a full-custom IC layout and design. In this case, the I^2L kit parts used in circuit breadboarding get used as logic "cells" in the circuit layout. Typical full-custom design and fabrication cycle takes about 16 to 20 weeks.

A Simple Design Example:

The 256 quad-output I^2L gates on the XR-400 Master Chip are arranged in 32 8-gate cells. A complex logic array can be partitioned into subsections, and each of these subsections can then be laid out in one or more of these "cells". Figure 10 shows a typical layout example for a toggle flip-flop on one of the 8-gate cells of the I^2L Master Chip. The logic diagram of the flip-flop is shown in Figure 10(a), in terms of its NAND gate implementation. Figure 10(b) shows the layout of the same flip-flop on the XR-400 Master Chip. Note that, in the cell layout, the circles correspond to the outputs. When the interconnection pattern is drawn, only the contact windows under the "darkened" contact areas of the layout are opened, to activate the gates. In this manner, metal interconnections can be routed over the unused parts of the gates. With this approach at least 80% of the available I^2L gates on the XR-400 Master Chip can be interconnected with a single metal interconnection layer.

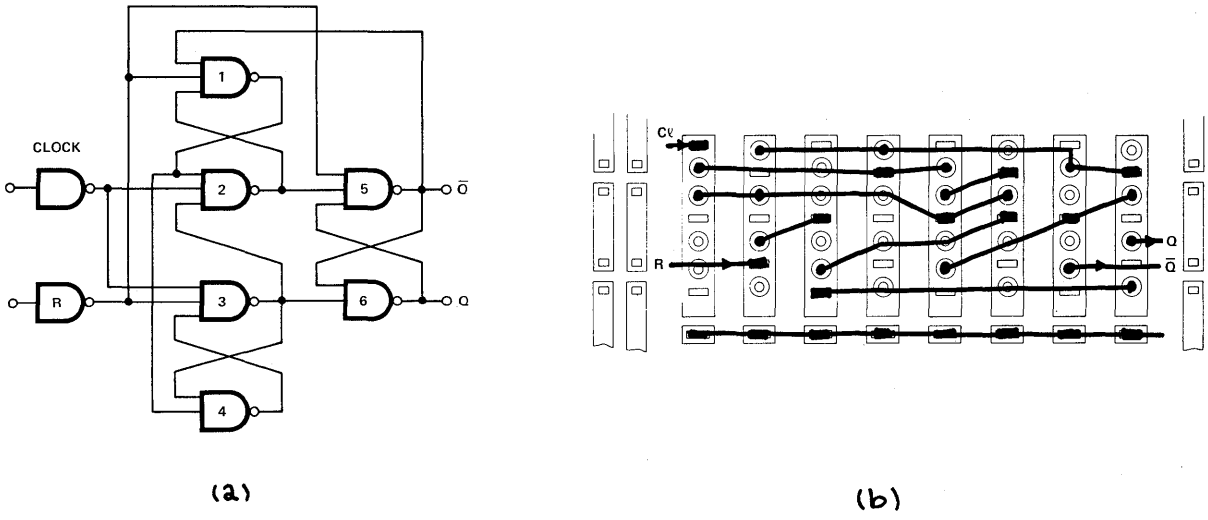


Figure 10

Logic Diagram of a Toggle Flip-Flop
and Its Layout on XR-400 Master Chip

9. A LOW COST 4 1/2-DIGIT A/D CONVERTER

BY LEE EVANS AND DAVE FULLAGAR
Intersil
Cupertino, CA 95014

INTRODUCTION

The dual-slope integrating A/D converter has long been recognized as the most attractive conversion technique where high accuracy is the prime requirement. Although relatively slow, a dual-slope converter has high noise immunity and demands only one close tolerance component -- a current or voltage reference. All the other components in the system are non-critical and inexpensive.

Despite the advantages mentioned above, the dual-slope converter does represent a considerable design challenge. If put together from readily available components such as FET switches, operational amplifiers, and logic circuits, a host of problems will be encountered. In all probability, one will end up with a circuit which may or may not perform as desired, which will cost $30 to $100 to build, and which will most certainly have required more than nine man-months of engineering effort.

An alternative approach is to use one of the monolithic or two-chip circuits which have become available in the last couple of years. However, these are of relatively low performance and very inflexible. Provided the limited performance is acceptable, and provided the system configuration fits in with the requirements, they may do the job -- albeit at greater cost than the circuits to be described.

Clearly, there is a need for an alternative approach -- one which provides the user with flexibility and high performance without demanding a substantial development effort. This is the thinking which lies behind Intersil's recent introduction of a family of five dual-slope building blocks.

A New Approach

The key component in the new converter system is an analog signal conditioner, the 8052. This monolithic chip contains all the linear components required in a high performance converter, i.e., a FET-input buffer, an integrator, a comparator, and a voltage reference. This circuit is designed to interface with any one of four purpose-built digital ICs:

1. The 8053 - consists of a 6-switch array with switch drivers. This circuit, when combined with

the 8052, provides the maximum degree of flexibility, but requires additional logic to complete the system.

2. The 7101 - This chip contains all the additional logic circuitry (including the hex switch array) required to build a complete 3 1/2-digit voltage converter. The scale can be selected to be ±2.000 or 0.2000V, with auto-zero and auto-polarity. Output is parallel BCD.

3. The 7103 - for 4 1/2-digit applications, with multiplexed BCD output. Other features as for 7101.

4. The 7104 - 16-bit logic system providing binary output for inputs of ±3.2768V (16-bits in 0.1mV increments). For use with digital processors.

As well as providing high performance, these circuits are low cost. The 8052/8053 pair, for example, is priced at $10 (1000 pieces) and a complete 4 1/2-digit converter can be built for the cost of most 3 1/2-digit designs.

A reliability advantage is also realized by this approach since the system component count is low and interfacing difficulties are minimized.

The dedicated circuits (7101, 7103 and 7104) will be the subject of another application note; the remainder of this note will be concerned with describing specific applications for the 8052/8053 combination. However, before plunging into circuit details for complete converter systems, it is important to review some fundamental principles.

A Review of Basic Principles

A dual-slope conversion can usually be divided into three distinct phases (See Figure 1):

1. Auto-zero phase: during this time, the errors in the analog components (offset voltage, etc.) will be automatically nulled out by grounding the input and closing the feedback loop such that error information is stored on the auto-zero capacitor. The manner in which this is done with the 8052/8053 will be described later.

2. Signal Integration phase: the input signal is integrated for a fixed number of clock pulses. For a 3 1/2-digit converter, 1000 to 2000 pulses is typical; for 4 1/2-digits, 20,000 is typical. On completion of the integration period, the

voltage V' in Figure 2 is directly proportional to the input signal.

3. Reference Integration phase: at the beginning of this phase, the integrator input is switched from V_{IN} to V_{REF}. The polarity of the reference is such that the integrator discharges back towards zero. The number of clock pulses counted between the beginning of this cycle and the time when the integrator output passes through zero is a digital measure of the magnitude of V_{IN}. This can be seen by studying Figure 2. For display applications, the counter output is converted to BCD format.

In practice, a number of difficulties must be overcome in a real-life circuit. Firstly, in order to handle both positive and negative inputs, the circuit must be able to sense the polarity of the input. Then the appropriate reference (positive or negative) can be applied. If the wrong polarity reference is applied, then the integrator will continue to move away from ground during the third phase of the conversion. Similarly, great care must be exercised for close-to-zero inputs; whatever circuitry is used to detect the polarity of the input (in order to apply the correct polarity reference) must be immune to false triggering. Ohterwise, there is, once again, the danger that the integrator will not return to zero during the final phase of the conversion. One well-known company's first venture into the panel meter field overlooked this problem: for inputs close to zero, the output would overrange!

Secondly, the charge injection of the switches must be considered. For example, the switch used to change the integrator input from V_{REF} to V_{IN} causes some erroneous charge to be dumped into the integration capacitor. This must be kept to an absolute minimum to maintain conversion accuracy.

Other factors which demand consideration include the input current and noise characteristics of the buffer, the leakage of the switches, and the speed of the switches vis á vis the clock frequency.

THE 8052/8053 SYSTEM

In the 8052/8053 pair, solutions to the problems discussed above have been engineered into the two chips. They are intended to interface with one another and eliminate all the critical or tricky parts of a dual-slope converter design. At the same time, the user retains full control over the output format, the design of which is far less critical. Several designs for complete converters are given in the application notes which follow.

The 8052 and 8053 functional diagrams are shown in Figure 3. A summary of the key specifications for the two circuits is given in Tables 1 and 2.

	8052	8052A
Buffer and Integrator Amplifiers		
Input Offset Voltage (max.)	50mV	50mV
Input Current (max.)	50pA	10pA
CMRR (min.)	70dB	70dB
Non-linear component of CMRR	110dB	110dB
Voltage Gain (min.)	20,000	20,000
Comparator		
Voltage Gain	4,000	4,000
Voltage Reference		
Output Voltage	1.85V	1.85V
Temperature Coefficient	40ppm/°C	20ppm/°C

TABLE 1: 8052 Specifications Summary

	8053	8053A
Switch ON-resistance (max.)	2500 ohms	2500 ohms
Total Leakage: SW 1,2,5,6 (max.)	30pA	10pA
SW 3,4 (max.)	30pA	10pA

TABLE 2: 8053 Specifications Summary

How They Work

The basic principles of the dual-slope technique have already been reviewed; now let's look at the 8052/8053 in detail. Figure 4 shows a functional diagram for a complete A/D converter. In Figures 5 through 7, this functional diagram has been greatly simplified to illustrate the switch position during the three phases of a conversion:

1. Auto-zero phase: (state 00 on state flip-flop)
   Referring to Figure 5, switches 1, 2, and 3 are ON. The input to the capacitor C_1 is grounded; the voltage across it is V_{REF}. The negative feedback loop through Sw3 causes a voltage to be applied to the non-inverting integrator input such that the integrator output is held at about -1.2V.
2. Signal Integration phase: (state 01)
   Switches 1, 2, and 3 are now opened; Sw4 is closed (Figure 6). If V_{IN} is equal to ground, the integrator output will not change. But if V_{IN} does not equal zero, the integrator will generate a ramp whose slope is proportional to V_{IN}. The duration of the integration will be a given number of clock pulses, as explained earlier. During this period, the auto-zero capacitor holds the error information derived during the auto-zero phase of the conversion. At the end of phase two, the sign of the ramp is latched into the polarity flip-flop.
3. Reference Integration phase: (states 10 and 11)
   In the final phase of the conversion, the switch driver decoder uses the output of the polarity flip-flop to decide whether to close Sw5 or Sw6. If the input was

positive, Sw6 is closed and a voltage which is one V_{REF} more negative than during auto-zero is impressed on the buffer input (Figure 7a). If the input was negative, Sw5 is closed and a voltage which is V_{REF} more positive than during auto-zero is impressed on the buffer input (Figure 7b). The reference capacitor C_1 generates the equivalent of a +ve or -ve reference from a single voltage source with negligible error. The reference returns the output of the integrator to zero. The time, or number of counts, required to do this is proportional to the input voltage. Since the reference cycle can be twice as long as the signal integrate cycle, the input voltage required to give full scale reading = $2V_{REF}$.

The circuit, as described to this point, is not new; it has been used successfully for several years. However, the 8052/8053 has made three major contributions to the accuracy of this circuit. These are: low charge injection, monolithic FET-input amplifiers, and the use of a zero crossing flip-flop.

Achieving Low Charge Injection

During auto-zero, there is no problem in charging the capacitors to the correct voltage. The problem is getting the switches off without changing this voltage. As the gate is driven off, the gate-to-drain capacitance of the switch injects a charge on the reference or auto-zero capacitor, changing its voltage. A

designer using discrete components is forced into critical board layouts, where charges of opposite polarity are injected to compensate or neutralize the driver injection. This balance will be upset by any unit-to-unit variation of switch capacitance so, at best, the final design is a compromise. In the 8052/8053, the critical layout has been done on the semiconductor chip and need not concern the user. Also, since a silicon-gate process is used for the switches, the unit-to-unit variation is extremely low. The net result is an error due to charge injection that is so low it is difficult to measure--less than 5uV referred to the input.

The Advantages of J-FET Op-amps

Both the buffer and integrator use junction FET inputs in a guarded circuit that reduces the voltage across the FET to 3 or 4 volts. At this voltage level, input leakage currents of 1pA are typical. For typical component values, 1pA leakage. contributes less than 1uV of error to the circuit. In theory, MOS FETs would contribute less leakage, but their increased noise would swamp out any improvement by orders of magnitude.

Purpose of the Zero-Crossing Flip-Flop

The problem that the zero-crossing flip-flop is designed to solve is shown in Figure 8. The integrator output is approaching the zero-crossing point where the clock will be stopped and the reading displayed. The pulses superimposed upon this ramp

will cause a false reading by stopping the count prematurely. For a 40,000 count instrument, the ramp is changing approximately .25mV per clock pulse (10 volts maximum integrator output divided by 40,000 counts). The clock pulses have to be less than 100uV peak to not cause significant errors. The circuit layout to achieve this can be time-consuming at best and impossible at worst.

The suggested circuit gets around this problem by feeding the zero-crossing information into a J-K flip-flop instead of using it directly; this can be seen by referring back to Figure 4. The flip-flop interrogates the data once every clock pulse after the transients of the previous clock pulse and half clock pulse have died down. Any false zero-crossings caused by clock pulses are not recognized. Of course, the flip-flop delays the true zero-crossing by one count in every instance. If a correction was not made, the display would always be one count too high. The correction is to change the four states of the converter one count early. In other words, instead of changing states at the beginning of count 0000, the states are changed at the beginning of count 9999. Since this pulse is always available as "carry" from a synchronous counter, no extra decoding is required. A bonus feature of this circuit is that latching the counter output becomes very simple with no potential race condition existing. The designer has one complete clock pulse to transfer the counter data to the latches and decouple them before a false reading will occur. The

timing diagram for a signal $\simeq$ 0 volts is shown in Figure 9.

BUILD THESE CIRCUITS

One of the significant advantages of the 8052/8053 system is the design flexibility offered by the "building block" approach. This feature will be illustrated by the circuits which follow. They have all been designed to provide a problem-free solution to a particular conversion requirement.

A. A Versatile 4½-digit Converter

Figure 10 shows the complete circuit for a 4½-digit (±2.000V full scale) A-D with LED readout and parallel BCD lines. In addition to the 8052/8053, this circuit uses 6 low-cost CMOS packaged for control and 5 TIL 306 as a combination LED readout, synchronous counter, and BCD latch. In this circuit, the clock runs continuously driving the 5 decade counters in the TIL 306's. The carry from the fourth decade is used to trigger the state F-F. Thus, each of the four states lasts for 10,000 counts. At the beginning of state 10, the fifth decade is cleared. None of the other counters need to be cleared since they automatically roll to 0000 at this point. When the zero-crossing F-F detects the end of the measurement, a latch pulse is initiated.. The R-C time constant of this pulse is selected long enough (50nSec) to assure the latches turn on, but short enough (3uSec)

to assure that the latches are decoupled before the next clock pulse. Selecting a typical time constant of 400nSec assures proper latching with wide variance in component value.

In order to give a visual indication of overload, the LED displays are blanked during state 00 if an overload exists. If overloaded, the instrument will blink a reading of 19999. A non-blinking reading of 19999 is a valid reading for the instrument.

By tying the clear terminals of the state flip-flop and the four decade counters to a common bus, the instrument can be synchronized to external events. If the bus is low, the instrument is held in auto-zero with the last measurement cycle at the beginning of state 00. The data valid pulse indicates the end of measurement cycle. For free-running condition, the bus is held high at +5 volts.

B. Generating a Family of A-D Converters

In Figure 10 the lines marked "MBS" and "MSB-1" are connected to Q_B and Q_A of the 4-bit state flip-flop respectively. This forces a change in state for each carry pulse (10,000 counts) from the decade counters. If the lines were moved to Q_C and Q_B respectively, two carry pulses (20,000 counts) would be required to change states. Since full scale is two states long,

the maximum count now becomes 40,000 (actually 39,999). Similarly, if Q_D and Q_C are used, the maximum count is now 7,999 (one less decade counter would be used in this case). The ability to easily change maximum count (full scale) is most useful where the A-D converter is measuring physical constants such as temperatures, distances, weights, etc. It allows the designer to match the digital reading of the instrument to the analog range of the transducer. Since the analog input required to generate full scale output is $2V_{REF}$ in every case, an almost endless variety of scale factors can be generated easily from one basic design. Table 3 summarizes how the family of DVM's is generated.

Full Scale	V_{REF}	Total Number Of Decade Counters	Connect MSB-1 to	Connect MSB to
±2000.0mV	+.1000V	4	Q_A	Q_B
±2.000V	+1.000V	4	Q_A	Q_B
±400.0mV	+.2000V	4	Q_B	Q_C
±4.000V	+2.000V	4	Q_B	Q_C
±800.0mV	+.4000V	4	Q_C	Q_D
±2.0000V	+1.0000V	5	Q_A	Q_B
±4.0000V	+2.0000V	5	Q_B	Q_C
±3.2768V	+1.6384V	4*	Q_C	Q_D

*Number of 4-bit binary counters

TABLE 3

Specific circuits demonstrating this principle are shown in

Figures 10 and 11. An 800mV full scale A/D can be obtained from the 2.0000V instrument shown in Figure 10 with the three following modifications:

1. Delete middle LED counter.
2. State decode moved to Q_D and Q_C.
3. Reference voltage adjusted to 0.4000V.

Figure 11 is the specific circuit for a 16-bit binary A-D. Here the decade counters and displays have been replaced by synchronous 4-bit counters and latches. To give a full scale reading of ±3.2768 volts, the reference is adjusted to 1.6384 volts.

Figure 12 shows the circuit for a 40,000 count instrument. This circuit conforms to all of the "family" rules with the exception that it uses a -2.0000 volt reference. If a positive reference was used, Pin 3 of the 8053 would have to swing to +6V (+4 volt input +2 volt reference). Since this exceeds the +5 volt supply, the switch would forward bias into the substrate. It can easily accommodate the +2 to -6 volt swing required of a negative reference. The only change required by a negative reference is that the drive to Pin 6 (+ reference driver) and Pin 10 (- reference driver) be interchanged. Also, since the internal reference is not used, no connections are made to Pins 3, 6, and 7 of the 8052.

C. Achieving Lowest Cost

In a 4½-digit (20,000 count) instrument where the family generating capabilities of the four-bit counter is not required, a dual D flip-flop can be substituted for this function with some reduction in parts costs. Also, a "±1" LED, driven by a dual D flip-flop, can replace the fifth TIL 306. Figure 13 shows a circuit with these two substitutions made. (Note Figure 13 is titled 4½-digit DVM 8052/8053, 20,000 count with Parallel BCD.)

If the parallel BCD capabilities of the TIL 306 are not required, a further reduction in parts cost can be achieved by using the circuit of Figure 14. In this circuit, the MM74C926 performs the counting, latch, and 7-segment decode function of the TIL 306 such that it can be used with any LED displays. Some modification of the clock and latch circuit is required since the 74C926 uses a ripple counter with a carry at 0000 instead of a synchronous carry at 9999. When a zero-crossing signal is detected and the latch-enable is initiated, a signal is simultaneously fed to the clock drive circuitry to delay the clock, and therefore the count, until the previous count can be latched. The latch time-constant is shorter than the clock-delay time-constant to assure that the latch is transferred and disabled before the clock resumes counting. A luS

time delay in the output of the clock driver assures that the slight delay (100nS) between the clock pulse and the clock-delay pulse does not clock the counter. Blinking is provided to give a visual indication of overload. However, the display will flash .0000 instead of 1.9999 due to the nature of the ripple counter.

D. Capacitor Selection

The reference capacitor and auto-zero capacitor are each shown as 1.0uF in the applications schematics. These relatively large values are selected to give greater immunity to PC board leakage since much smaller capacitors are adequate for charge injection errors or leakage errors from the 8052/8053. The ratio of integrating resistor and capacitor is selected to give 9 volt swing for full scale inputs. This is a compromise between possibly saturating the integrator (at ±14V) due to tolerance build-up between the resistor, capacitor, and clock and the errors a lower voltage swing could induce due to offsets referred to the output of the comparator (see discussion below). Again, the .22uF value for the integrating capacitor is selected for PC considerations alone since the very small leakage at the integrator input is nulled at auto-zero.

A very important characteristics of the integrating capacitor is low dielectric absorption. A polypropylene capacitor made by TRW gave excellent results in the

application. In fact, a good test for dielectric absorption is to test the subject capacitor in this circuit with the input tied to reference. This ratiometric condition should read 1.0000 and any deviation is probably due to dielectric absorption. In this test, polycarbonate capacitors typically read .9992; polystyrene, .9997; and polypropylene, 1.0000. The increased temperature coefficient of the polypropylene is of no consequence in this circuit. The dielectric absorption of the reference capacitor and auto-zero capacitor are only important at power-on or when the circuit is recovering from an overload. Thus, smaller or cheaper capacitors can be used here if accurate readings are not required for the first few seconds of recovery.

E. The Reference

A stable reference is the most important single component in a dual-slope converter. Three factors should be considered when selecting a reference: temperature coefficient, intended range of operation, and the accuracy of conversion.

The scale factor temperature coefficient for the 8052/8053 pair is typically 3ppm/^{o}C. This is the error term which ultimately limits the temperature performance. For references with performance more poor than this, accuracy as a function of temperature will be limited by the reference.

The reference provided on the 8052 chip has a typical temperature coefficient of 20 and 40ppm/°C respectively for the 8052 and the 8052A. Maximum values are about 100ppm/°C; contact Product Marketing for further information.

F. Other Design Features

In the designs shown, the output of the comparator is clamped to the +5V supply to prevent the positive swing of the comparator from forward biasing the auto-zero switch to its substrate and injecting minority carriers that would be collected as leakage currents. In addition, a voltage translation network connects the output of the comparator to the auto-zero switch. The purpose of this network is to assure that during auto-zero the output of the comparator is at or near the threshold of the CMOS logic (+2.5V) while the auto-zero capacitor is being charged to V_{REF} (+1V in the case of 2.0000V instruments). Otherwise, even with zero signal in, some reference integrate period would be required to drive the comparator output to the threshold region. This would show up as an equivalent offset error. Once the divider chain has been selected, the unit-to-unit variation should contribute less than a few tenths of a count error in the worst case (40,000 count instrument) and proportionately less in other instruments. For a 3½-digit

instrument, the error is unmeasurable.

Finally, the back-to-back diodes are used to help noise. In the normal operating mode they offer a high impedance and long integrating time constant to any noise pulses charging the auto-zero capacitor. At start-up or recovery from an overload, their impedance is low to large signals so the capacitor can be charged in one auto-zero cycle.

The maximum conversion rate of most dual-slope A/D converters is limited by the frequency response of the comparator. Even though the comparator in this circuit is all NPN with an open-loop gain-bandwidth product of 300MHz, it is no exception. The comparator output follows the integrator ramp with a 3uS delay. At a clock frequency of 160KHz (6uS period), half of the first reference integrate period is lost in delay. This means that the meter reading will change from 0 to 1 with 50uV in, 1 to 2 with 150uV, 2 to 3 at 250uV, etc. This transition at mid-points is considered desirable by most users. However, if the clock frequency is increased appreciably above this, the instrument will flash 1 on noise peaks even when the input is shorted.

Some circuits use positive feedback or a latch to solve the problem. However, unless the comparator voltage swing, the comparator gain, and the integrator gain are

carefully controlled, this circuit can generate anticipation errors that greatly exceed the 3uS delay error. Also, it is very susceptible to noise spikes. A more controlled approach for extending the conversion rate is the use of a small resistor in the integrator feedback loop. This feeds a small pulse to the comparator to get it moving quickly and partially compensate for its delay.

The minimum clock frequency is established by leakage on the auto-zero and reference capacitor. With most devices measurement cycles as long as 10 seconds gave no measurable leakage error.

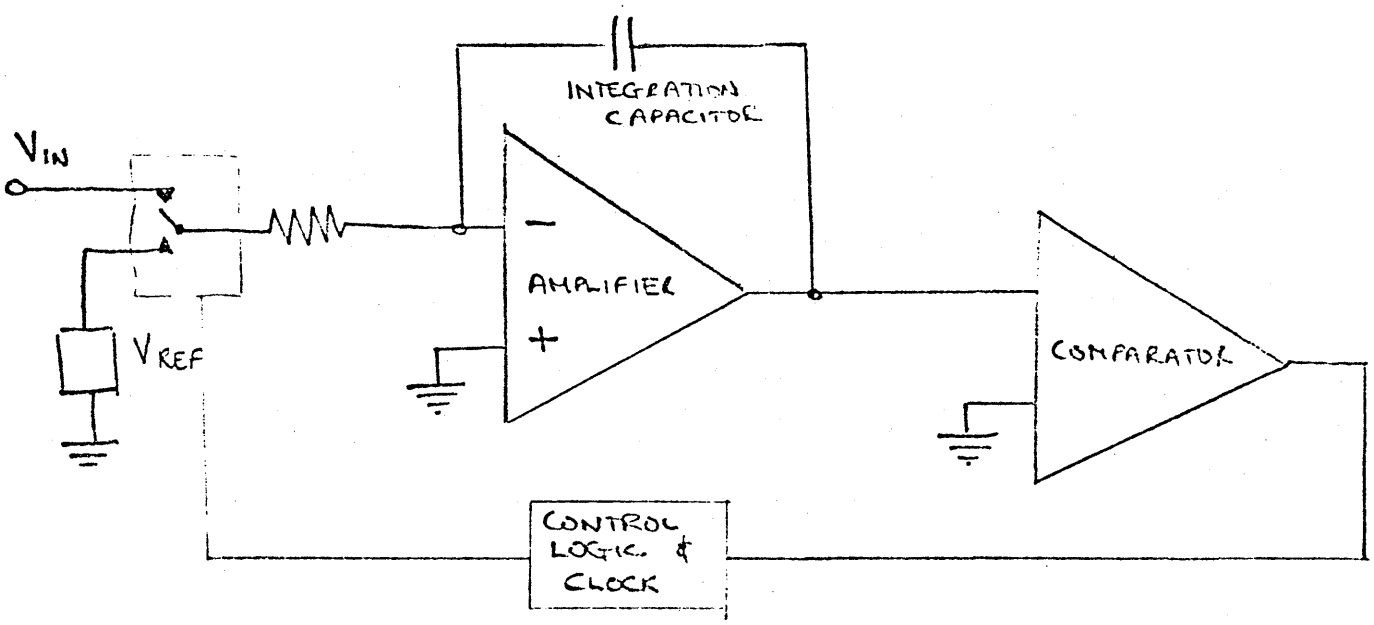


Fig. 1: Simplified Dual Slope Converter

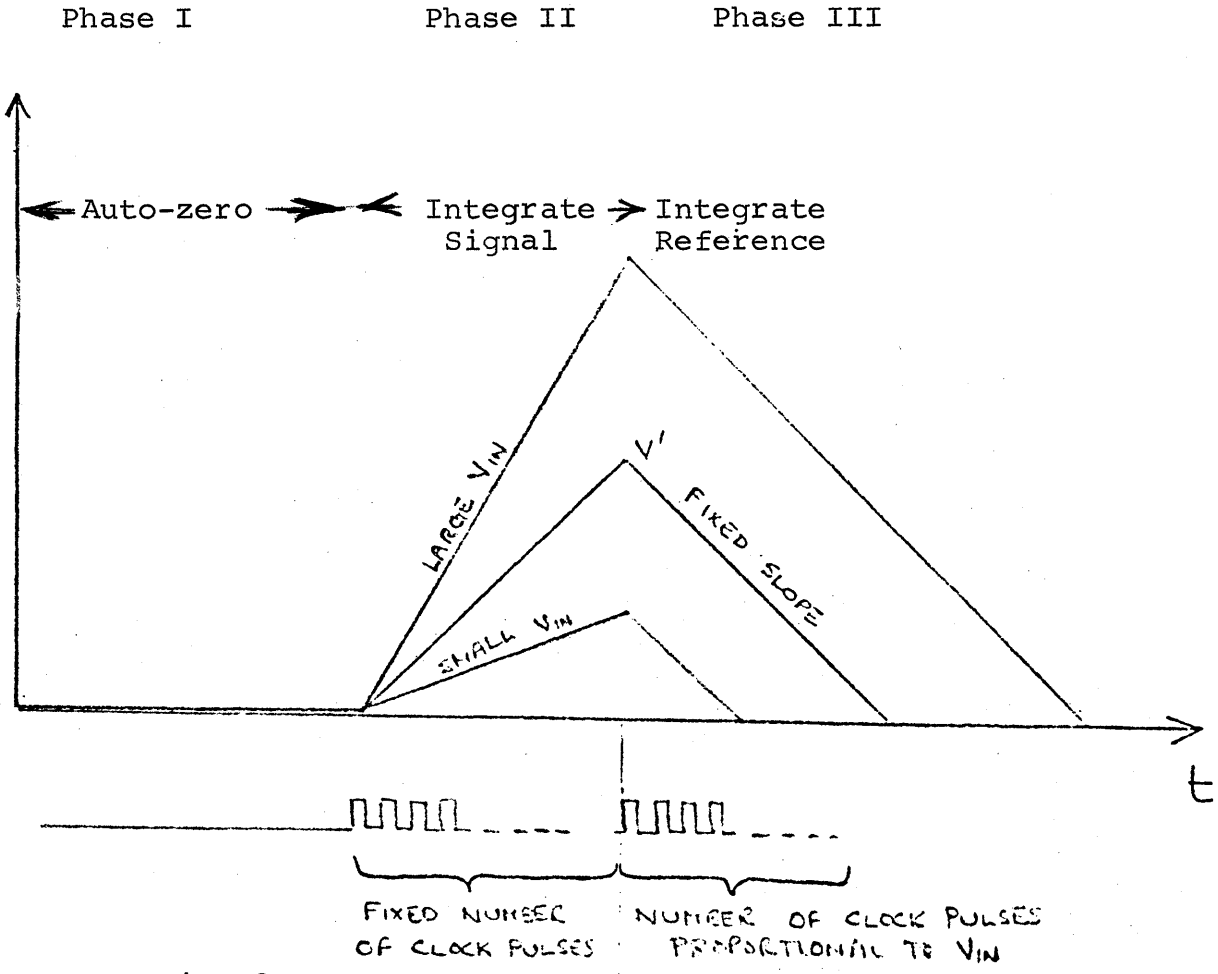


Fig. 2: Dual Slope Converter Wave Forms

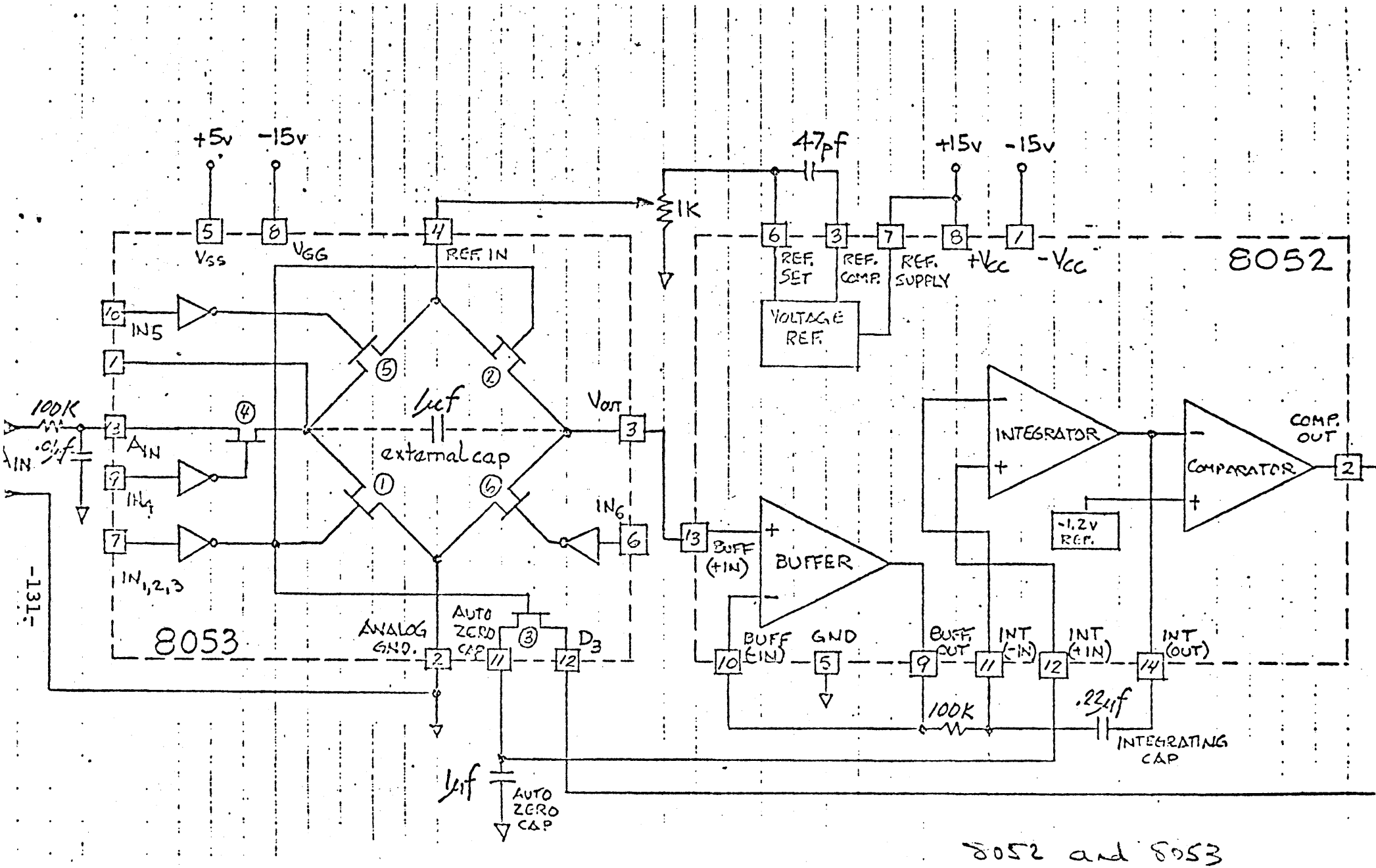


Fig. 3: Functional Diagram for A/D Converter

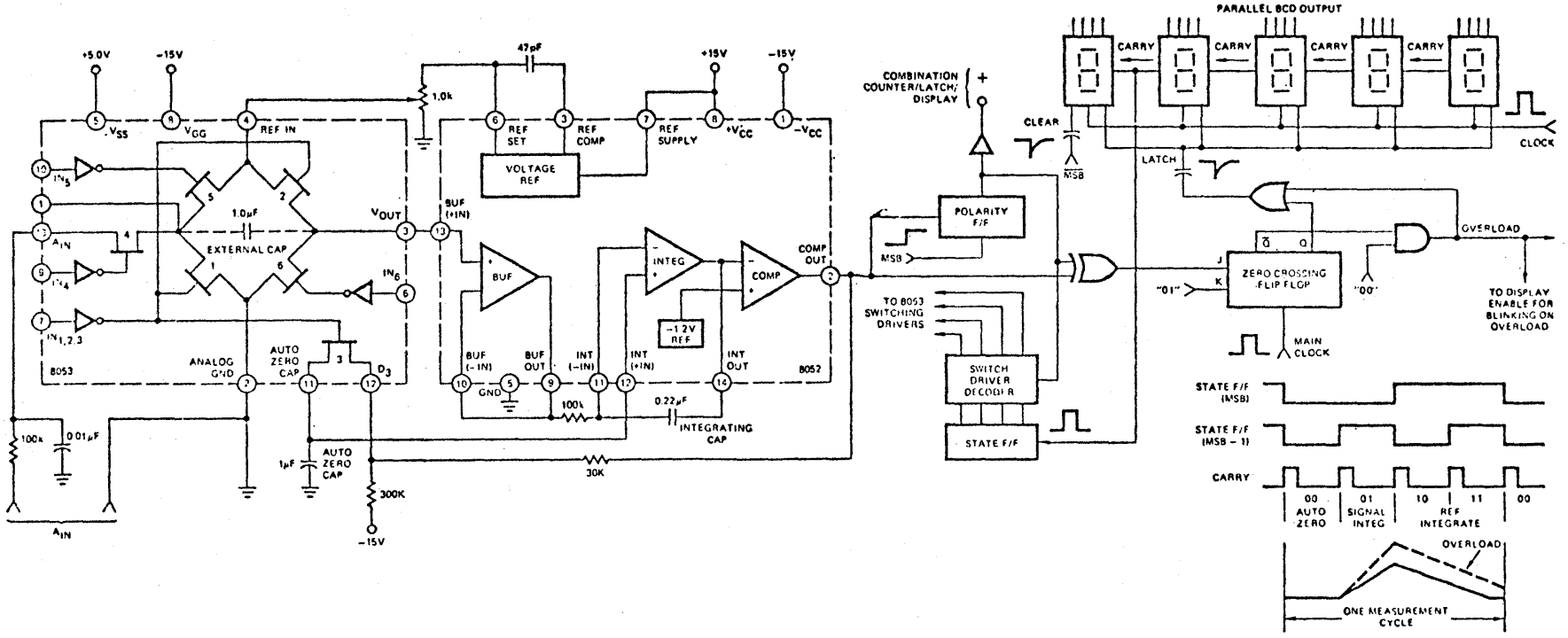


Fig. 4: Functional Diagram for A/C Converter

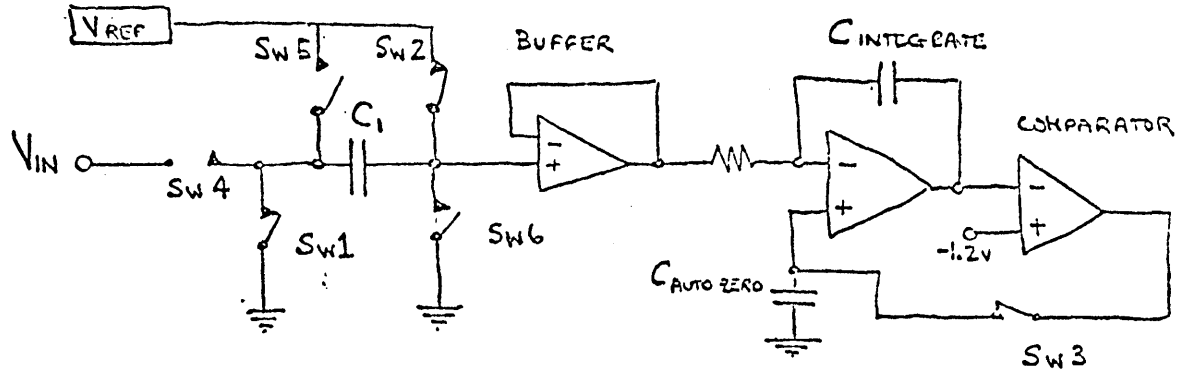


Fig. 5: Auto Zero

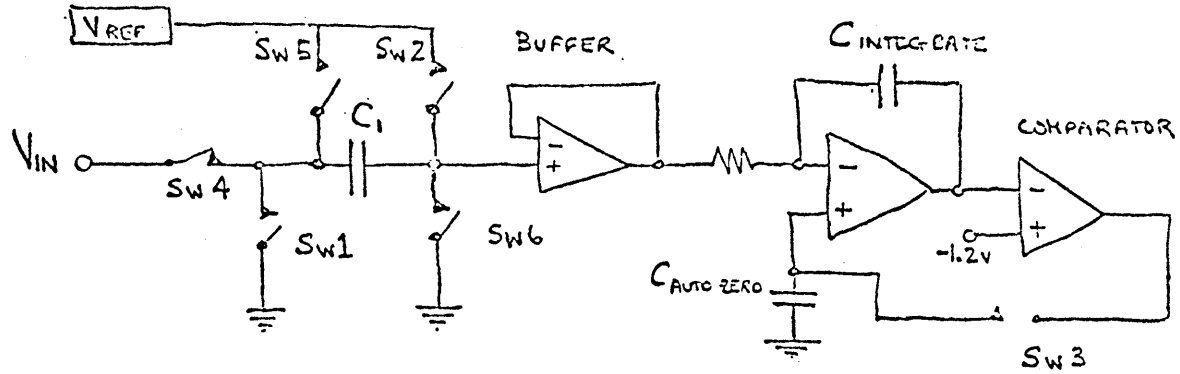


Fig. 6: Integrate Signal

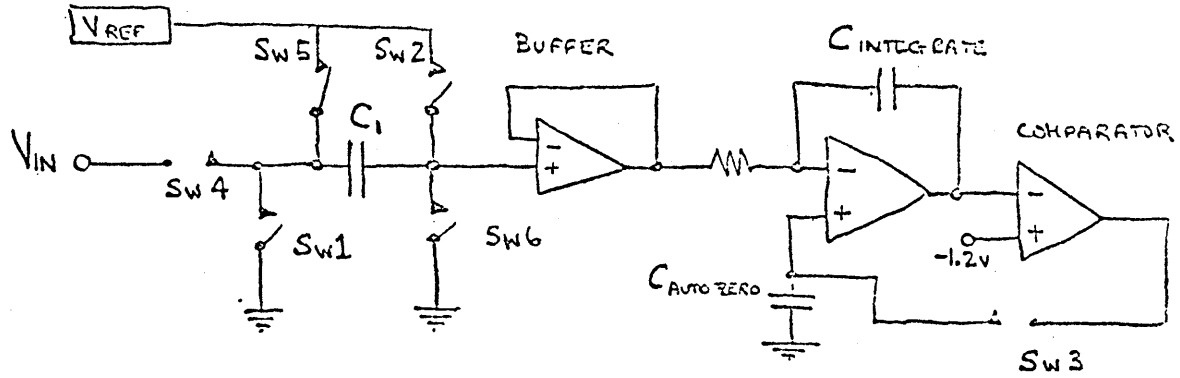


Fig. 7a: Integrate REF ($+v_c$ l/p)

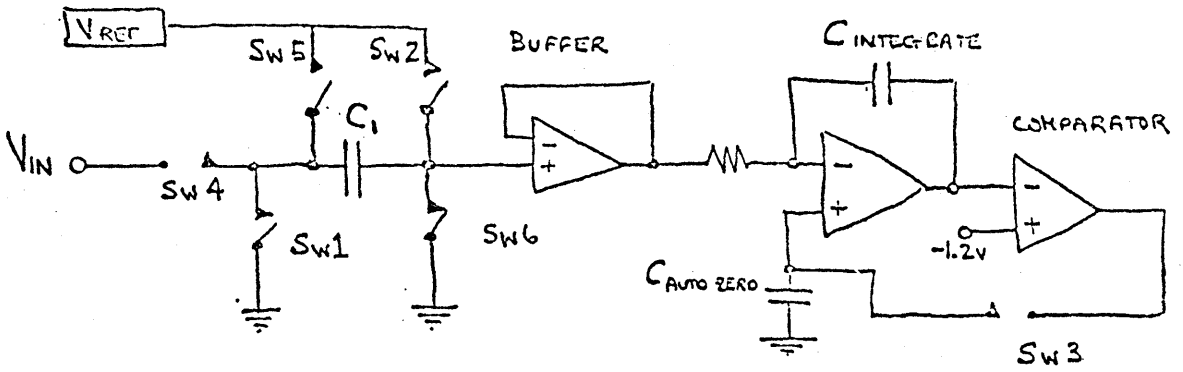


Fig. 7b: Integrate REF ($-v_c$ l/p)

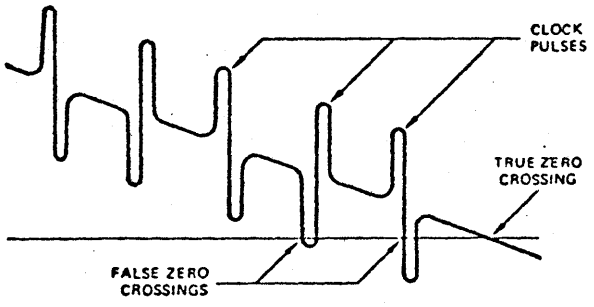


FIGURE 8
INTEGRATOR OUTPUT NEAR ZERO-CROSSING.

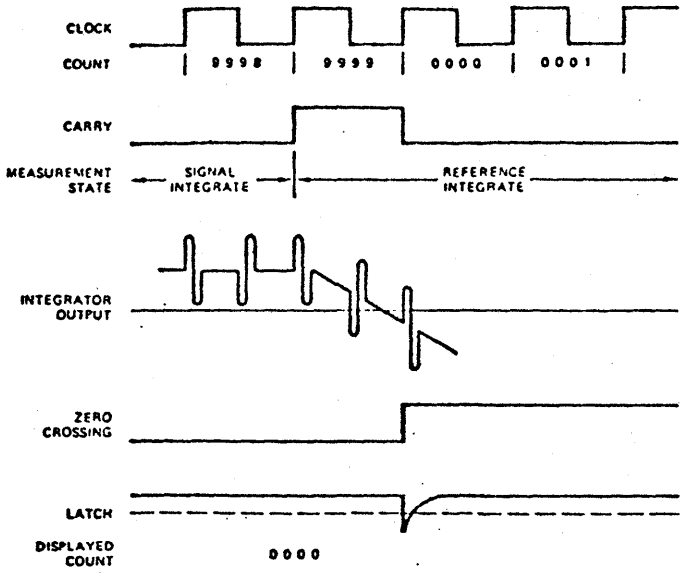


FIGURE 9

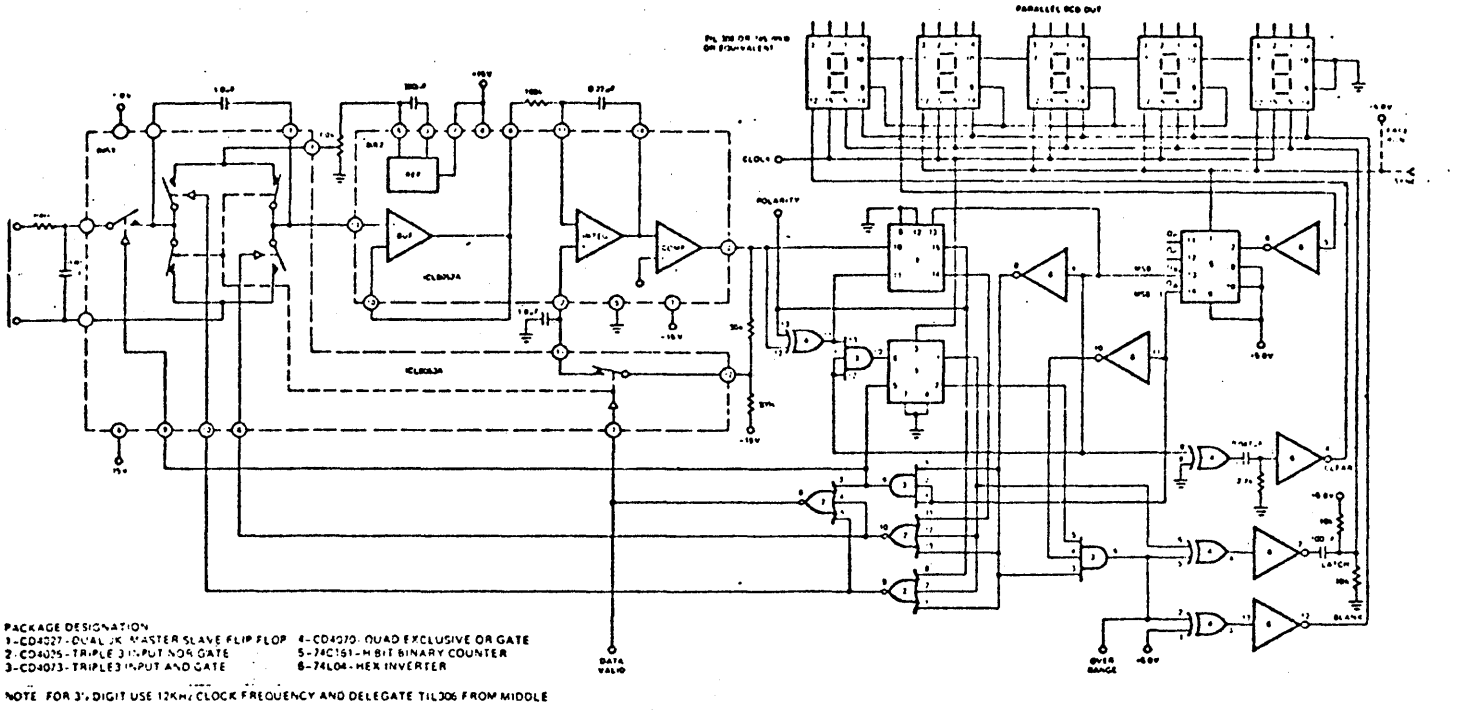


Figure 10: General Circuit For A Family of DVM's.

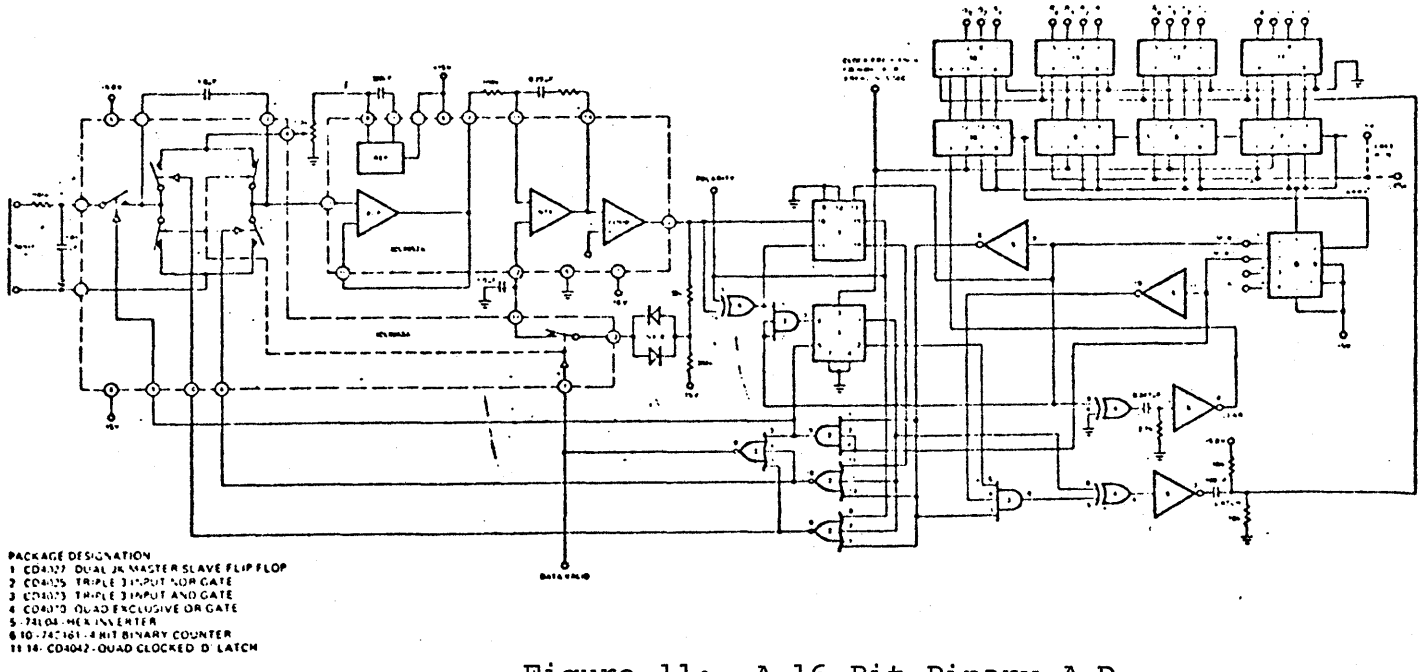


Figure 11: A 16 Bit Binary A-D

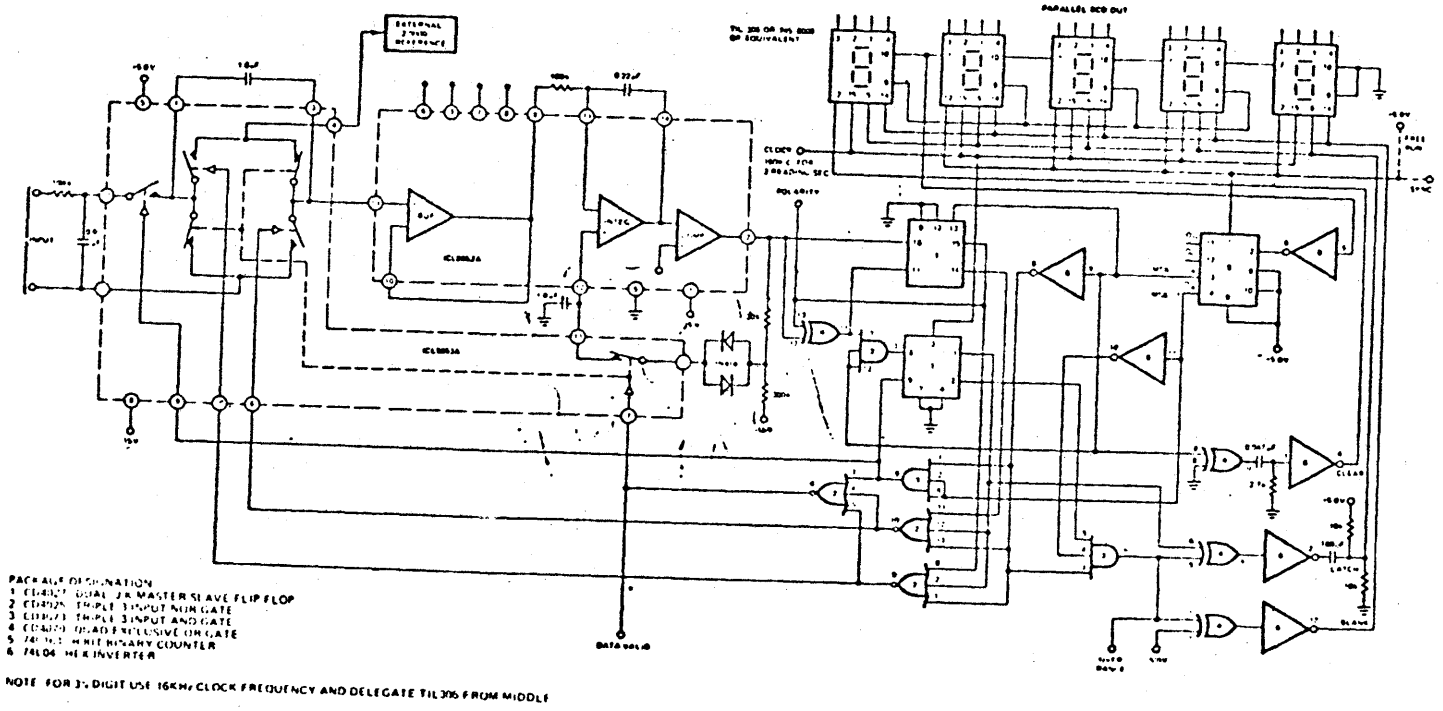


Figure 12: 40,000 Count DVM

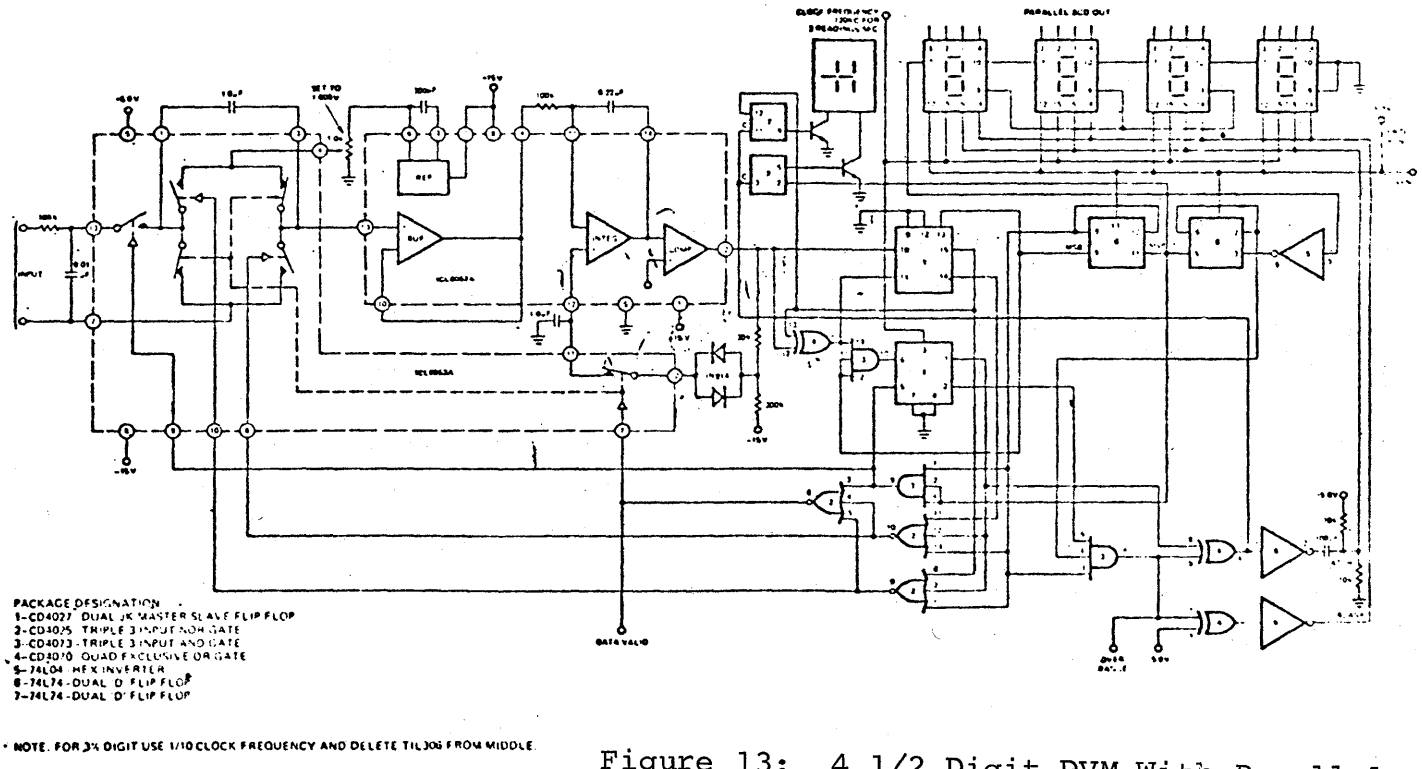


Figure 13: 4 1/2 Digit DVM With Parallel BCD

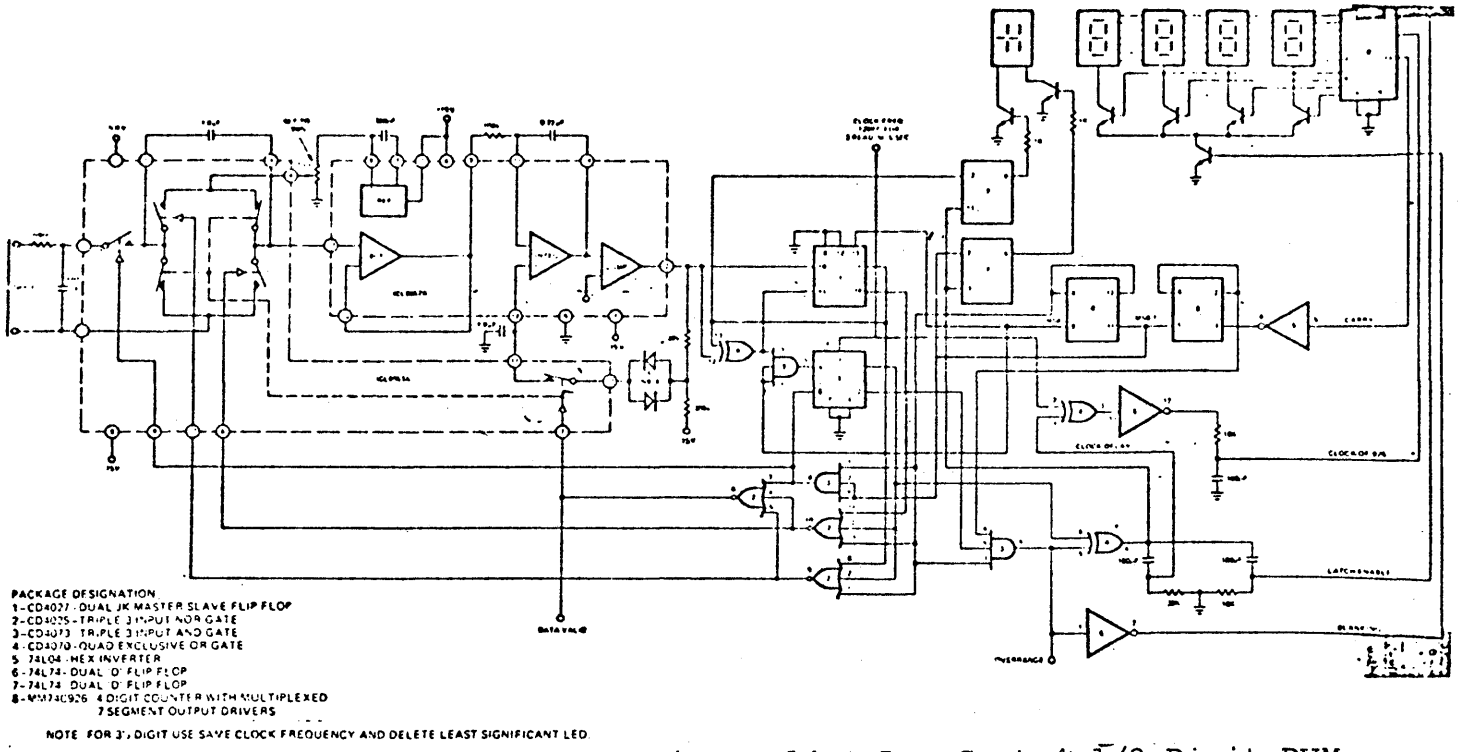


Figure 14: Low Cost 4 1/2-Digit DVM

10. A Logic Compatible High Current Switch

Marvin K. Vander Kooi
Applications Manager
Siliconix Incorporated
Santa Clara, California

The new VMOS technology high current logic compatible MOSPOWER™ FET switch VMP-1 offers a very significant breakthrough in solid state switching. The VMP-1 will replace power Darlington bipolars due not only to its' ease of use, but also because it can switch ampere level currents 200 times faster than the minority carrier storage type bipolar devices. In the following pages we will take a look at VMOS features, structures, pertinent transfer curves and specifications, and a fairly large number of application examples.

The VMP-1 features:

- Direct CMOS Logic Compatibility
- High Speed Switching (5nsec Typically at 1 amp)
- No Thermal Runaway
- No Secondary Breakdown
- 1000 MΩ Input Impedance
- Zener Protected Gate
- 35 Watt Power Dissipation

Figure 1 shows the ease of using the VMP-1 as a replacement for a conventional CMOS logic element. The input threshold begins at about 1V in the mA load current

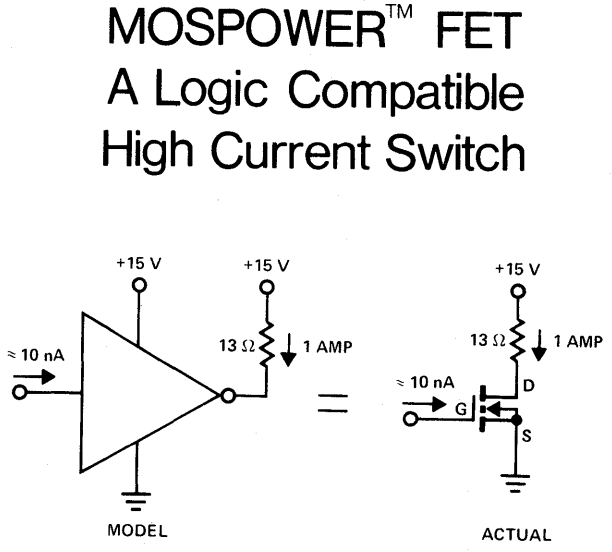


Figure 1

range and is guaranteed to support 1 amp of drain current at 10 Volts of gate enhancement. Note the complete lack of external resistors for input biasing, current limiting, or pull-up. The VMP-1 is a normally OFF device (an enhancement mode FET) when the input is at ground and a fully ON device which draws no input current when the input is high. The rather arbitrary 10nA gate leakage current shown gives the device a "Beta" of 100,000,000 if one compared it to a Darlington bipolar.

The VMOS or Vertical Metal Oxide over Silicon device shown in Figure 2a differs from the older lateral MOS structure shown in Figure 2b in that the critical P type gated body region width for VMOS is an easily controlled vertical diffusion dimension whereas the conventional MOS device uses a photolithographically controlled body width. VMOS therefore offers higher

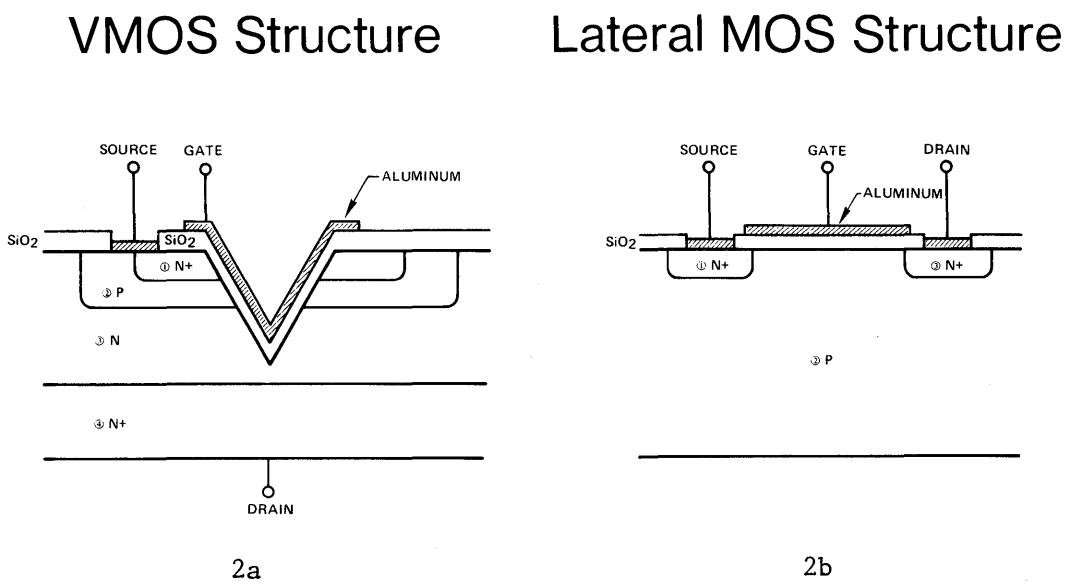


Figure 2

current densities, higher speed, and lower ON resistance than conventional lateral MOS with the added bonus of smaller chip area. These considerations have effectively kept any power MOSFET's off the market until the advent of VMOS.

The drain to source ON resistance versus gate voltage curve of Figure 3 shows the ON resistance that can be expected for various logic input levels. Unlike a bipolar device the VMOS structure gives a simple resistive characteristic rather than the built in offset voltage found in a bipolar.

Drain-to-Source ON Resistance vs Gate-to-Source Voltage

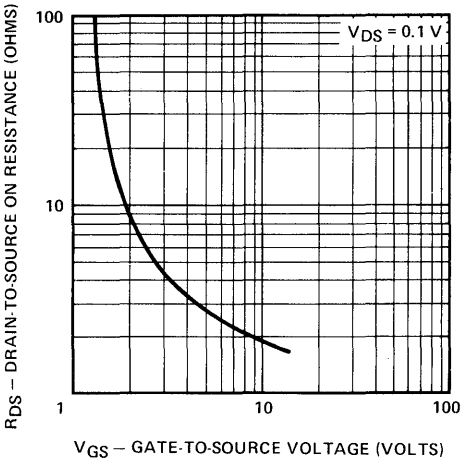


Figure 3

Typically a 2N3055 will exhibit 1.5 to 10mV of offset voltage (with from 2 to 200mA of base drive) even under zero collector load current conditions. The VMP-1 shows no offset voltage under low load conditions, although both a bipolar and FET device will exhibit volt range drops for higher load currents.

The typical output characteristics shown in Figure 4 demonstrate the exceptional voltage versus current linearity possible with the VMP-1 operated above its threshold voltage. This curve also indicates the amount of enhancement voltage normally required to sustain various switching load currents.

Output Characteristics

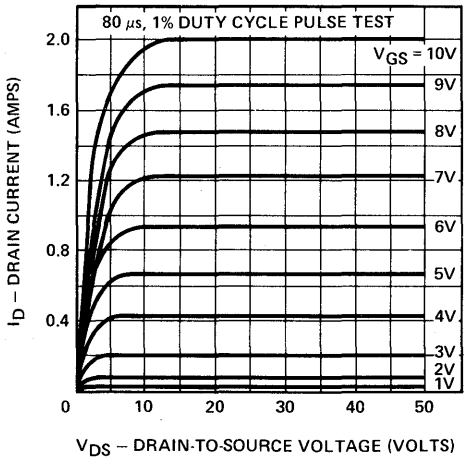


Figure 4

The VMP-1 is presently being offered in a TO-3 package and is rated at 35 Watts for a 25°C case temperature. Figure 5 shows the power derrating curve for higher case temperature with a fairly conservative maximum junction temperature of 150°C.

Power Dissipation vs Case Temperature

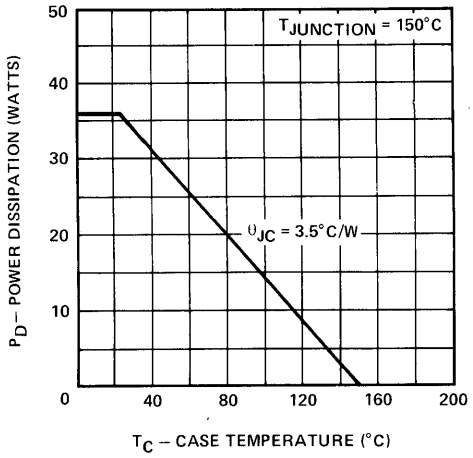


Figure 5

Experiments to determine the speed of the VMP-1 in high current switching applications in typical CMOS logic system applications reveal that numbers approaching the data sheet specification of 1 amp in 5 nano-seconds are easily attainable. Figure 6 shows the VMP-1 being directly driven from four parallel sections of a 34011 quad two input NAND gate. The 20 nsec rise and fall times closely approximate the theoretical limit.

1 Amp High Speed CMOS Buffer

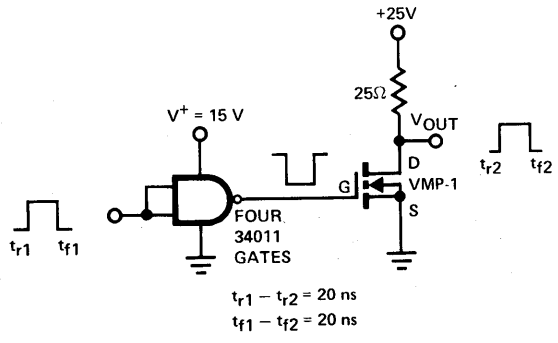


Figure 6

Figure 7 shows the actual waveforms observed on the CMOS gate input and the output drain connection of the VMP-1.

CMOS Logic Plus
VMP-1 Switching Times

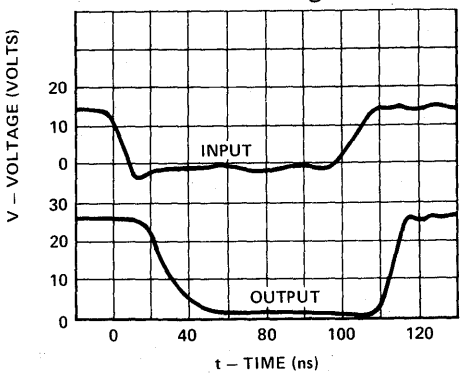


Figure 7

Using only a single NAND gate to drive this 1 amp switch results in 20 nsec fall time delays, but more like 50 nsec rise time delays. This is due to the difficulty of a single CMOS gate pulling down the approximately 40pF input capacitance of the VMP-1.

Figure 8 shows the tremendous parts savings and reliability improvement possible in a microprocessor system application when VMP-1's replace Darlington bipolars.

Microprocessor Interface Drivers

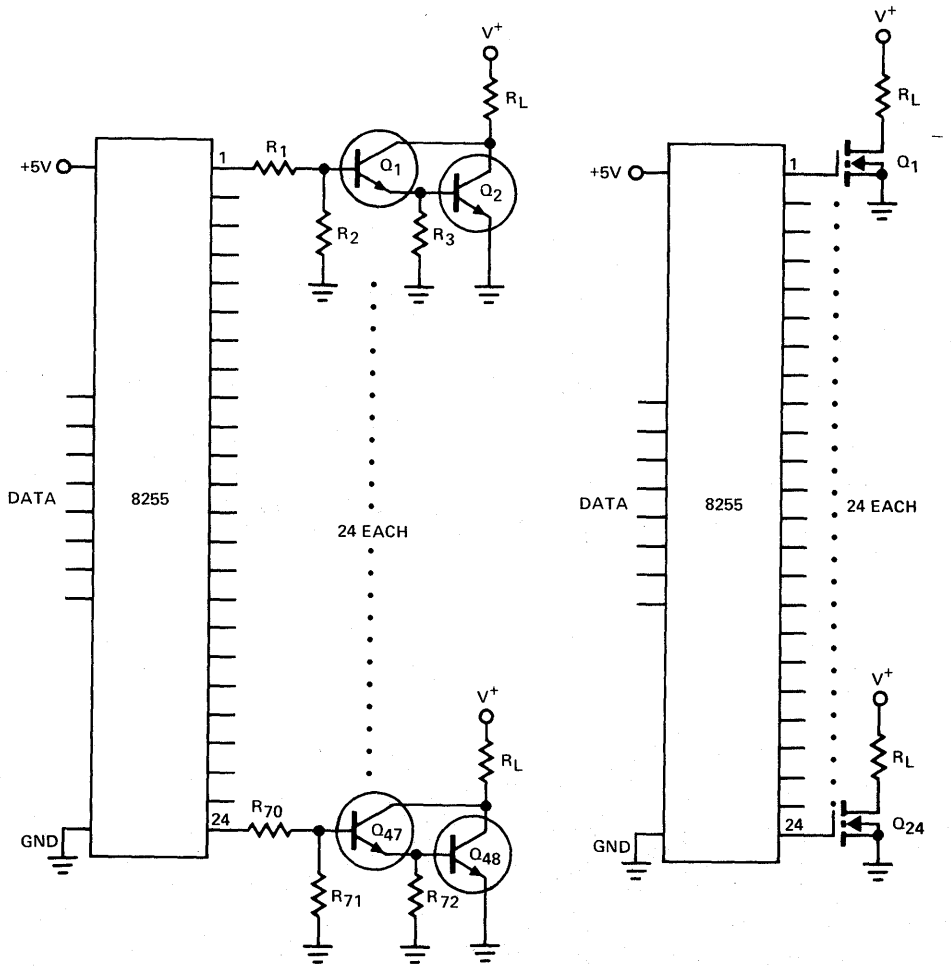


Figure 8

The 72 resistors and 48 bipolar transistors can be replaced by 24 VMOS devices in this 8000 series Peripheral Interface circuit designed to control printers, solenoids, displays, actuators, etc., under the command of an 8080 type microprocessor. The 1000 MΩ range input impedances of the VMOS devices not only allow direct connection to the 8255, but also dramatically cut its power consumption.

For applications requiring higher load currents than offered by a single VMP-1 one can simply parallel as many as required. No special care need be taken to insure even distribution of load current between devices since the VMP-1's have positive drain to source resistance temperature coefficients. This varies markedly from the negative temperature coefficient found with bipolar devices which tend to send the most current to the hottest device in a thermal runaway type of action. The three VMOS devices shown in Figure 9

Parallel Operation

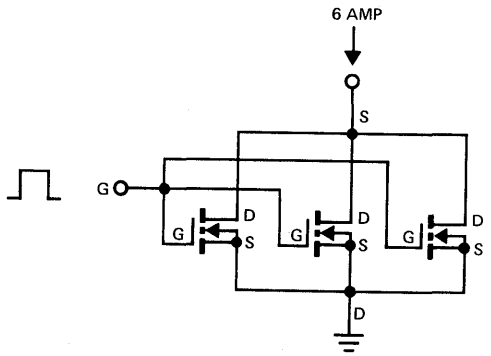


Figure 9

will handle 6 amps of load current as shown and one can continue adding an almost infinite number of devices at 2 amps more load current per device with NO INCREASE IN DC DRIVE CURRENT requirement. This means a designer or user can finally just clamp together more and more power modules without concern for cascading additional drive circuits, etc., when making higher and higher current motor control, power supply, or other solid state switching systems.

The circuit of Figure 10 shows how the VMP-1 can be used with common open collector 7400 series TTL logic. Decreasing the value of the 10K TTL pullup resistor will slightly speed up the circuit, but for a lamp type application such as this anything from 1K to 100K would be adequate.

TTL Logic Compatible Lamp Driver

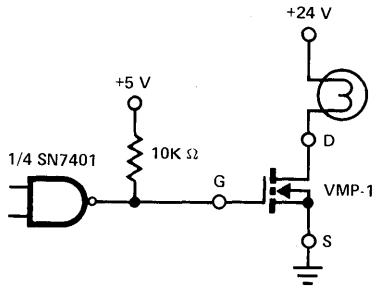


Figure 10

The fairly inexpensive and minimum parts count 1 KHz logic controlled audio oscillator shown in Figure 11 demonstrates both the logic compatibility and the practicality of the VMP-1 in consumer type circuits.

Audio Alarm

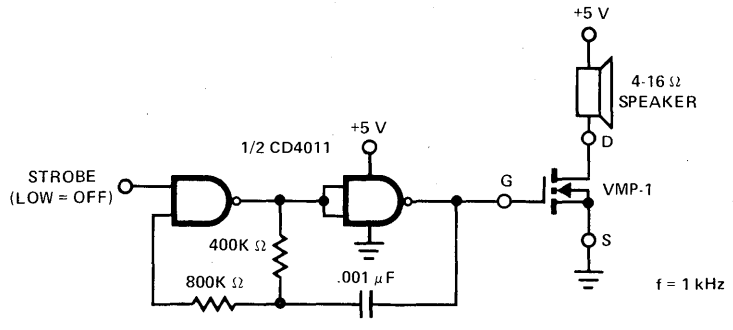


Figure 11

The CMOS logic controlled analog switch shown in Figure 12 has an ON resistance of from 2Ω to 3Ω for analog signals from 0V to 10V. The OFF leakage is less than 0.5μA. Since the body is tied to the source (necessary in a 3 lead power type package) the analog current flow should always be from drain-to-source. Reverse current flow would encounter the forward biased body-drain PN diode shown in Figure 2a.

Low Resistance Analog Switch

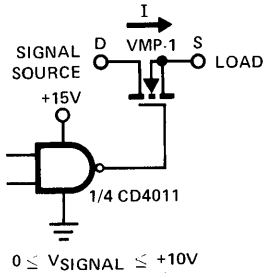


Figure 12

The VMP-1 MOSPOWER™ FET switch offers several new dimensions in high current, high speed, and high "Beta" capability for power switching designs. Its elegant simplicity for logic compatible designs and its versatility in improving on the performance of Darlington bipolars promise a very wide spread acceptance of this device.

11.

EVOLUTION OF THE IC OP AMP

JIM SOLOMAN, TOM FREDERIKSEN, AND
NELLO SEVASTOPOULOS
National Semiconductor
Santa Clara, California

Early op amps made use of lateral PNP transistors (Fig. 1) to solve dc level shifting problems in design. These monolithic transistors have relatively poor f_T resulting in limited overall frequency response in the op amp. In addition, bipolar transistors (when biased for good frequency response) make the input current relatively large.

The Super- NPN transistors first announced in the National LM108 op amp (Fig. 2), reduced the input current. This was a major step forward in op amp technology. Unfortunately, lateral PNP transistors were still needed and therefore, the speed of the op amp was essentially unchanged.

Slew-Rate enhancement techniques (Fig. 3) have been developed, but these tend to degrade both V_{OS} and V_{OS} drift due to the increased circuitry added at the input stage. More important are the problems which arise due to the undesired gain enhancement during slew which can cause transient instabilities. Finally, a large differential input overdrive is required for enhancement.

A useful technique for increasing the bandwidth is to feed the signal around the slow responding stage (Fig. 4). This "Feedforward" technique was introduced in a monolithic amplifier by National (LM118). Although the bandwidth is significantly improved, the settling time and input current are relatively unaffected.

Today we see two trends in new op amp products (Fig. 5). One of these is to lower the cost of the standard op amp. National has led the industry in the introduction of the low-cost Quad op amps. The other trend is to provide a high performance op amp for the more demanding applications.

As we have seen, the limitations of the standard linear IC process allow only an improvement in a single specification at a time. To overcome these limitations we need some better active devices to design with.

The BI-FET process (Fig. 6) gives us an improved active device, an ion implanted P-channel JFET which can be build on the same chip with standard bipolar transistors. The good control of the implant results in well matched JFETs. They are also wider band devices than lateral PNP transistors with roughly the same breakdown voltages. Their input current, however, is orders of magnitude less.

Notice that the matching of these FETs (Fig. 7) gives essentially the same V_{OS} and V_{OS} drift as that of the bipolar transistors. The input current (I_g) is down several orders of magnitude, the noise performance is improved-especially for large source resistances, and we have also overcome the bandwidth limitations of the lateral PNP transistors.

Most IC op amps can be modeled as a cascade of a transconductance amplifier and a transimpedance amplifier (Fig. 8). The first stage converts a differential input voltage into a single-ended output current. This current is the input to the second stage. The maximum value of the current determines the rate at which the voltage across the compensation capacitor (C_C) can change which is the slew rate of the op amp.

The overall gain is the product of the first stage transconductance (g_m) and the second stage transimpedance ($1/j\omega C_C$). The bandwidth, the frequency where this gain is unity (ω_{BW}) therefore is determined by the ratio of g_m to C_C.

Introducing the second equation into the first (Fig. 9), we see that slew rate is proportional to the ratio of the maximum first stage output current (i_{MAX}) to the first stage transconductance (g_m). For a bipolar transistor this ratio is kt/q. For a FET operating at I_{DSS}, this ratio is one-half the pinch-off voltage (V_{PO}). Therefore, the slew rate of a FET input op amp can exceed that of a bipolar input op amp by a factor of 20.

The basic design of the LF 156 (Fig. 10) is a differential JFET input stage followed by a differential bipolar second stage (for symmetrical second stage bias current loading). FET current source loads are used for the first stage to minimize V_{OS} and V_{OS} drift. As a result, a common-mode feedback loop is required.

Common-mode feedback loops are interesting. At dc, the compensation capacitor (10pF) is an open circuit and the feedback to the sources of the input FETs is common-mode. For ac inputs, the compensation capacitor will absorb the output current of the first stage. Note that the other differential output current is constrained to be essentially zero (there is no place for current to be absorbed). Therefore, the entire differential input voltage is impressed across the gate-source of the non-inverting input FET. This provides a gain doubling differential to single-ended conversion.

The performance of the output stage of an op amp has also been limited by the poor frequency response of the monolithic lateral or vertical PNPs. A significant improvement in the open loop output impedance (at both high output currents and high frequencies) has been made in the LF156 by use of a FET.

The ability of an op amp to absorb high frequency load transients or to drive large capacitive loads (and maintain stability) is determined by the high frequency open loop output impedance (Fig. 11). In order to keep this small, a high frequency FET-NPN composite is used for the lower side and the upper NPN output transistor remains biased "ON" even during output current sinking to bypass the composite at high frequencies.

Adjusting the offset voltage of the LF 156 (Fig. 12) essentially does not affect V_{OS} drift. With conventional op amp V_{OS} adjust techniques, external resistances shunt on-chip resistances to change V_{OS}. The resulting TCs of the two equivalent resistances do not match and therefore V_{OS} drift is degraded. In addition, the

signal path is affected and CMRR and gain can be degraded.

The offset adjust schematic of the LF 156 uses a single 25 k potentiometer to set differential FET currents which modify dc biasing only and therefore does not degrade the amplifier.

The LF 157 (Fig. 13) is a broadbanded version of the LF 156. This amplifier is useful in applications requiring: wider bandwidth, faster slew rate or larger power bandwidths. Using the conventional feedback configuration the closed loop gain must be 5 or greater to insure stability.

The LF 157 can be used as a unity gain inverter in the connection shown in Fig. 14.

This unusual feedback schematic maintains the slew rate and power bandwidth of the LF 157.

In keeping with the current "National" policy to conserve energy, a lower current drain version of the LF 156 (the LF 155) is also available (Fig. 15) Although the input commonmode voltage range and the output voltage swing become a smaller portion of the supply voltage, the LF 155 can be operated on 6V batteries.

The noise benefits of a FET input op amp are most significant for large source resistance (Fig. 16). This is a result of the very small magnitude of the input current. The noise current for a FET is so small it is hard to measure, but it can be calculated from the standard shot noise equation:

$$\frac{i_n}{\sqrt{Hz}} = \sqrt{2qI_G}$$

The benefits of the new V_{OS} adjust circuitry which is used in the LF 156 are shown in Fig. 17. A change in V_{OS} drift of only $0.5\mu V/^{\circ}C$ per mV of adjust is typically achieved.

Notice that the long term V_{OS} drift of the LF 156 approaches that of the better bipolar op amps (Fig. 18).

The requirements for fast settling time, low bias current and low offset voltage of the current to voltage converter op amp which is used at the output of a DAC make the LF356A a good choice (Fig. 19). As a result of the switching which takes place at the input of this op amp, transition "glitches" can appear in the output voltage waveform. A high speed, fast settling op amp reduces the total time during which there is an output voltage error due to these transitions.

A less obvious problem is the requirements on the op amp which drives the "apparently dc" bias line for the current switches. The high frequency open loop output impedance and the settling time of this amplifier can limit the overall settling time of the complete DAC.

In a precision application, where a large loop gain is required, the available open loop gain of standard op amps may be insufficient

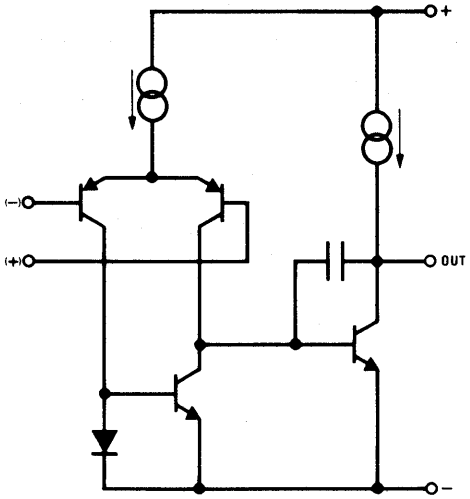
even as relatively low frequencies. The large gain bandwidth of the LF356 is needed in many "apparently" low frequency applications to increase the loop gain.(Fig. 20).

In this standard three op amp instrumentation amplifier of Fig. 21 the CMRR depends upon the matching of the four resistors which are tied to the output op amp. The input op amps can degrade the overall CMRR if they generate a difference between their output voltages as a result of a common-mode input signal. In addition, the CMRR of the output op amp must also be large. Therefore, all three op amps require large CMRR and in addition all the resistors need to be well matched for a high quality instrumentation amplifier.

Active filters (see Fig. 22) require a wide gain-bandwidth op amp building block to provide high Q at high center frequencies ($\cong$10 kHz). In addition, the low input bias current and high input impedance of the LF356 allows the use of relatively small valued, less expensive capacitors which is especially important at low frequencies.

In voltage tunable oscillators (see Fig. 23) which have a wide tuning range, the low input bias current of the LF 156 is required. This easily allows extensions of the low frequency range. Also, the high frequency limit is extended over that obtained with standard op amps.

LM709/LM101/LM741

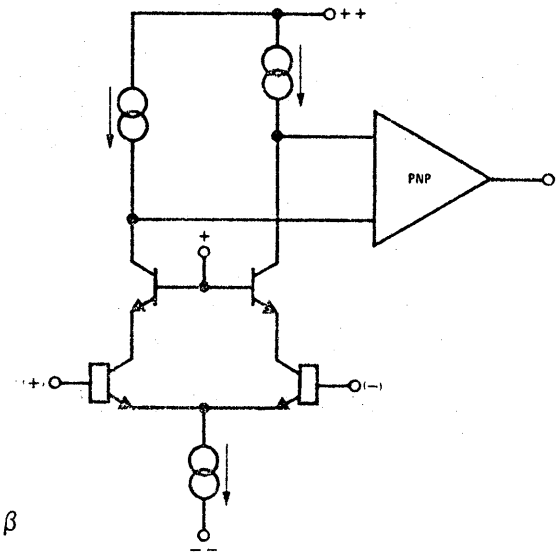


LATERAL PNP'S LIMITS SPEED

ALSO BIPOLAR CAUSES HI I_{BIAS}

Fig. 1

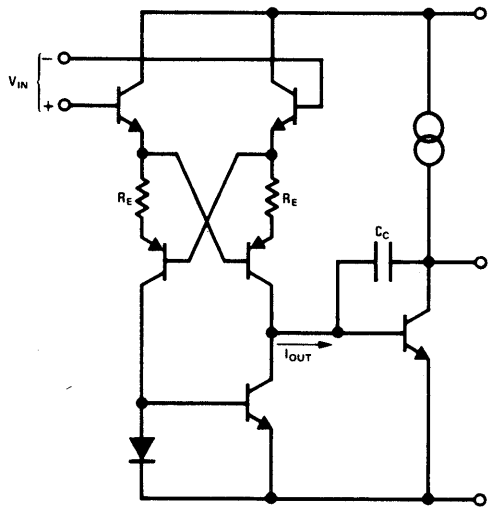
LM108

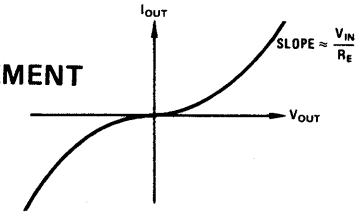


SUPER β

REDUCES I_{BIAS}, BUT SPEED . . .

Fig. 2



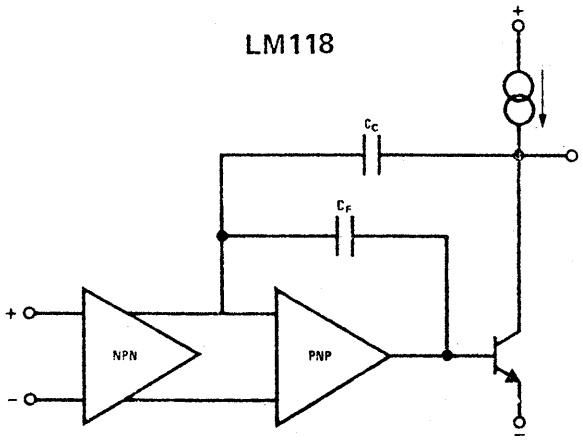


SLEW ENHANCEMENT

LARGE I_{OUT} DRIVES C_C FOR HI SLEW . . .
BUT *NOT* HI BW

Fig. 3

LM118



FEEDFORWARD

GIVES LARGE BW & SLEW
BUT *NOT* SETTLING OR I_{BIAS}

Fig. 4

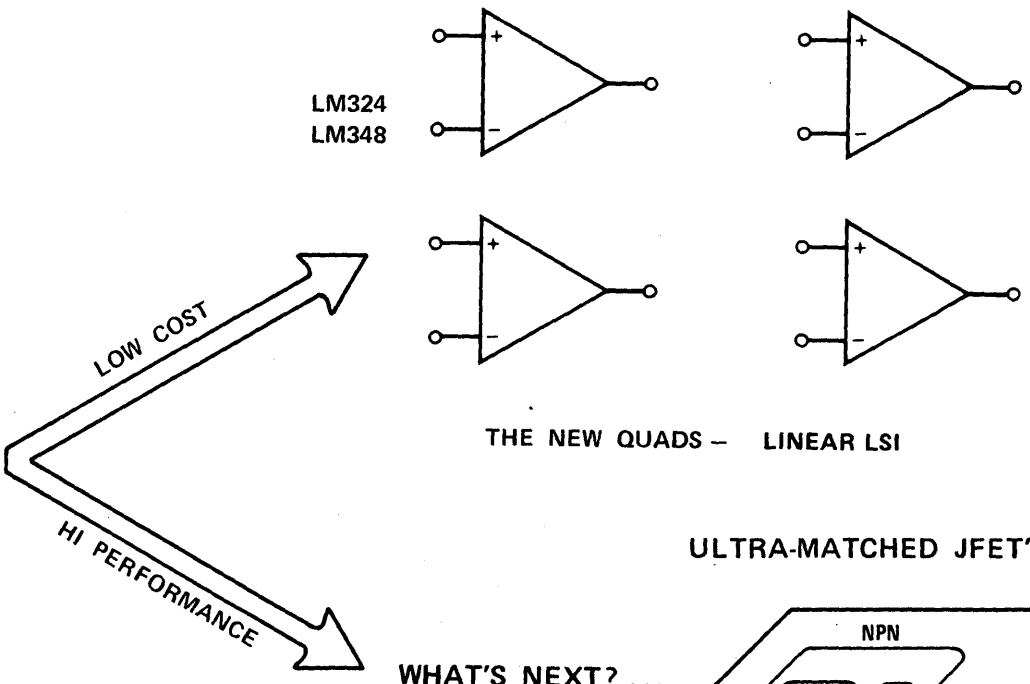
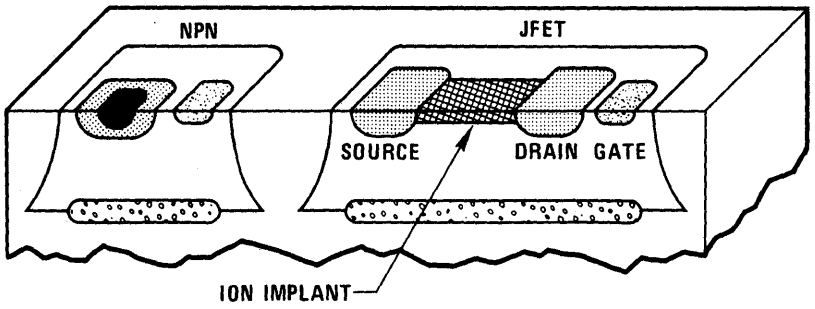


Fig. 5

ULTRA-MATCHED JFET'S AND BIPOLAR TRANSISTORS



LOW V_{OS}, LOW I_{BIAS}
WIDE BW & SLEW RATE? ...

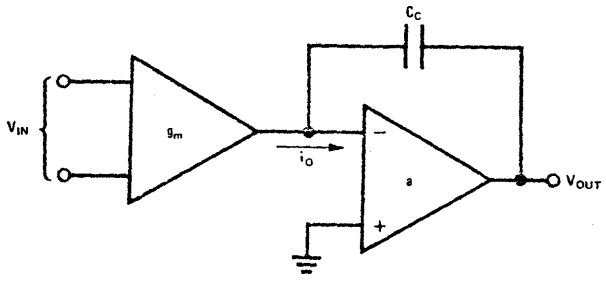
Fig. 6

COMPARISON OF JFET AND BIPOLAR TRANSISTORS

	P-CHANNEL JFET'S		PNP BIPOLAR DEVICES
I_G	30 pA	I_B	3000 pA
V_{OS}	2 mV		ABOUT THE SAME
$\Delta V_{OS}/\Delta T$	3.5μV/°C		ABOUT THE SAME
BW	10 MHz		1 MHz
BV_{GSS}	50V		
NOISE VOLTAGE	8 nV$\sqrt{Hz}$		
NOISE CURRENT	"FORGET IT"		

Fig. 7

CONSIDER SIMPLE OP AMP MODEL:



$$\text{SLEW RATE} \approx \frac{i_{o(MAX)}}{C_C}$$

$$\omega_{BW} \approx \frac{g_m}{C_C}$$

Fig. 8

COMBINE SLEW AND BW EQUATIONS:

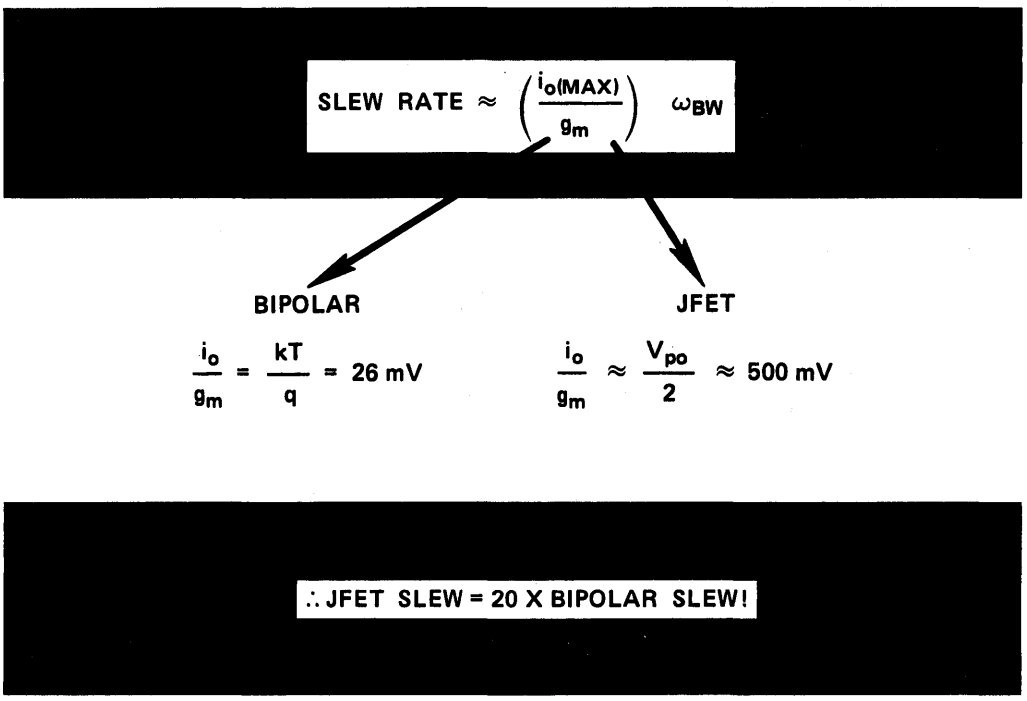


Fig. 9

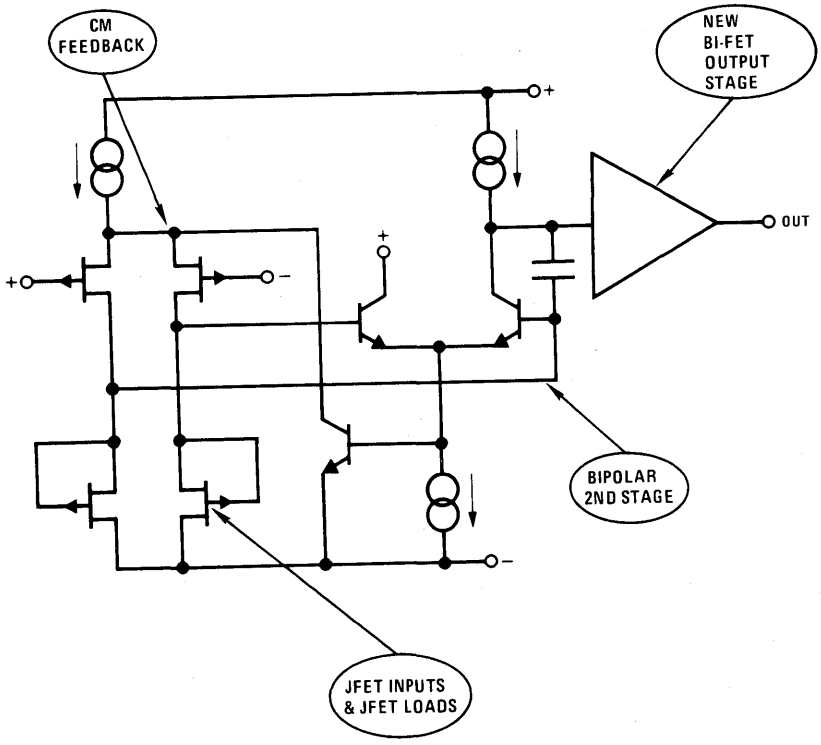


Fig. 10

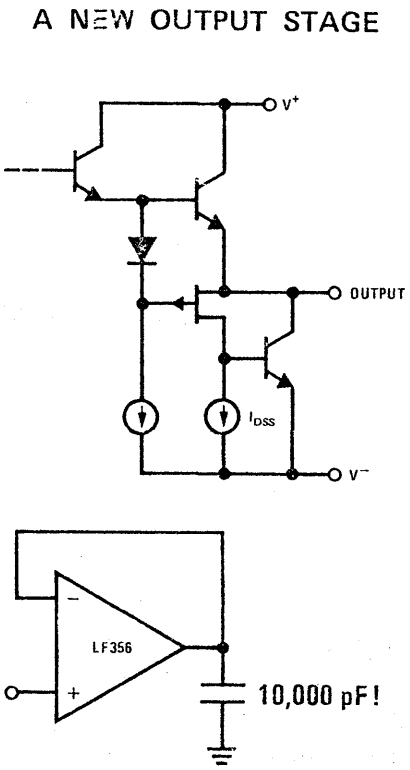


Fig. 11

LF156 COMPLETE SCHEMATIC

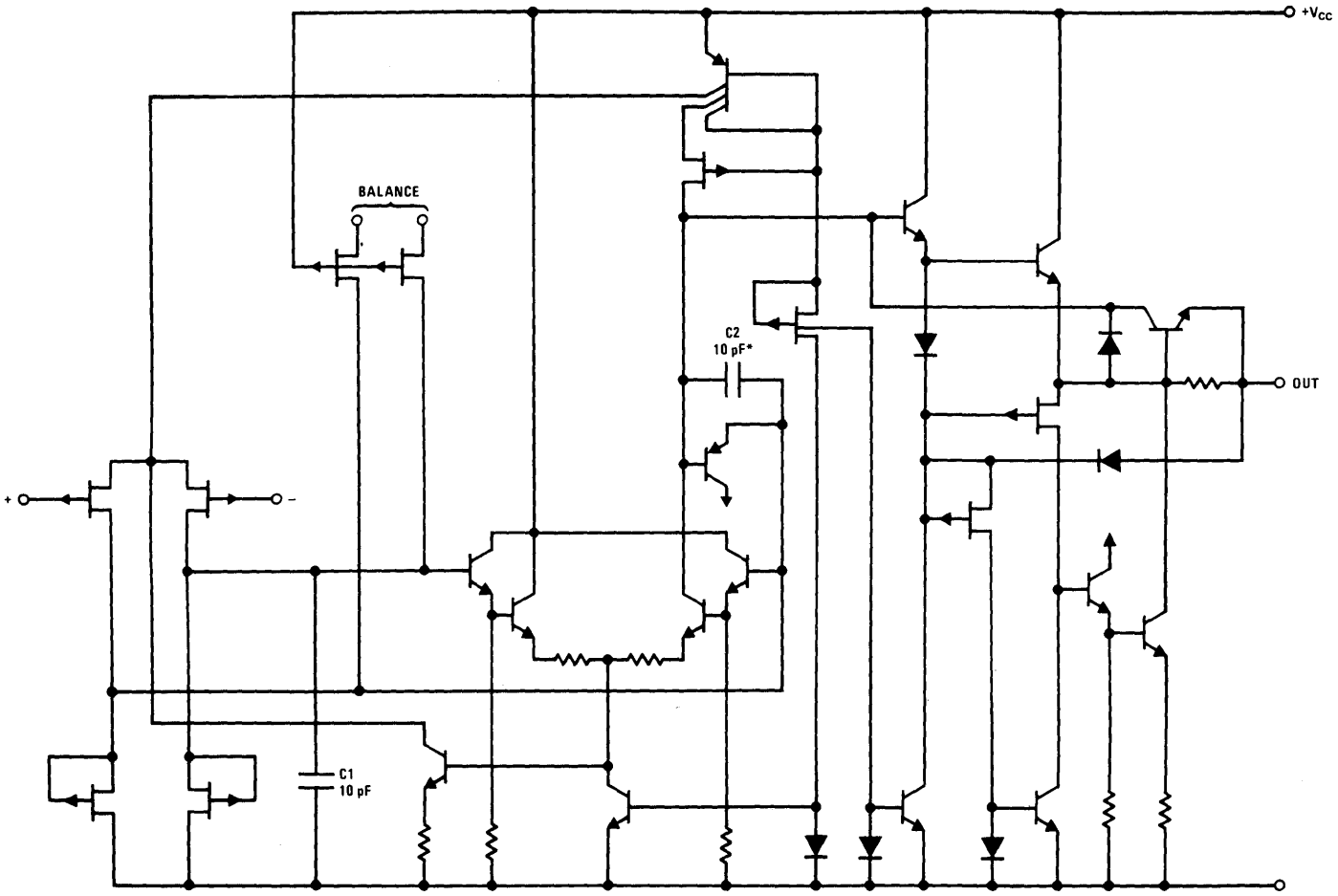


Fig. 12

LF156A/LF356A:

V_{OS}	2 mV MAX
I_{OS}	10 pA MAX
V_{OS} DRIFT	5μV/°C MAX
GBW	4 MHz MIN
SLEW RATE	10 V/μs MIN
0.01% SETTLING	1.4μs TYP
e_{NOISE}	12 nV/$\sqrt{Hz}$ TYP
i_{NOISE}	0.01 pA/$\sqrt{Hz}$ TYP
CMRR	80 dB MIN

Fig. 12 (a)

SUPER FAST FOR $A_V \geq 5$ *LF157*

SLEW RATE	50 V/μs
GBW	20 MHz
POWER BW	500 kHz

OTHER SPECS SAME AS LF156 . . .

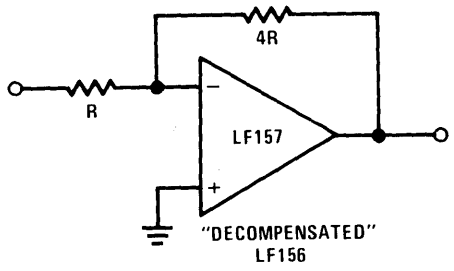
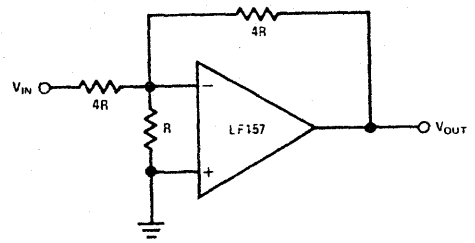


Fig. 13

LF157 AS UNITY GAIN INVERTER



- $V_{OUT}/V_{IN = 0} \cong \pm 5 \cdot (V_{OS})$
- BW = 5 MHz
- POWER BW = 500 kHz

Fig. 14

LOW CURRENT DRAIN LF155

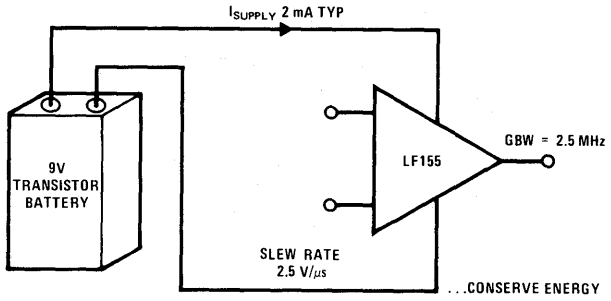
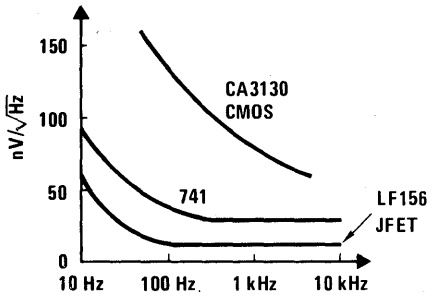


Fig. 15

INPUT NOISE VOLTAGE



INPUT NOISE CURRENT

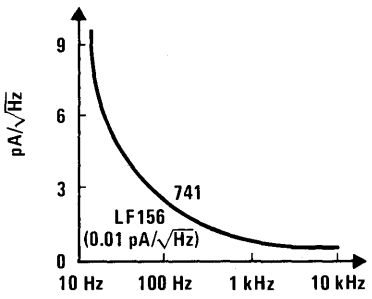
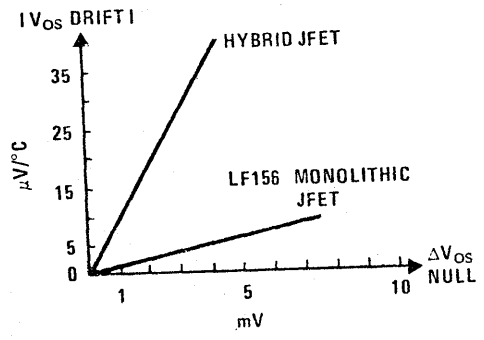


Fig. 16

THE LF156 HAS LESS DRIFT PER MILLIVOLT OF OFFSET ADJUST THAN ANY OTHER FET OP AMP



CHANGE IN OFFSET DRIFT WITH V_{OS} NULL

Fig. 17

LONG TERM DRIFT OF OFFSET VOLTAGE (T_j = 125°C)

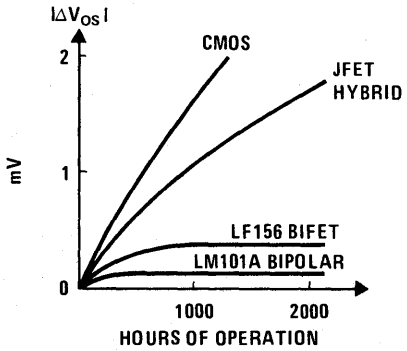


Fig. 18

FAST 0.01% ACCURATE 12-BIT DAC

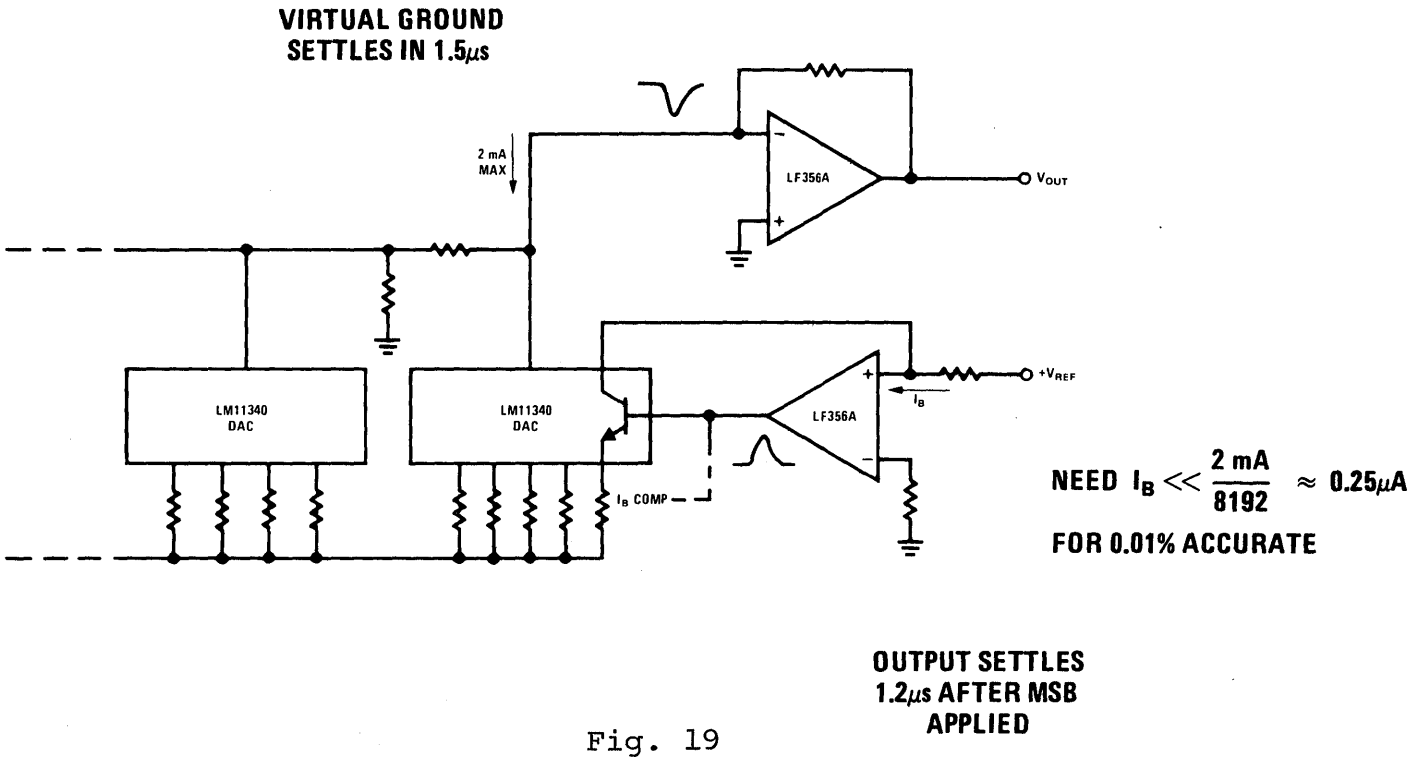


Fig. 19

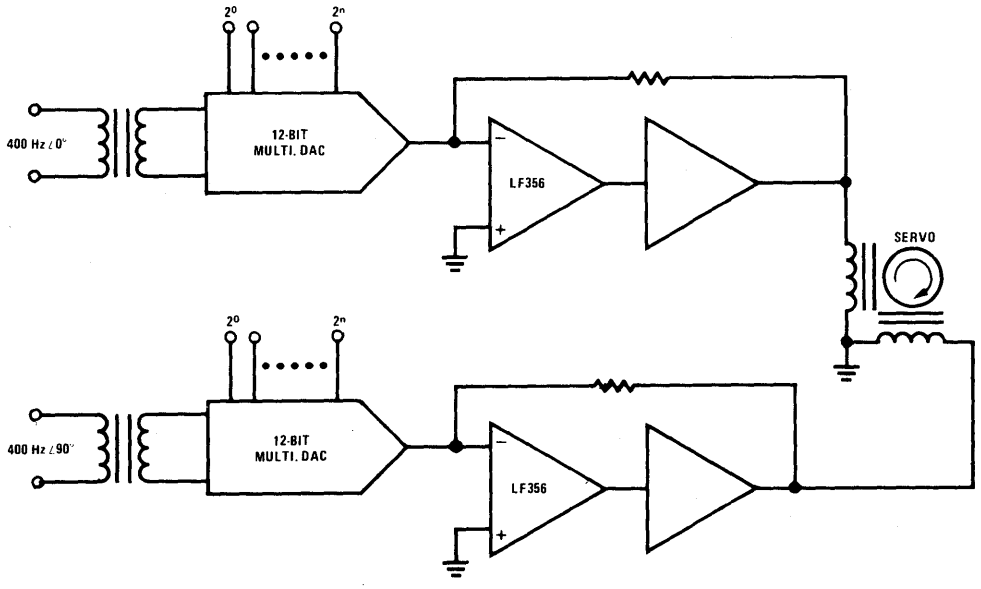


Fig. 20

HIGH IMPEDANCE INSTRUMENTATION AMPLIFIER

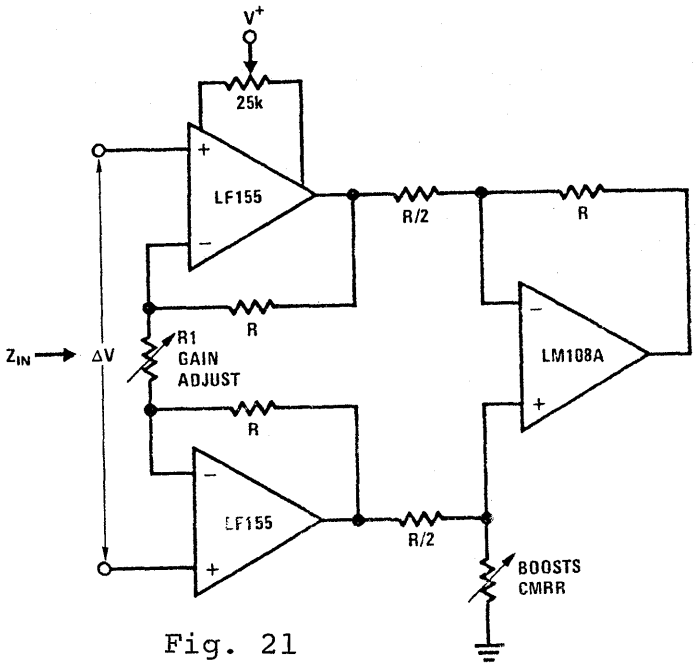
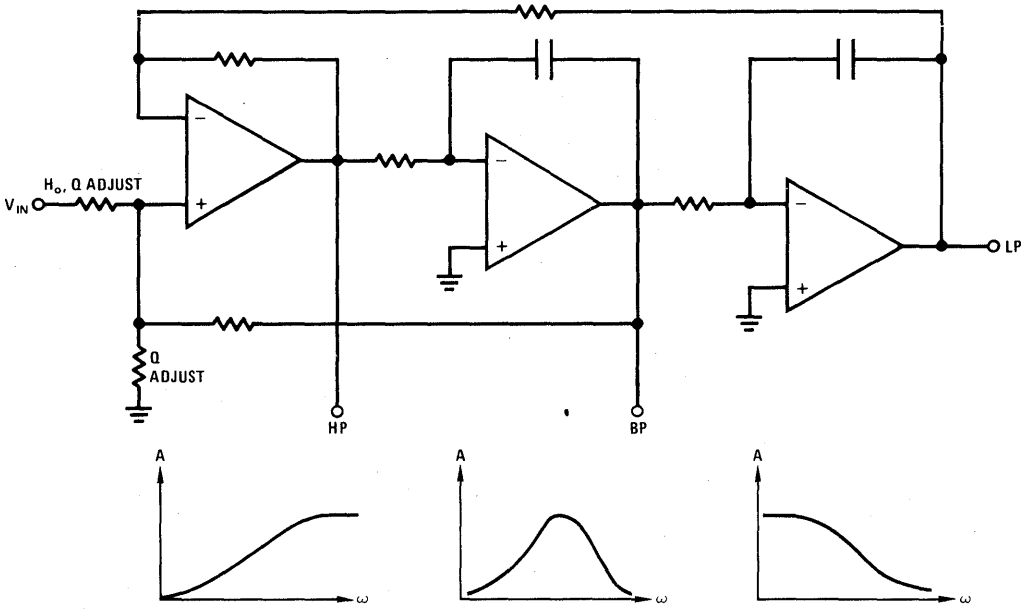


Fig. 21

STATE SPACE FILTER



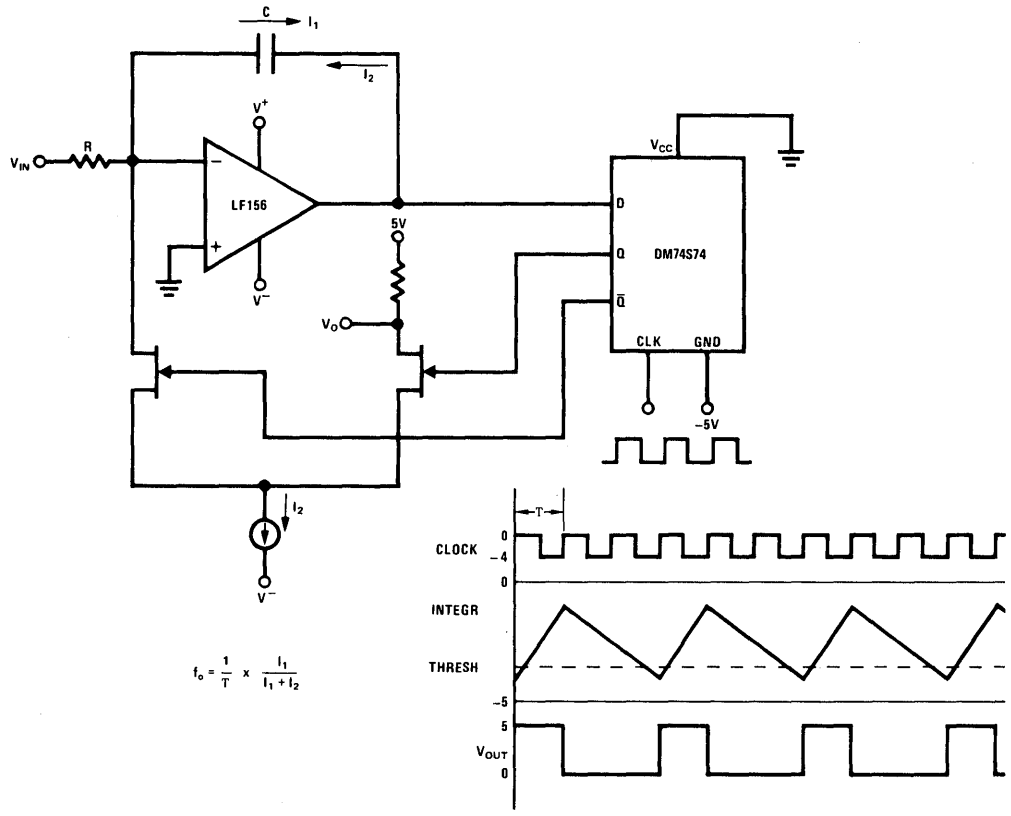
NEED $Q \times f_o \leq \left.\frac{GBW}{10}\right|$ OP AMP

$$V_{OUT} = 2\left[\frac{2R}{R1} + 1\right]\Delta V$$

- LF356 ALLOWS Q ≈ 50 AT f_o ≈ 10 kHz
- 500 kHz 10Vp-p POWER BW ELIMINATES DISTORTION DUE TO SLEW
- CMRR: 100 dB

Fig. 22

THE LF156 IN A V/F CONVERTER



LOW I_{BIAS} ALLOWS WIDE RANGE

Fig. 23

12. 164 CHANNEL FREQUENCY SYNTHESIZER
FOR CITIZENS BAND, 82 CHANNEL TELEVISION
CATV AND MARINE RADIO

Andrew C. Tickle
Director of Advanced Products
NITRON
Cupertino, California

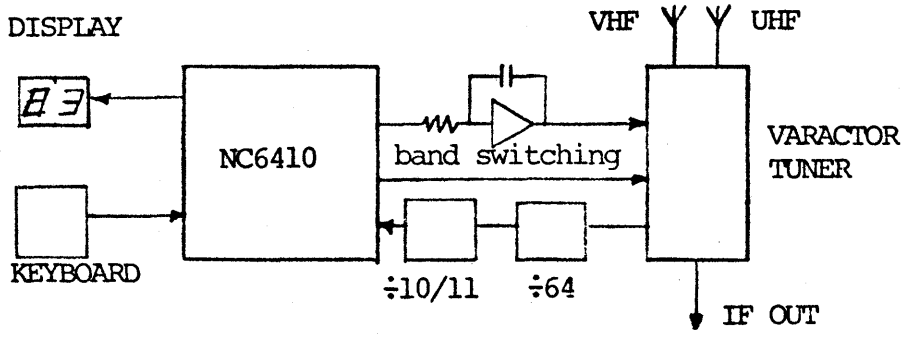
The NC6400 is a single LSI circuit which replaces most of the standard digital circuits in multichannel synthesized radio and television applications. 164 frequencies are stored in a 2378 bit on-chip ROM accessed directly by a keyboard. The frequency range of the chip is extended up to 1 GHz by using standard external prescaling counters.

MAIN FEATURES

164 mask programmable frequencies
Independent frequencies for transmit and recieve
Keyboard decode
Output to channel number display
Single 5V supply, TTL compatible

APPLICATIONS

Citizens band radio transceivers up to 82 channels
82 Channel TV tuning
Marine radio telephone, 55 to 82 channels
Airborne and navigational radio
Signal generators and testers



BLOCK DIAGRAM SHOWING 82 TV CHANNEL SELECTION

13. 64x4-BIT NONVOLATILE MEMORY
4-BIT BYTE ALTERABLE

Andrew C. Tickle
Director of Advanced Products
NITRON
Cupertino, California

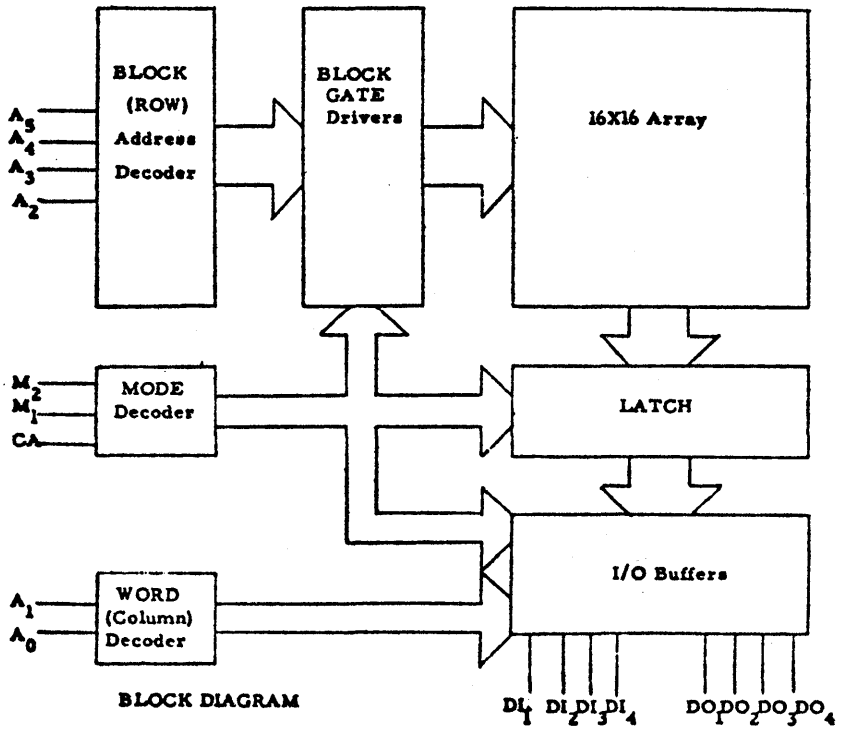
The NCM7040 is designed for simple interfacing and operation in systems that must retain data during power outages without battery backup. Data may be altered or updated either by operator command or by the system itself. An external programming system is not required.

MAIN FEATURES

Power Off Storage
Fully Decoded
Single BYTE Alterable
Chip Eraseable
Tri-Level Outputs
TLL and CMOS Compatible

APPLICATIONS

Program Storage Microprocessors
Preset Frequency Tuning
Repertory Dialing
Machine and Process Control
POS and Credit Status
Calibration and Error Correction



BLOCK DIAGRAM

14.

SELECTING, UNDERSTANDING AND USING 3-TERMINAL REGULATORS

JIM SOLOMAN, TOM FREDERIKSEN, AND NELLO SEVASTOPOULOS
National Semiconductor
Santa Clara, CA

The following chart illustrates some of the features of three-terminal and multiple-terminal regulators.

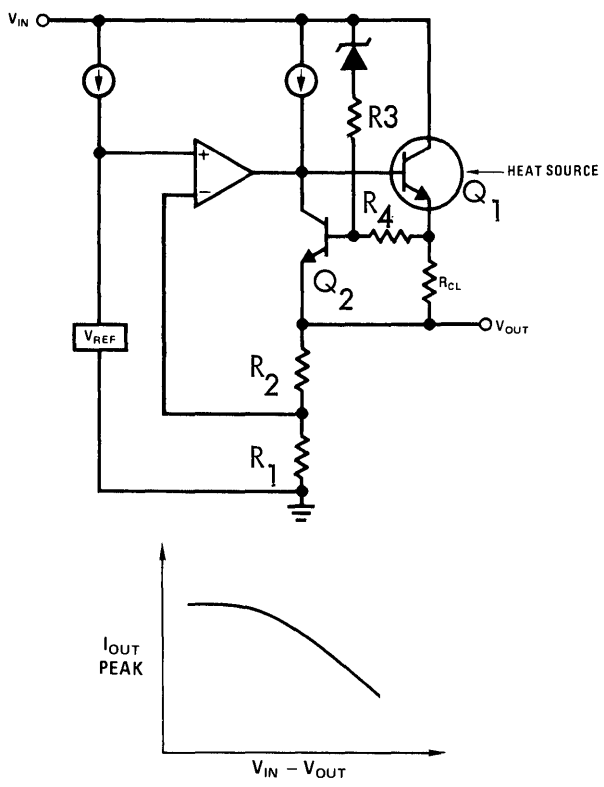
FEATURE	3 TERMINAL	MULTIPLE
V_{OUT}	Fixed (Same Tolerance)	Adjustable
Current Limit	Internal, Non-Programmable	External, Single Resistor Programmable
Foldback	Internal	Practical for + Complex for −
Thermal Shutdown	Internal	Complex Circuitry
Ease of Application	Very Easy	Uses Large Number of External Components

Note: Foldback current limiting in multiple terminal regulators (ex. LM104) is more complex because it occurs between $-V_{OUT}$ and $-V_{IN}$ (instead $+V_{OUT}$ and ground for positive regulators). Consequently, it requires more circuitry than simply 2 resistors.

So why a 3 terminal regulator? Simplicity: (1) faster thermal sensing, better protection; (2) regulator in a power transistor package; (3) space savings, cost savings.

Three terminal regulators have the pass transistor and the protection circuitry in the same package. This allows a faster thermal sensing of the pass transistor and consequently, a better protection. With multiple terminal regulators, thermal sensing of an external pass transistor will require added complex circuitry.

A SIMPLIFIED SCHEMATIC OF A 3 TERMINAL REGULATOR



Circuit description:

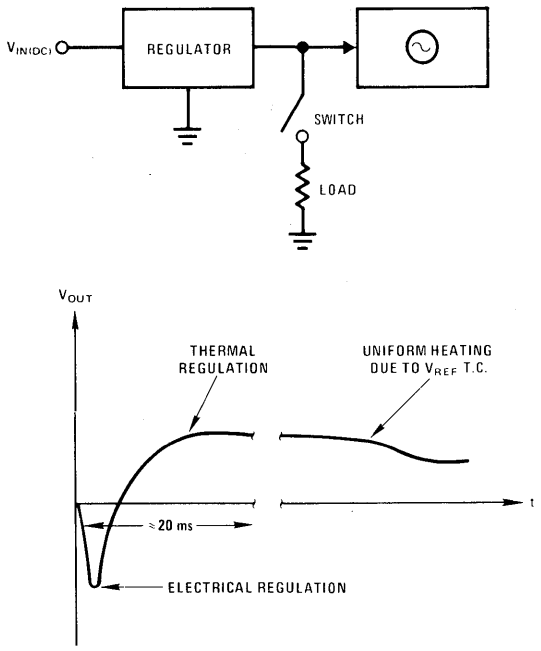
Q2, R_{CL}, R3, R4, Zener are elements of the protection circuitry of the positive 3 terminal regulator. Current limiting occurs by turning Q2 "ON" which removes base drive from Q1. When $V_{IN} - V_{OUT}$ is less than approximately 7V, R_{CL} senses the full output current which, in this case, is about V_{BE}/R_{CL}. This is represented by the flat portion of the curves. When $V_{IN} - V_{OUT}$ exceeds 7V the zener breaks down.

The current flowing through R3, R4 and R_{CL} prebiases transistor Q2 and current circuit limit will occur at lower output currents. This is represented by the negative slope portion of the curves. The rate of reduction of current limit (or output peak current) with increase in $V_{IN} - V_{OUT}$ is given by:

$$\frac{\Delta I_{OUTPEAK}}{\Delta(V_{IN}-V_{OUT})} = -\frac{R4}{R3\,R_{CL}}$$

Also,the regulator has another transistor (not shown in the above schematic) physically located close to the pass transistor Q1. It is normally biased "OFF" for junction temperature less than 150°C. At high junction temperatures (150°C to 190°C) this transistor turns "ON," removes base drive from Q1 and shuts down the regulator.

SOMETHING ABOUT THERMALS !



In the previous slide, we show the pass transistor of the 3 terminal regulator acting as a heat source. In fact, few designers realize that many of the specifications limits of high power regulators are determined by thermal considerations, rather than electronic ones. If we apply a high current step load to a 3 terminal regulator the voltage output will typically look as illustrated in the figure above. This response is due to both electronic and thermal effects, and it will consist of:

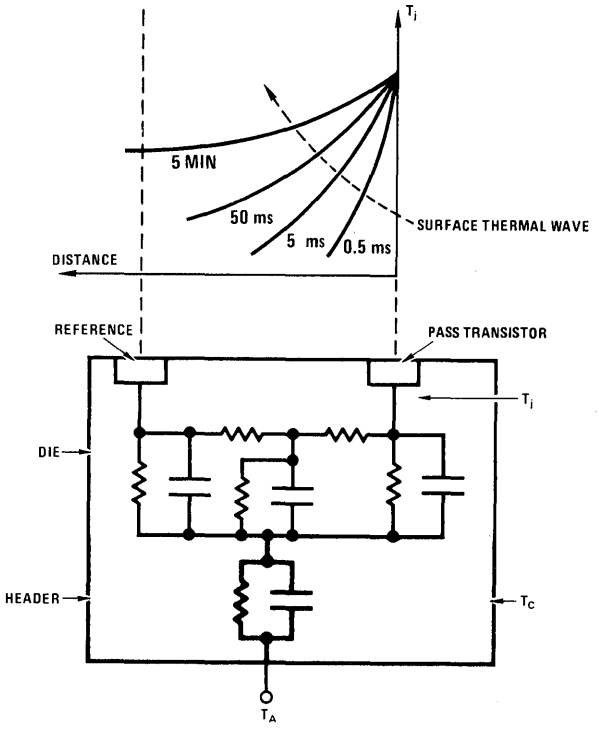
1) Initially a negative spike (not shown in the figure) due to the presence of regulator and circuit lead inductance. The duration of the spike generally is less than a μs. (This spike does not have the energy to destroy TTL's. Ex. 0.1μH, for 100 ns and 1A load will generate a −1V spike).

2) The negative spike will be followed by the electronic response of the regulator which will consist of a small negative step of a few μs duration. (It will depend on the load capacitor, and wire bond resistance in the regulator. Note that the LM123 (3A regulator) uses electronic compensation to cancel the effects of wire resistance.

3) The exponential decay of the electronic response will be followed by another exponential with a time constant of 20–40 ms. This is the major thermal response which results from a thermal gradient established across the die. It will be analyzed in the following slide.

4) The fourth portion of the regulator response shows a long term settling effect which is due to uniform heating in the die, header and heat sink. This heating causes a normal temperature drift effect in the voltage reference which results as small change in the output voltage.

A SIMPLE THERMAL MODEL



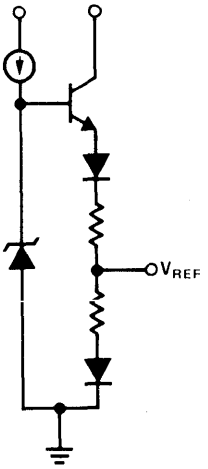
To study the thermal response of the regulator, we represent a cross section of the die where the power pass transistor and the reference voltage circuitry are coupled through a distributed RC transmission line. This line is the electrical analog of a thermal line with temperature replacing voltage, power replacing current and thermal resistance replacing electrical resistance:

$$\frac{\Delta T}{\text{POWER}} = \theta \text{ (thermal resistance)}$$

A step increase in power will cause a pass transistor temperature rise T_j. Temperature gradients (voltage drops) then begin to set up across the die as the heat propagates through the die (transmission line). The various components of the reference circuitry are no longer at the same temperature and small thermally-induced shifts occur in the reference voltage. This phenomenon reflects at the output as a change in the output voltage with respect to a change in power dissipation of the pass transistor. After approximately five minutes, the die is uniformly heated which causes the last portion of the regulator response curve (see previous slide).

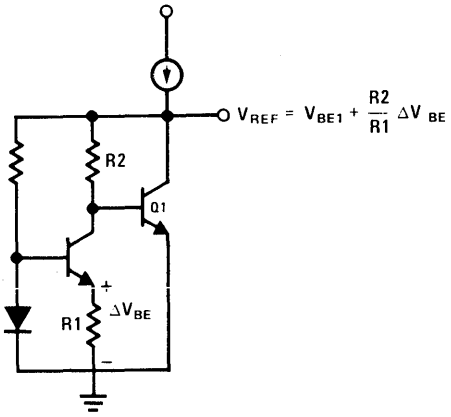
WHICH IS THE BEST V_{REF} ?

ZENER REFERENCE



- BETTER INITIAL TOLERANCE
- LOWER INITIAL DRIFT

BAND GAP V_{REF}



- LOWER NOISE
- BETTER TEMPERATURE COEFFICIENT

From the previous discussion, it follows that the voltage reference circuitry greatly affects regulator performance. The band gap reference (or ΔV_{BE} reference) uses the predictable base-emitter voltage difference (ΔV_{BE}) of two transistors operating at different current densities. It offers low noise, and very good long term stability. Because of the physical spread of components, the ΔV_{BE} reference is more sensitive to thermal gradients than temperature compensated zeners. On the other hand, the zener reference has better initial tolerance than ΔV_{BE} and better initial drift. Its major drawbacks are noise and long term stability due to surface contamination.

The long term stability of the zener reference can be improved by using ion implant to create a subsurface breakdown and to prevent surface contamination of the zener. Because of the good initial tolerance of such a reference, National now offers ±2%, 100μA regulator (LM340L).

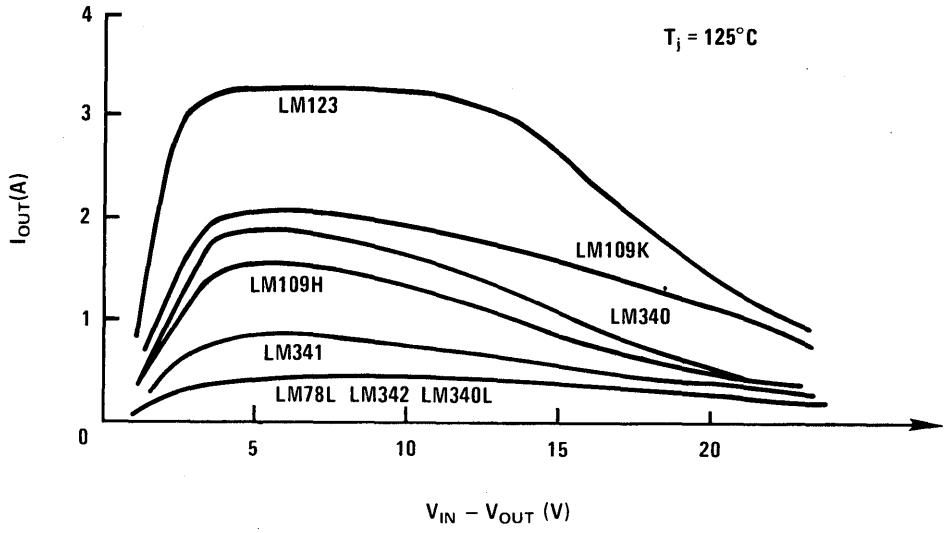
How do you select the proper regulator? You need to know: (1) V_{OUT} and tolerance, (2) V_{IN} Range, (3) I_{OUT}(Max), (4) T_A(Max), (5) T_j(Max).

First, pick a T_j(Max). Rule of thumb is T_j(Max) is less than or equal to 125 degrees C.

Even though the specifications allow T_j(Max) = 150°C, we suggest operating the regulator at a junction temperature at or less than 125°C for optimum reliability consistent with practical design. A more detailed discussion of T_j(Max) will follow.

Second, make an initial selection of possible regulators.

Third, narrow the choice by using curves to check I_{OUT}Peak.



Note that these curves show typical performance of positive regulators. However, they give a very good indication of the maximum current each device can provide with respect to $V_{IN} - V_{OUT}$.

THEN COMPUTE P_D FOR HEAT SINK REQUIREMENT

$$P_D = [I_{OUT} \times (V_{IN} - V_{OUT})] + [V_{IN} \times I_Q]$$

DETERMINE IF HEAT SINK IS NEEDED

$$\Theta_{JA(TOT)} = \frac{T_{j(MAX)} - T_{A(MAX)}}{P_D} \quad °C/W$$

Is the regulator capable of dissipating the required power, even if it is connected to an infinite heat sink? This is determined by considering the following inequality:

$$\theta_{JA(TOT)} > \theta_{JC} \quad \text{(of the chosen regulator)}$$

If this inequality is not satisfied, a higher power regulator must be selected.

With the proper regulator selected, determine if a heat sink is required.

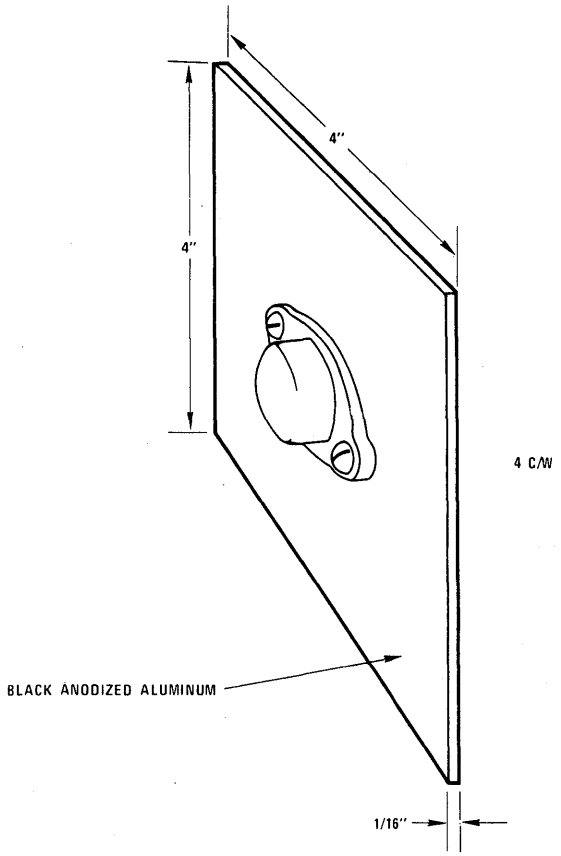
1) If $\theta_{JA(TOT)} > \theta_{JA}$ (of the chosen regulator) heat sink is not needed.

2) If θ_{JC}(regulator) $< \theta_{JA(TOT)} < \theta_{JA}$(regulator) heat sink is needed and its thermal resistance θ_{SA} can be computed from:

$$\theta_{SA} = \theta_{JA(TOT)} - \theta_{JC}(\text{regulator}) - \theta_{CS}$$

θ_{CS} is the thermal resistance from regulator case to heat sink, and it depends upon the quality of heat sink mounting. In any case θ_{CS} is not a necessarily negligible number.

Select proper heat sink using our selection guide or design your own. Remember: (1) mount cooling fin vertically, (2) anodize or paint fin surface for better radiation heat flow!



AN EXAMPLE

V_{OUT} = 5V ±5%, V_{IN} = 15V

$I_{OUT(MAX)}$ = 0.7A

$T_{A(MAX)}$ = 60°C; T_j = 125°C

1. FOR 0.7 AMPS

   LM340T-05, LM309K, LM340K-05, LM341P, LM309H, LM323K

2. FOR THIS POWER DISSIPATION

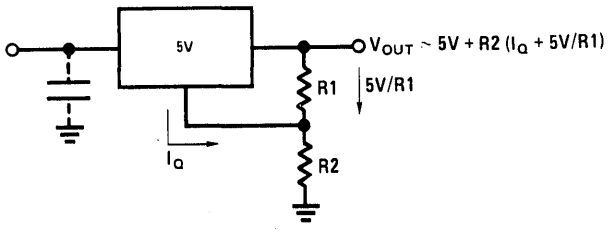
   $\Theta_{JA(TOT)}$ = 9.3°C/W: LM340T-05, LM340K-05, LM309K, LM323K

3. USE:

	$\Theta_{JA(TOT)}$(°C/W)	Θ_{JC}(°C/W)	Θ_{CA}(°C/W)
LM323K	9.3	2	7.3
LM309K	9.3	3	6.3
LM340K-5.0	9.3	4	5.3
LM340T-5.0	9.3	4	5.3

Note: Here for simplicity θ_{CS} is neglected.

ADJUSTING THE OUTPUT VOLTAGE:

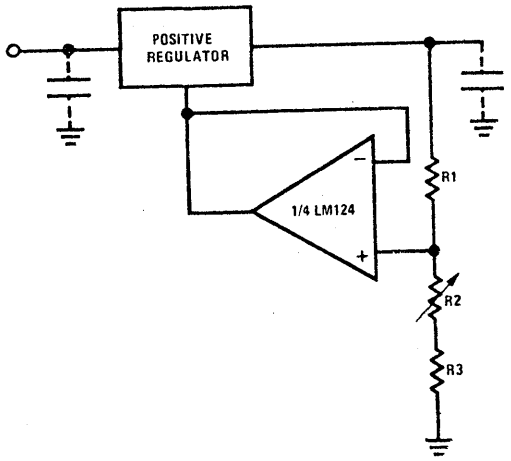


By varying the output voltage as shown, the load/line regulation degrades by the percentage of the adjusted voltage plus the drop across R2 due to I_Q variations. That is:

$$\frac{\text{Load Regulation}}{\text{1A of Load Change}} = \frac{\text{Specified Regulator Load Regulation}}{\text{1A of Load Change}} \times \left(\frac{R1 + R2}{R1}\right) + \frac{\Delta I_Q}{\text{1A of Load Change}} \times R2$$

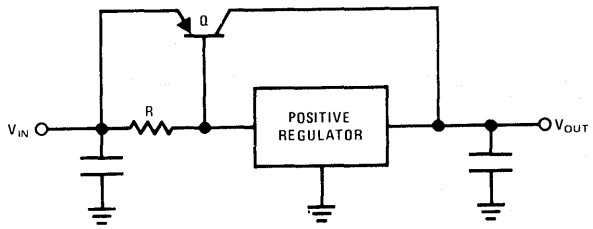
For predictable results the regulated current through R1 should be at least 5 times the maximum specified I_Q, which implies a certain power loss.

A BETTER APPROACH !!!
BUT WE NEED AN OP AMP



The overall performance is improved because the low output impedance of the op amp compensates for ΔI_Q variations. Also the current through R1, R2, R3 can now be very small (< 1 mA) and consequently there is no loss of available power. If negative supply is available the circuit can be modified so that the regulator adjustable output range will include ground. (See Voltage Regulator Handbook for details.)

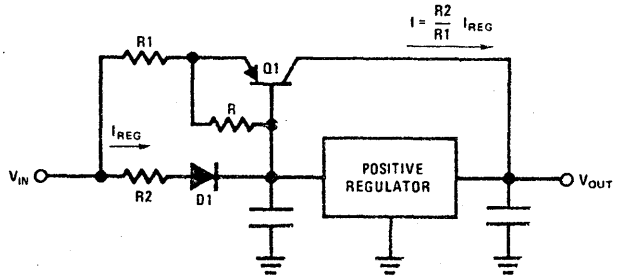
BOOSTING POSITIVE REGULATORS REQUIRES POWER PNP



PNP NOT PROTECTED

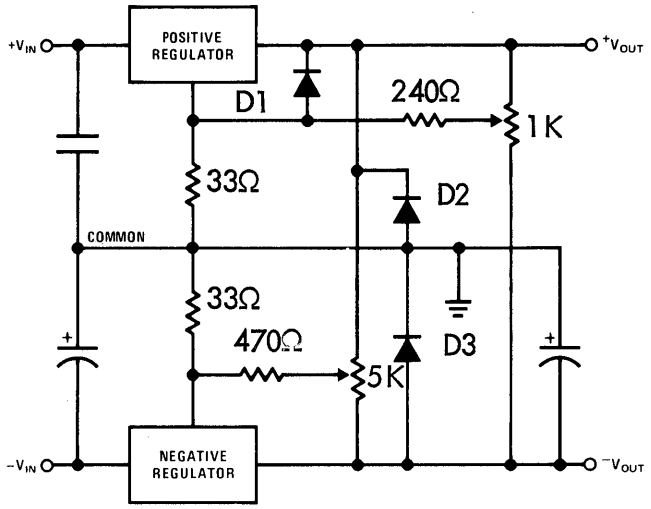
This is the simplist way to boost a 3 terminal positive regulator (we need an NPN for negative regulators). The PNP is not short circuit protected. An unexpected failure mode can occur in the following way: if the load is removed, the transistor is going to saturate before turning "OFF" and force approximately the input voltage to the output of the regulator. The regulator will experience a catastrophic failure. This is not true for the LM120 series.

CURRENT SHARING PROTECTS PNP TRANSISTOR



Using a current sharing technique, the short circuit current of the PNP is (R2/R1) I_{SC} (regulator) and the external pass transistor is protected. The disadvantage of the circuit is the use of low value power resistors R1, R2.

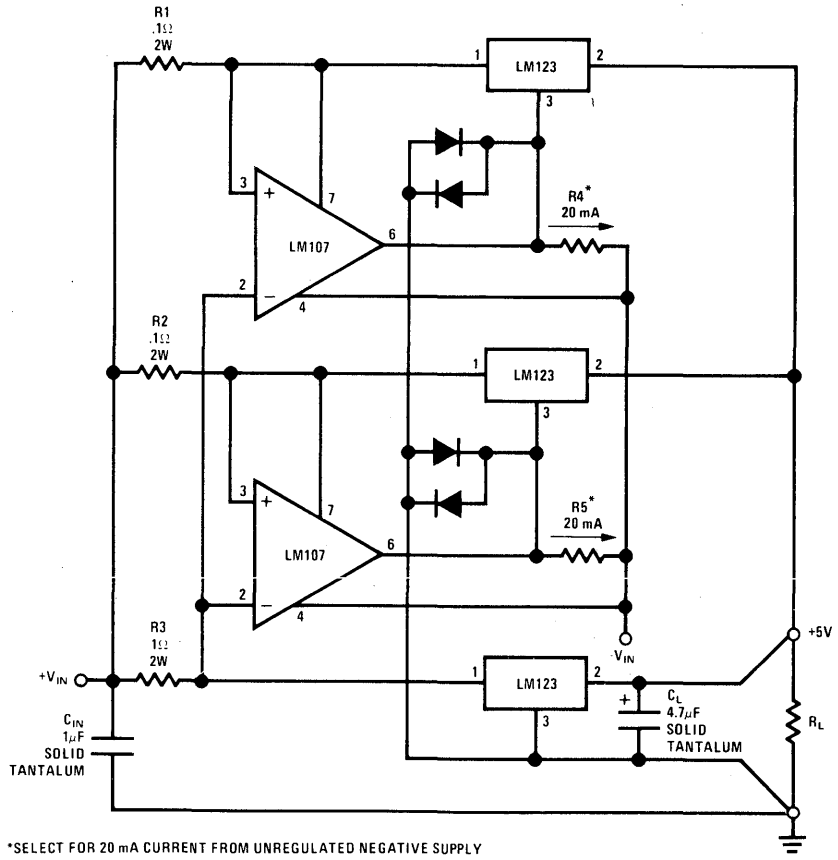
DUAL SUPPLIES CAN BE TRIMMED



- NO TRACKING ACTION

By using both the positive and the negative output of a dual supply each regulator output can be trimmed below its nominal value. Diode D1 is a germanium diode and helps the positive regulator start with a negative load greater than 600μA (assuming that the negative regulator has already started). Diodes D2, D3 are for protection purposes and should be rated at the full output load.

10 AMP REGULATOR

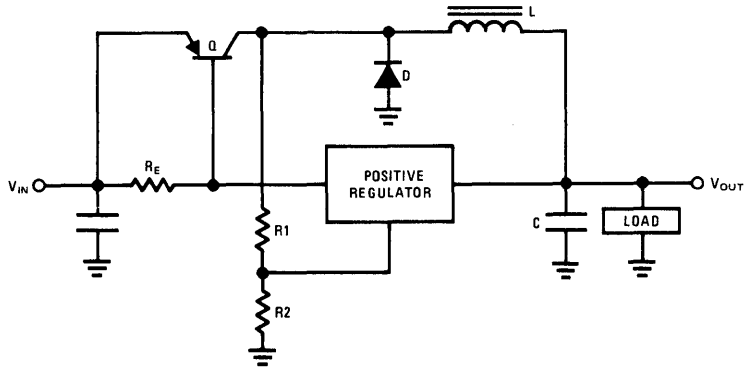


*SELECT FOR 20 mA CURRENT FROM UNREGULATED NEGATIVE SUPPLY

The overload protection built in the monolithic 3-terminal regulator makes them very attractive for high current operation if there is an elegant way to parallel them. The major difficulty to the parallel operation is the problem of current sharing between devices. Variations in V_{OUT} will cause some regulators to be in current limit while others are conducting no current at all. If the current limited devices dissipate enough power, they will temporarily shut-down and then begin cycling "ON"-"OFF." The drawback is obvious: the reliability of the power supply will degrade.

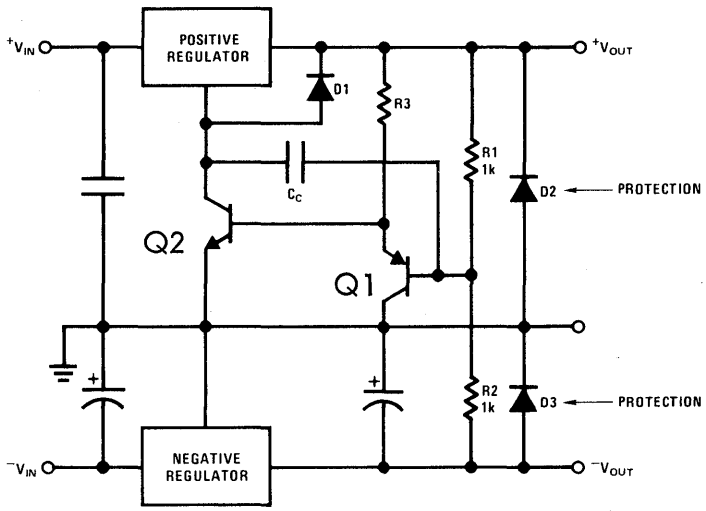
The above schematic shows a way to connect three LM123 in parallel without having the problems already discussed. The price is complex circuitry! The two operational amplifiers together with resistors R1, R2, R3 control the ground pin potential of the regulators, such as all three of them conduct current simultaneously regardless of their output voltage tolerance.

SWITCHING REGULATOR



Self-oscillating mode switching power supplies can be made using 3-terminal regulators. Positive feedback through resistive divider R1, R2 is applied to the ground pin. With the input supply "ON," the load draws current through the regulator which turns "ON" transistor Q. Power is applied to inductor L and the ground pin of the regulator is lifted by a few mV. As the inductor current increases the regulator supplies less current to the load to finally turn "OFF" Q. The ground pin of the regulator pulls down and the inductor keeps providing load current through D until the output reaches the nominal regulator voltage where the cycle starts again. Because of the high saturation voltage of the regulator the efficiency of the power supply can barely reach 80%.

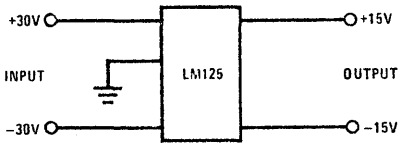
TRACKING DUAL SUPPLIES



With the 1% tolerance for R1 and R2, the positive regulator tracks the negative within 100 mV. The base of Q1 is held at a virtual ground and Q2 conducts the quiescent current of the positive regulator. If $-V_{OUT}$ falls, the collector-base junction of Q1 forward biases to raise the collector voltage of Q2 and $+V_{OUT}$ increases until V_A reaches again its virtual ground voltage. C_C is a compensation capacitor and D1 a germanium diode as already discussed.

- STARTING UNDER COMMON LOAD ?
- HOW THEY TRACK ?

YOU CAN DO EVEN BETTER:



USE NATIONAL'S DUAL TRACKING REGULATORS

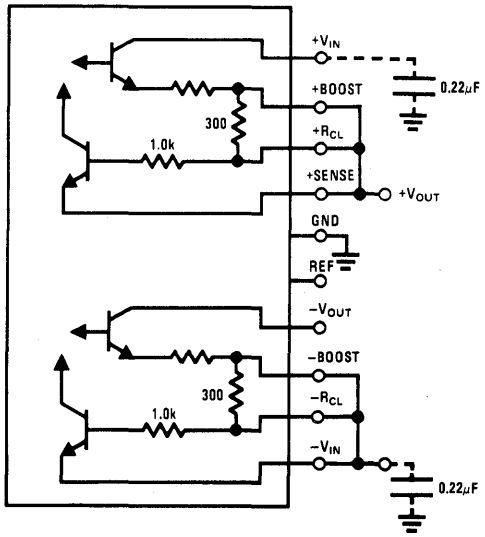
LM125

LM126

LM127

- FIXED $\pm V_{OUT}$
- + TRACKS –
- YOU CAN ADD: FOLDBACK
  CURRENT LIMIT
  BOOSTABLE
  ELECTRONIC SHUTDOWN

BASIC CONNECTION

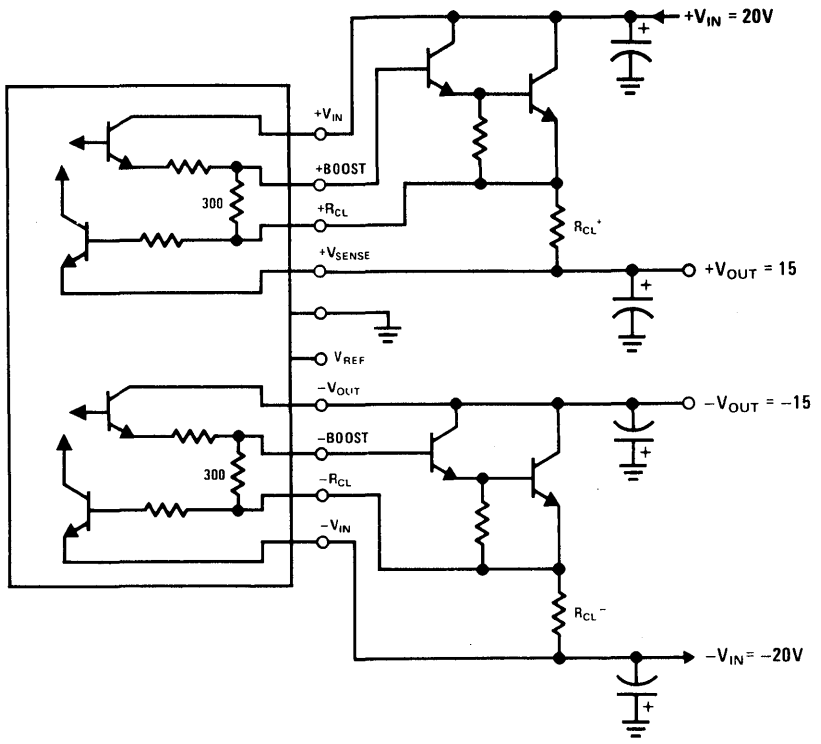


AS SIMPLE TO USE AS A 3-TERMINAL REGULATOR

Note: National Semiconductor now offers a dual tracking adjustable regulator, the LM128, with output voltages ranging from ±7V to ±28V. The low output voltage limit can be lowered further by using a single operational amplifier.

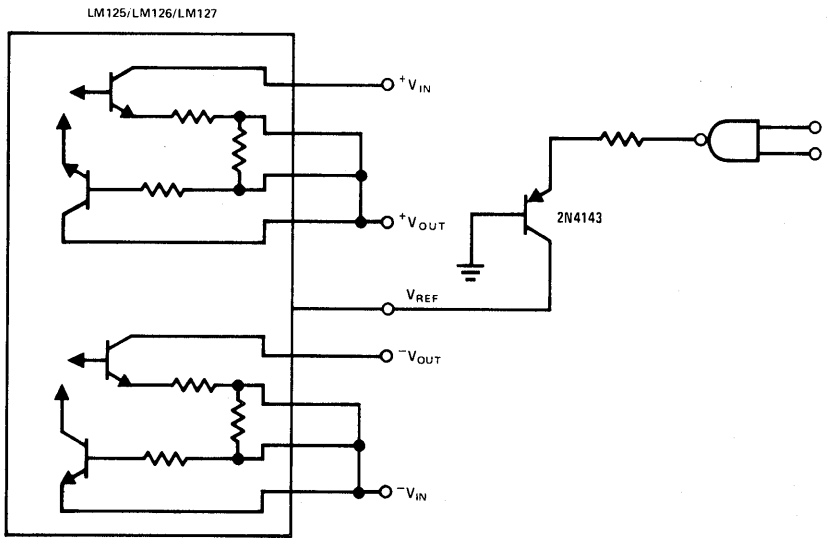
BOOSTING TO 7A

LM125/LM126/LM127



With a 2N3715 as the driver and a 2N3772 as the power transistor, the maximum output current is limited only by the power dissipation of the 2N3772 (150W max). This implies that the short circuit current of the dual supply cannot exceed 7.5A for a 20V input voltage. Foldback current limiting protection can be added to make the full output load 10A and the short circuit current 2.5A. (See Voltage Regulator Handbook.)

ELECTRONIC SHUTDOWN

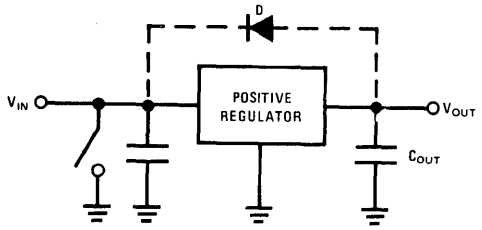


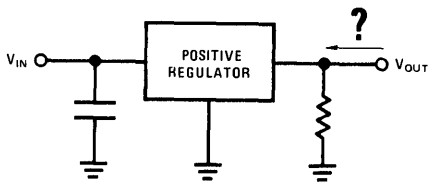
- TTL COMPATIBLE SHUTDOWN

By grounding the V_{REF} pin the positive and negative outputs are forced to 700 mV and 300 mV respectively. Both outputs are fully active so the full output current can still be supplied into a low impedance level. A complete electronic power shutdown requires more external components and it is illustrated in the application section of the Voltage Regulator Handbook.

Are 3-terminal regulators really blow-out-proof? Well, almost, but not quite. Under some conditions the regulators may be inadvertantly blown-out. However, by understanding the most common fault modes and avoiding them trouble-free performance can be achieved.

How to blow up your regulator:

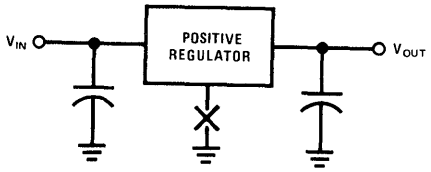


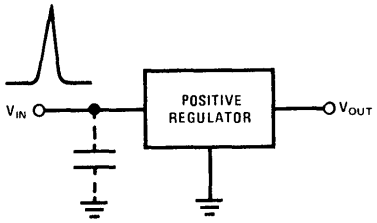


- SHORTING V_{IN} WITH C_{OUT} PRESENT
- TRYING TO SINK CURRENT

For example, suppose V_{IN} is shorted with C_{OUT} present. The input voltage will fall instantaneously to zero, and the output will try to follow it. Since the output capacitor cannot discharge instantaneously, it will force a reverse voltage across the 6.3V breakdown emitter-base junction of the pass transistor. Regulators with $V_{OUT} > 6V$ will be zapped. If this type of fault is anticipated, a fast diode connected from output to input will discharge the capacitor and protect the regulator.

In a similar manner, forcing a positive regulator to sink current (or a negative regulator to source current) will also cause permanent damage to the device.

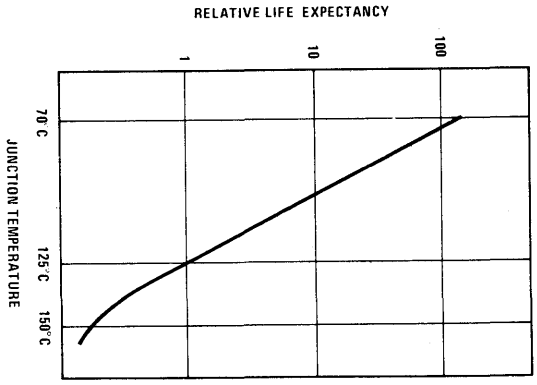




- LOSING GROUND PIN WITH C_{OUT} PRESENT
- MOMENTARY OVERVOLTAGE AT THE INPUT

Suppose the ground pin was disconnected with C_{OUT} present and the circuit in operation. (This may occur when inserting a regulator into a live test socket, if the input and output leads make contact before the ground.) V_{OUT} will pull-up to V_{IN}, charging the output capacitor close to the input voltage level. When the ground leg is connected, the capacitor will try to discharge through the regulator and the fault mode previously described will occur. Momentary overvoltages at the input can cause device failure if they exceed the maximum forward (or reverse, if the overvoltages are negative) ratings of the device. The solution to this is to use a large input capacitor, a transient suppressor, or an input choke.

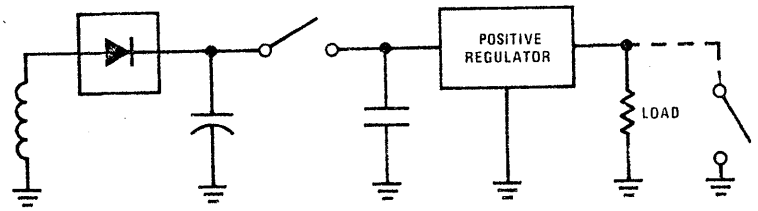
Aside from unusual circuit conditions as discussed previously, the single greatest threat to regulator reliability is heat. Most failure modes are due to die surface related effects, such as zener voltage drift due to field effect changes caused by movement of ions in the oxide. The failure rates are directly related to die temperature. Thus, the cooler you keep it, the longer it lives.



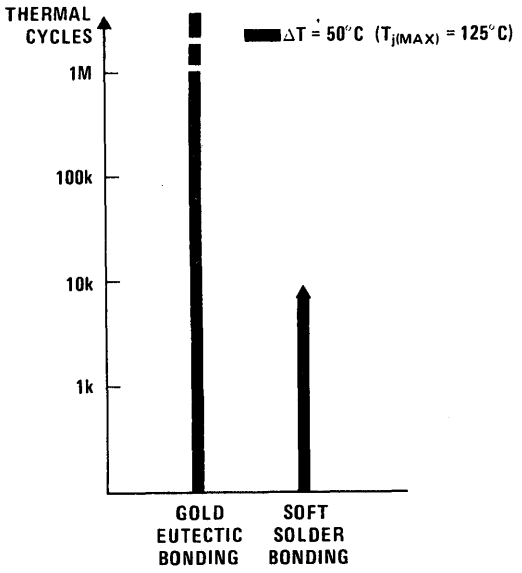
The life on any semiconductor falls off rapidly with temperature. An IC device running at a steady state T_j = 125°C would experience a failure rate approximately 145 times greater than at T_j = 70°C. The "acceleration factor" from T_j = 125°C to T_j = 150°C is 6.3 (and from 70°C to 150°C it is 145 x 6.3 = 9131).

When designing in regulators, a good rule of thumb is to allow a $T_{j\ MAX}$ of 125°C. This temperature has been found to be a suitable compromise between good reliability and realistic design practices.

THERMAL CYCLING



Thermal cycling also reduces the life of a regulator. Powering up or switching a load "ON" and "OFF" continuously subjects the die attach material to thermal stress. This stress tends to work-harden the material and eventually can cause poor die attach. The increase in thermal resistance due to poor die attach can cause over-heating and eventual failure of the die.



How important is thermal cycling? It depends on the bonding material used. Soft solder tends to work-harden more easily than other types, but it gives very uniform initial bond. Regulators bonded with soft solder are very reliable in applications where thermal cycling is kept to a minimum, such as power supplies that are turned "ON" once or twice a day and continually supply power.

Gold eutectic bonding, on the other hand, shows a tremendous resistance to work-hardening caused by thermal stress. This bonding material will typically withstand over one million thermal cycles without die bond failure (for $\Delta T = 50°C$).

15. RECENT ADVANCES IN LINEAR ICs

JIM SOLOMAN, TOM FREDERIKSEN
AND NELLO SEVASTOPOULOS
National Semiconductor
Santa Clara, CA

The temperature stabilizer on the LM199 is electrically independent of the reference. It operates over a 9V to 40V range and holds the chip temperature constant at 90°C.

The reference of the LM199 is formed by zener D3 and the emitter base junction of Q1. Active circuitry buffers the zener to give a dynamic impedance of 1Ω.

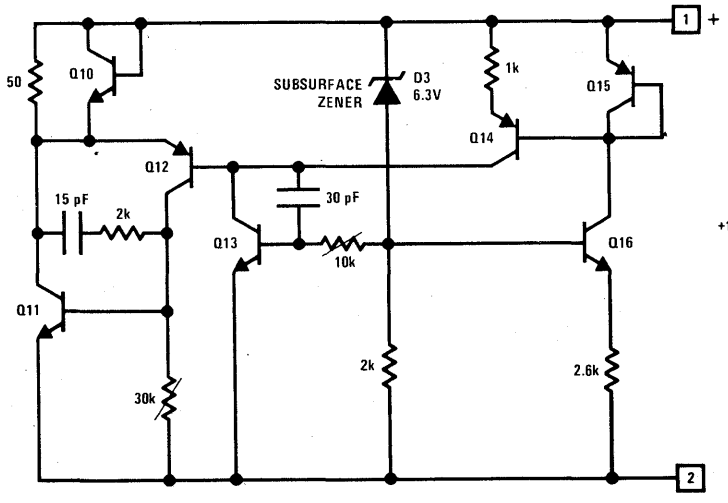
A unique thermal shield encloses the LM199. The microcircuit is actually packaged in a standard TO-46 hermetic metal can. The thermal shield minimized power dissipation as well as improving thermal regulation.

A simple buffered reference can provide a 10V reference with worst case 3 ppm/°C stability. The drift is 1 ppm/°C from the LM199, 8μV/°C from the LM108A and about 1 ppm from the resistors.

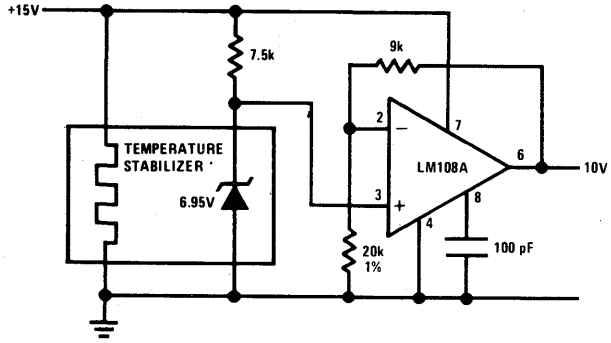
LM199/LM299 SPECS

- **ZENER CURRENT** — 0.5 TO 10 mA
- **DYNAMIC IMPEDANCE** — 1Ω
- **NOISE** — $20\mu V_{RMS}$
- **V_Z** — 6.95V ±2%
- **TEMPERATURE COEFFICIENT** — 1 ppM/°C
- **LONG TERM DRIFT** — 20 ppM/1000 HRS

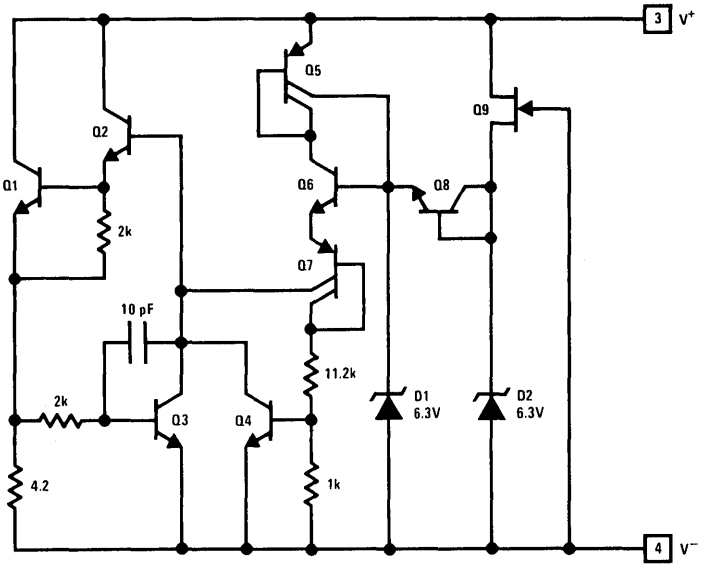
REFERENCE



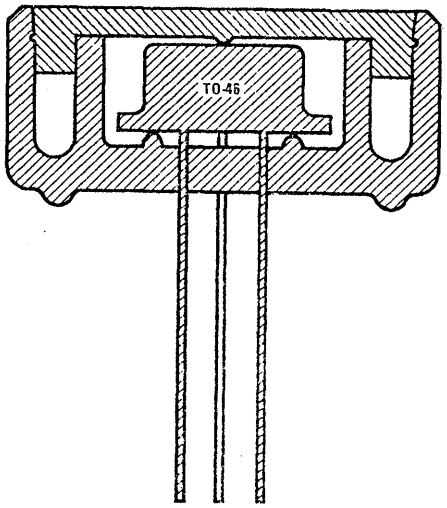
BUFFERED REFERENCE WITH SINGLE SUPPLY



TEMPERATURE STABILIZER

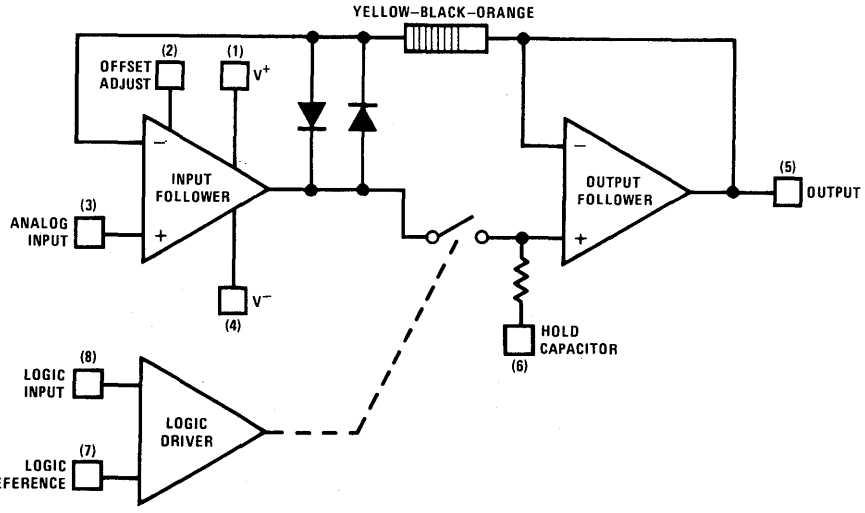


THERMAL SHIELD

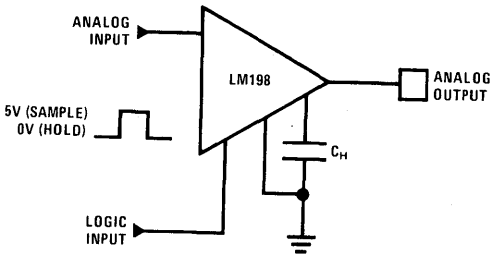


Recently introduced Bi-FET integrated circuits include the LM198 sample and hold circuit and the LF11331 analog switch family.

LM198 SAMPLE AND HOLD BLOCK DIAGRAM

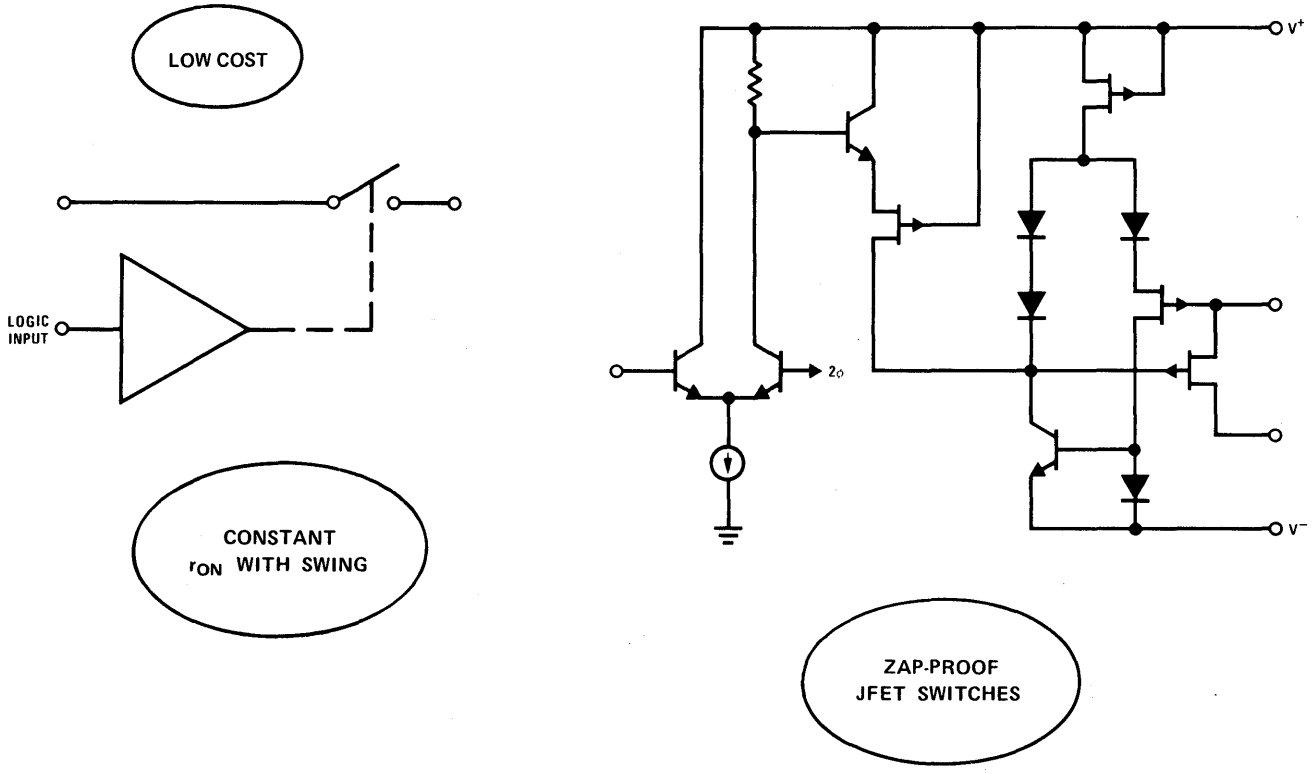


BASIC CONNECTION DIAGRAM



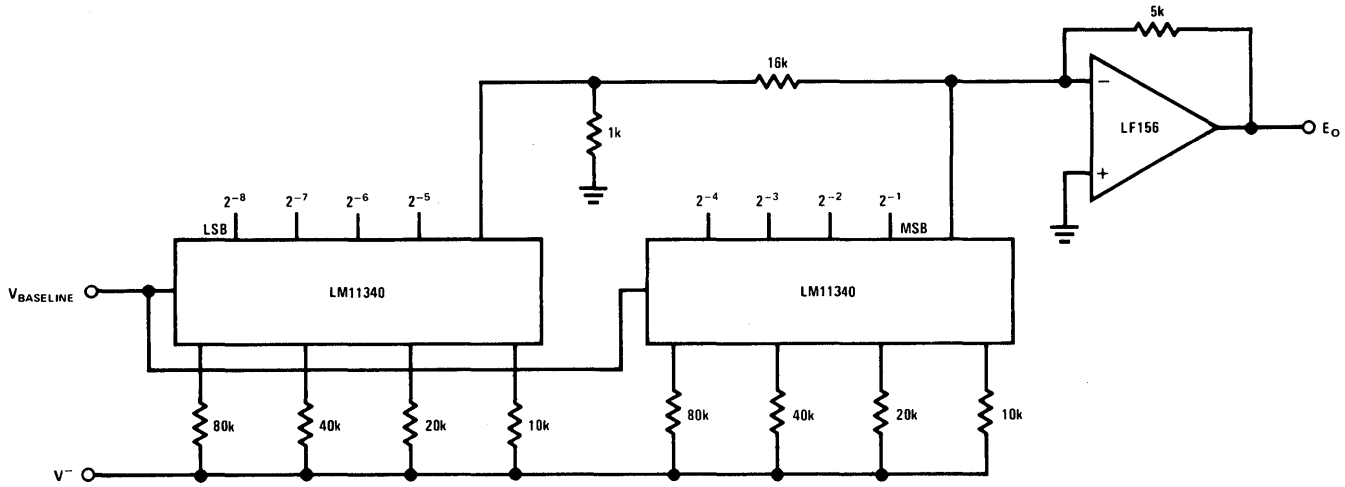
The LM198 is a monolithic sample and hold circuit. It features high accuracy with wide bandwidth and fast settling. Gain is fixed at unity with a typical error of 0.002%. Logic inputs are compatible with nearly all logic families, including TTL, PMOS, CMOS and even ±15V op amp outputs. Input impedance is typically $10^{10}\Omega$, allowing high sample accuracy with high source impedances. Leakage current on the hold capacitor is very low ≃ 20 pA). The LM198 will operate with supply voltages from ±5V to ±18V. Pin count has been held to eight, allowing the circuit to fit in a standard 8-pin TO-5 hermetic package

LF11331 FAMILY QUAD JFET ANALOG SWITCHES



The LF11331 family contains 4 independent break-before-make JFET analog switches. In contrast to CMOS devices, these switches have a constant "ON" resistance with analog swing. The "ON" resistance is specified at 200Ω max while switch to switch matching is guaranteed less than 20Ω. Other characteristics are similar to those of CMOS competition, but dangers of latch-up and accidental burn-out are eliminated. With 5 options in the LF11331 family, a variety of input logic controls are available. Three of these options are provided with a single disable pin which opens all switches regardless of logic input. The other two options are pin compatible with available CMOS parts.

QUAD CURRENT SWITCH LM11340



8-Bit D/A Converter

LOW THERMAL FEEDBACK

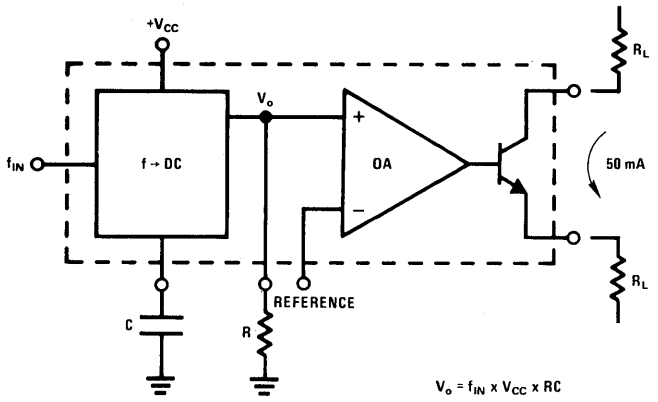
100 ns CONVERSION TIME I-MODE

12-BIT ACCURACY AND LINEARITY

The LM11340/LM11341 are high speed 4-bit current switches designed for use in D/A and A/D converters having accuracies up to 12 bits. The relative binary weighting of each of the four switch current magnitudes is fixed by an external precision resistor. The weighted currents are summed internally to produce a single output current proportional to the logic input code, with either binary or BCD output weighting user selectable. Also, combining the reference transistor with external circuitry provides compensation for β and V_{be} variations with temperature. Both TTL and CMOS compatibility result from an unusually low logic input sink current of 100μA or less.

Logic levels of the LM11340, which are ground referenced, are independent of baseline voltage variations. Logic levels of the LM11341 are baseline referenced: maintaining the correct levels requires a nominal baseline voltage setting of −4.35V.

LM1907 TACH/SPEED SWITCH



The LM2907/LM2917 low cost tachometer series provides the most versatile frequency to voltage converters yet. Features include frequency doubling with only one resistor/ capacitor pair, an output that swings to ground (0.0V!) When the input frequency is zero, a simple, easy to use expression describing its function:

$V_{OUT} = f_{IN} \times V_{CC} \times RC$, with 0.3% linearity

And a fully protected tachometer input (±28V!); perfect for magnetic variable reluctance pickups! At the output you have an op amp driving a 28V/50 mA uncommitted NPN transistor! Your load can be ground referred, referred to V_{CC} or any voltage above V_{CC} up to 28V!

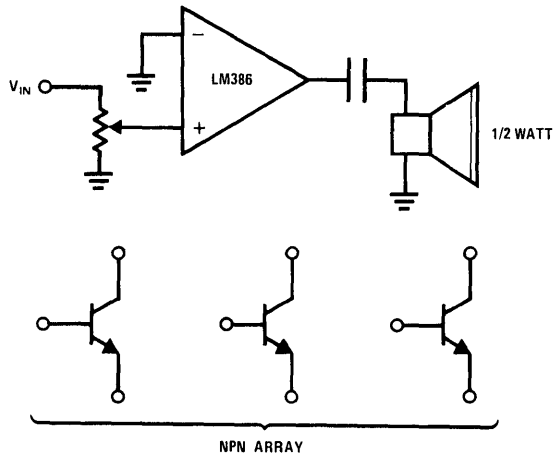
But one part can't fit every need so there are four:

You can have:

- Single or differential tachometer input
- Both uncommitted op amp inputs
- Zener regulated V_{CC} (LM2917)

See data sheet for part numbers.

LM389 = AUDIO AMP + 3 TRANSISTORS



The LM389 is an array of three NPN transistors plus a half-watt audio amplifier, all on the same chip. The amplifier is similar to the LM386. The LM389 has all the active components necessary to make an AM radio, tape recorder, phono amplifier, siren, etc. The LM389 reduces board area, cost and inventory. Almost every audio amplifier needs three transistors with it, and the LM389 is the answer.

The LM148 is the industry's only true quad 741 containing exactly the same input and class AB output circuitry as a single 741. Because of this, all applications which work with the single can be directly applied to the new quad, with no danger of operating quirks. In addition, each amplifier has a typical current drain of only 0.5 mA/amp (3x lower than the single and lower than all competition having class AB outputs), has 5x lower input bias current than a 741 and has extremely low crosstalk between amps.

The LM149 is a quad op amp, identical to the LM148 except the compensation capacitor is reduced by x5. This results in much improved gain-bandwidth, slew rate and power bandwidth as noted. Strictly speaking, the device is stable only for $A_V \geq 5$, but operation can be easily extended to +1 or −1 by adding one resistor.

LM741 QUAD

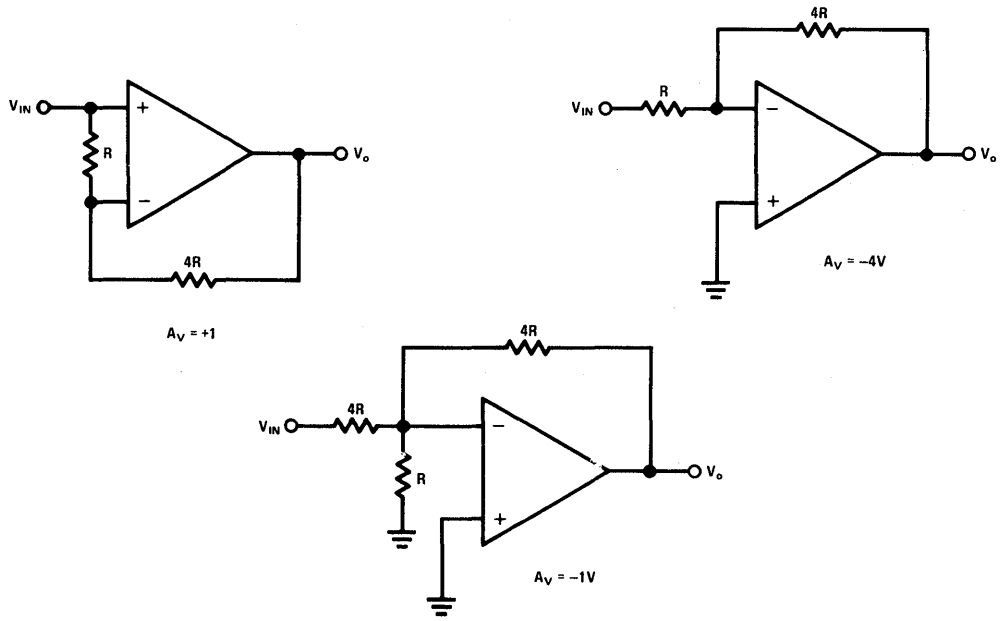
- 0.5 mA/AMP CURRENT DRAIN
- AB OUTPUT STAGE
- CORRECT INPUT CHARACTERISTICS

LM149

A "BROAD BANDED" LM148

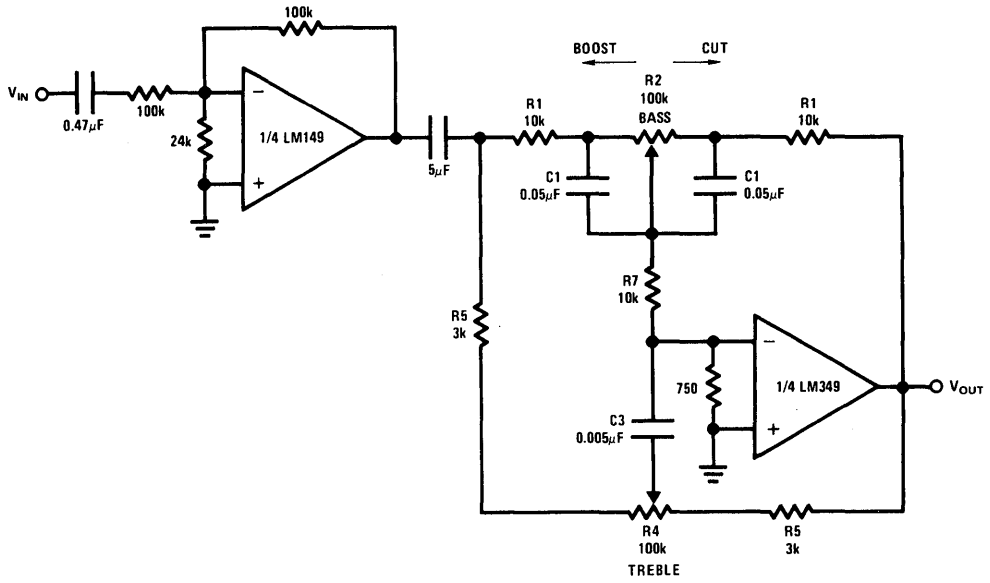
- GBW = 3 MHz
- SLEW RATE = 2.5 V/μs
- POWER BW = 50 kHz
- VOLTAGE GAIN = 5 (MIN)

THE LM149 GIVES 50 kHz PBW AT ANY GAIN



Use of the LM149 for various gains. Normally $A_{V\ MIN} = +5$ or -4, but as shown $A_V = -1$ or $+1$ is also possible. Under these low gain conditions, the device exhibits the fully improved slew rate and power bandwidth, but has a higher output dc offset by x5 compared with a normal LM148.

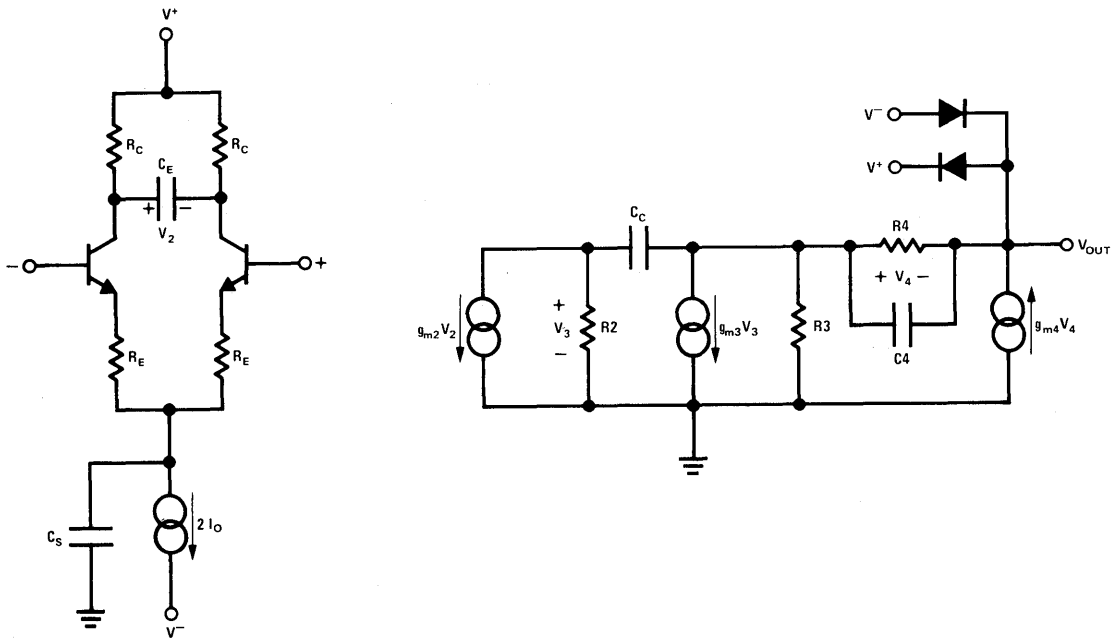
ACTIVE TONE CONTROL



FULL OUTPUT SWING TO 20 kHz

Extremely high quality audio applications are finally possible using op amp approaches. With the LM149 high THD caused by the normal 741 10 kHz power bandwidth is eliminated, and less than 0.05% THD is possible beyond 20 kHz.

OP AMP MACROMODEL FOR COMPUTER ANALYSIS



- ALLOWS COMPUTER SIMULATION OF COMPLEX SYSTEMS
- REDUCES COMPUTER MEMORY AND RUN TIMES BY X10
- MODELS AVAILABLE FROM NATIONAL FOR LM148/LM149, LF155/LF156/LF157, LM118

Computer simulation of systems using more than one IC has been very costly and often impractical in the past. The problem can be seen by noting that each transistor model in a simulation contains approximately 12 elements, each amplifier contains 30–40 transistors and a typical modern system uses at least four amplifiers plus other circuitry. Simulation of this "average" system requires handling of 1800 to 2000 elements, a task that exceeds the memory capacity of many large computers and leads to costs per run of $500 to $1000. Using National's new "macromodels" to represent the IC's, complexities can be reduced by greater than x10 and run costs reduced by x50. Complete dc nonlinear, ac small signal and large signal transient responses are accurately simulated by the macromodel. See following table and accompanying paper.

TABLE OF OP AMP MACROMODEL PARAMETERS

	β_o	R_E KΩ	R_C KΩ	C_E pF	C_S pF	I_O μA	g_{m2} μ℧	g_{m4} m℧	C_c pF	R_3 KΩ	R_4 KΩ	C_4 pF
LM741	100.5	2.754	5.305	10.5	4.66	10.05	188.5	7.72	30	106.1	2.0	250
LM301	36	2.918	5.767	10.5	6.0	9.0	173.4	7.72	-	92.26	2.0	159
LM308	3000	7.565	13.26	5.46	12.86	4.5	75.4	3.86	-	398	7.77	136
LM318	2083	1.887	1.99	2.09	2.04	250	502.7	30.89	5	39.8	.970	55
LM324*	40	12.28	26.53	3.0	1.2	1.8	37.7	3.86	6	265.3	3.1	342
LM348	300.0	3.046	5.895	10.5	6.0	9.0	169.7	7.72	30	88.42	2.0	250
LM349	300.0	3.046	5.895	21.0	6.0	9.0	169.7	7.72	6	88.42	2.0	250
LM355*	$1.0 \cdot 10^6$	5.512	6.366	1.34	2.0	30.0	157.1	57.9	10pF	127.3	1.38	38
LM356*	$2.5 \cdot 10^6$	3.195	3.537	1.34	2.5	75.0	282.7	77.2	10pF	70.74	1.04	25
LM357*	$2.5 \cdot 10^6$	3.195	3.537	2.7	2.5	75.0	282.7	77.2	2pF	70.74	1.04	25

NOTES:

Use Ebers-Moll Transistor Model, spec. only β_o & $I_S = 10^{-15}$

For diodes use only $I_S = 10^{-15}$

* These amps use pnp input transistors, the rest use npn inputs.

LM741 macro can be used for LM1458, LM747, or LM358 and 301 for 307.

$R_2 = 1.3M\Omega$ and $g_{m3} = 7.72mV$ for all amps

16. EXPLODING THE ADDRESS MULTIPLEXING MYTH

BRUCE THREEWITT
Manager, MOS Applications
Fairchild Semiconductor
Mountain View, California

Since the introduction of the 4096-bit read/write RAM, semiconductor memory costs have been comparable with those of core memories. The transition from core designs to semiconductor designs has been painful but inevitable. Part of the pain has been caused by a lack of one industry-standard 4K RAM. The competitive nature of the semiconductor industry has hindered the development of a standard memory component until recently. Users have applied sufficient pressure to the semiconductor industry to cause the proliferation of 4K RAM versions to cease. The definition of the 4K RAM is beginning to converge around a multiplexed address approach and one particular non-multiplexed address approach.

A brief history of these two arenas would help outline the evolution of the present versions of the 4K RAM. The first Monolithic 4096-bit RAM was introduced in February of 1972. It was a relatively slow device utilizing an 1103-like cell (3 transistors per cell). This device, called the 2107A, was offered in a 22-pin package which, at that time, was not a standard package. The package pin spacing was 0.4 inches, not well-suited for wirewrap systems or automatic insertion.

The next version was introduced by another company in the same pin-outs based on a single transistor cell. This device was more nearly speed-compatible with what the computer mainframe builders required to replace core. Soon thereafter the 2107A was followed by the 2107B, still pin-compatible but based on a single transistor cell.

At this point it was clear that at least one multisourced version was emerging as an industry standard. Furthermore, the single transistor cell appeared to be the favored technique for achieving the required density. Other advantages offered by the single transistor cell include better noise immunity, less power, and higher speed. To achieve the requisite speed, power, and low cost the N-channel MOS Technology seemed necessary. This necessity limited the contestants in the 4K RAM competition to a few semiconductor manufacturers. Thus, a combination of circuit and process innovations were required to make the 4K RAM possible. This use of a new technology contributed to the pain of transition from core, a well-known magnetic technology, to an esoteric state-of-the-art semiconductor approach for dense, high-capacity computer memories.

Meanwhile, another bid was made for an N-channel 22-pin version called the 6605. This device was introduced by two semiconductor houses in active co-operation to alleviate the multisource problem. Meanwhile, a novel approach to the architecture of the device allowed a 16-pin version to be introduced, called the MK4096. The architectural change involved time-multiplexing the address lines. Since twelve addresses are required to uniquely access 4096 bits, two pairs of six addresses could be supplied to the device. Since the cell matrix was organized in 64 rows and 64 columns, supplying the row addresses first and the column addresses next was easily implemented with on-chip latches and simple logic. This multiplexing technique caused no penalty in access time because the row address path was considerably slower than the column address path. Thus, while row addresses were propagating toward the cell matrix, the column addresses were entered and used to select a particular sense amplifier and prepare the output latch for data.

The advent of the 16-pin 4K RAM caused panic amongst the 22-pin suppliers. They began

looking for reasons why the address multiplexing would cost more money or consume more power or result in a major performance penalty. Of course, none of these drawbacks really apply to address multiplexing as we will see in a later section. Nevertheless, the myth against multiplexing was born.

To further combat the inroads, the 16-pin 4K RAM was making in the market, competitors modified their 22-pin approach to make it fit in 18 pins. This package offered some of the same board space savings as the 16-pin package since it had 0.3-inch pin spacing. In fact, it is only about 10% bigger than the 16-pin package. To achieve 18 pins, however, required some significant system performance sacrifices. Figure 1 shows a comparison of the 22-pin, 18-pin, and 16-pin packages. Notice that the 22-pin design only consumes 21 pins. To reduce the pin count to 18 requires the elimination of the $\overline{CS}$ pin (the TTL-level chip select), the combining of the input and output to form an I/O structure, and the elimination of the Vcc (+5-volt) power supply pin. Vcc is used only on the output buffer to provide an active pull-up. Its elimination implies a bare drain output structure with an external passive pull-up. These changes reduce the count to 18 pins.

Two problems are immediately apparent. First, to select an 18-pin RAM requires manipulating the Chip Enable clock input. This input requires a 12-volt swing. It is not TTL-compatible. Secondly, an external resistor is required to pull the output to a logic "1". This characteristic reduces the load-driving capability of the 18-pin RAM. Also, read and write operations require careful timing to avoid data overlap problems on the I/O pin. Usually, an external bidirectional three-state buffer is required to control the data interface to the RAM. This circuitry is shown schematically in Figure 2.

A less apparent problem is that the 18-pin design does not allow future RAM designs, specifically the 16K RAM, to use the same pin outs. Also, the same twelve address lines must be distributed throughout the memory matrix as with the 22-pin design. The cost comparison, later in the discussion, will show the adverse cost impact of this characteristic in both board space and component/pin count.

An examination of the reasons for using address multiplexing is now in order. In offering the 4K RAM in 16-pins, a large savings in board area is realized compared to the 22-pin approach. The 16-pin package is only one-half the size of the 22-pin package. Another important area factor is often overlooked. The multiplexed-address approach requires only one-half the number of address lines required for the 18- or 22-pin approaches. In double layer printed circuit boards, the address traces account for a significant amount of area -- nearly as much as the memories themselves.

Another benefit from having fewer address lines is that fewer address drivers are needed (half as many, in fact). This fact results in reduced peripheral components, reduced peripheral power, and fewer interconnects for higher reliability. Fewer address lines also improves noise susceptibility since only half as many signals are changing state at any given time.

The 16-pin 4K RAM offers the most RAM per pin currently available. Since the solder system is a major contributor to reduced reliability, the 16-pin 4K RAM offers the potential for higher overall system reliability. In addition, the automatic insertion and DIP handling equipment in use today is based on standard 16-pin packages. Since the 4K RAM is aimed at large volume mainframe use, its 16-pin nature could be of major importance.

Finally, looking to the future, indications are that the 16K RAM will be offered in the 16-pin package with the same pin-outs as the 4K. The highest order address takes the place of the $\overline{CS}$ pin. Chips are selected by decoding the TTL-level Row Address Strobe. This technique can also be applied to the existing 4K RAM to conserve power.

In all fairness, there are a few liabilities in address multiplexing. The only significant one is the increased complexity of the timing. Since addresses are entered into the RAM in two sections, there must be two address strobes (clocks) to properly synchronize these inputs. However, the two-phase timing is made easier by the input voltage and timing characteristics of the two clocks. First, the clock inputs are low capacitance: one-fifth the clock capacitance of the 18- or 22-pin approaches. Secondly, the input voltages are TTL-compatible. These two factors reduce peripheral clock power to one-sixtieth the amount required by the 18- or 22-pin designs. Some extra logic is required to generate two phases. Since the clocks are TTL-level, though, this logic is composed of simple, inexpensive TTL SSI circuits.

At first glance, there is a speed penalty for multiplexing since some time is required to switch from row address to column addresses and back. As we will see later, this penalty is actually non-existent in typical memory systems.

Figure 3 schematically illustrates one approach to address multiplexing. The "MUX" signal is derived from the clock timing to coincide with minimum address hold time and/or minimum row-to-column-strobe-lead-time (t_{RCL} on the 4096 data sheets). Sufficient time must be allowed to drive the address lines with their associated capacitance (10pF, worst case, per address input). The amount of time required depends on the particular address driver used and the number of 4K RAM devices driven by each address driver. Figure 4 shows an alternative approach using hex three-state buffers. From a logical standpoint, the and-or (multiplex) function is equivalent to OR-tied buffers with three-state enable controls.

From the previous discussion address multiplexing is clearly not the problem that 18-pin manufacturers have claimed. What follows explains how a memory system might be implemented using the 16-pin Fairchild 4096. This is one of several 16-pin 4K RAM devices in large volume production at this time. The same system is shown using the 18-pin 4K RAM for comparison purposes.

Figure 5 is a timing diagram for a typical 4096-based system. The system block diagrams are shown in Figure 6. Notice that the primary differences lie in the timing circuitry and the interface between the CPU and the memory address inputs. The 18- or 22-pin designs require a two-way multiplex to refresh the memory. The lowest-order six memory address lines are switched to refresh addresses once every 31.25 μsec (64 rows X 31.25 μsec = 2 msec). The same scheme is used in the 16-pin approach except that an additional multiplex operation switches between ROW and COLUMN address and back at the times shown in Figure 5. Notice that if the timing is done in this way, there is no speed penalty for address multiplexing.

The logic required for the 16-pin approach is indicated in Figure 7a alongside the equivalent circuitry required for the 22-pin design. The extra logic required to do row and column address multiplexing consists of one extra 16-pin package. The area savings offered by a 16-pin RAM with only six address lines makes up for this one extra package many times over.

In summary, the 16-pin address multiplexing approach offers improved noise immunity, lower power, lower cost, and improved system reliability without increasing system complexity. Apparently, major semiconductor manufacturers agree that the 16-pin approach is viable since all of them either now offer or will be offering 16-pin 4K RAMs, now the industry standard.

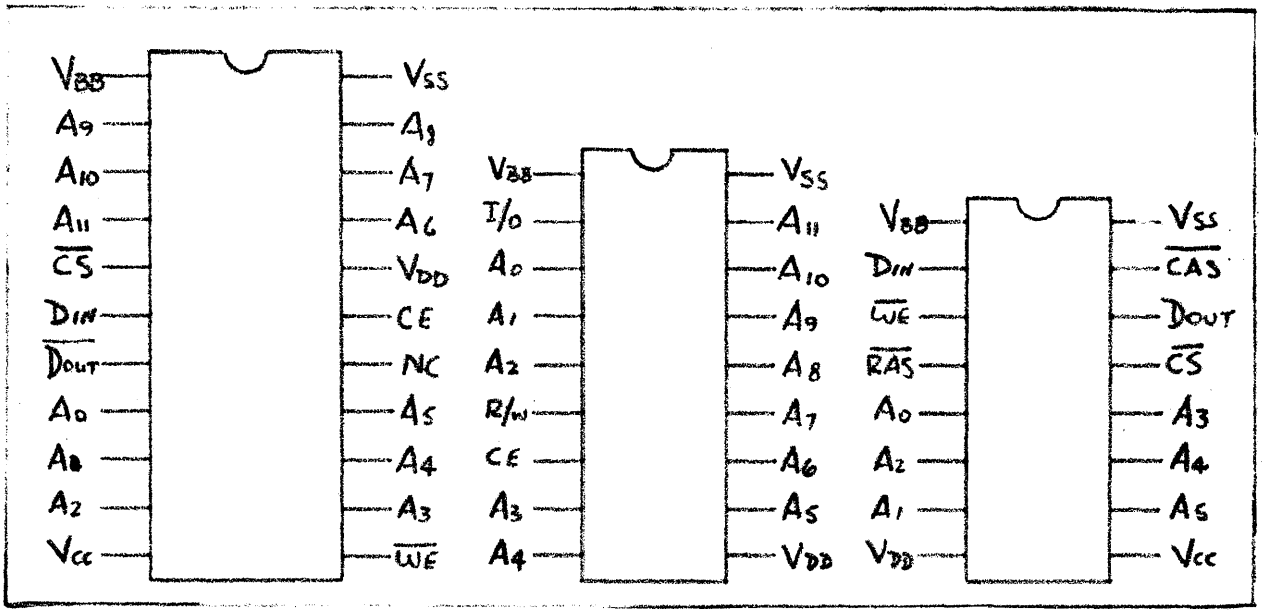


Figure 1 - Pin Designations for 16, 18, and 22-pin 4K RAMs.

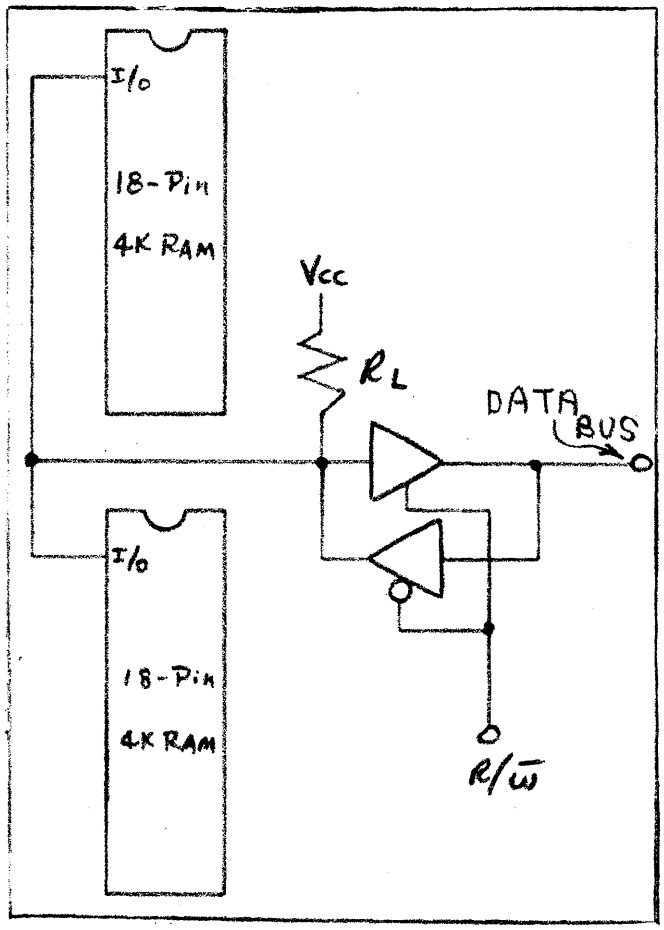


Figure 2 - Data Bus Interface for 18-Pin 4K RAM.

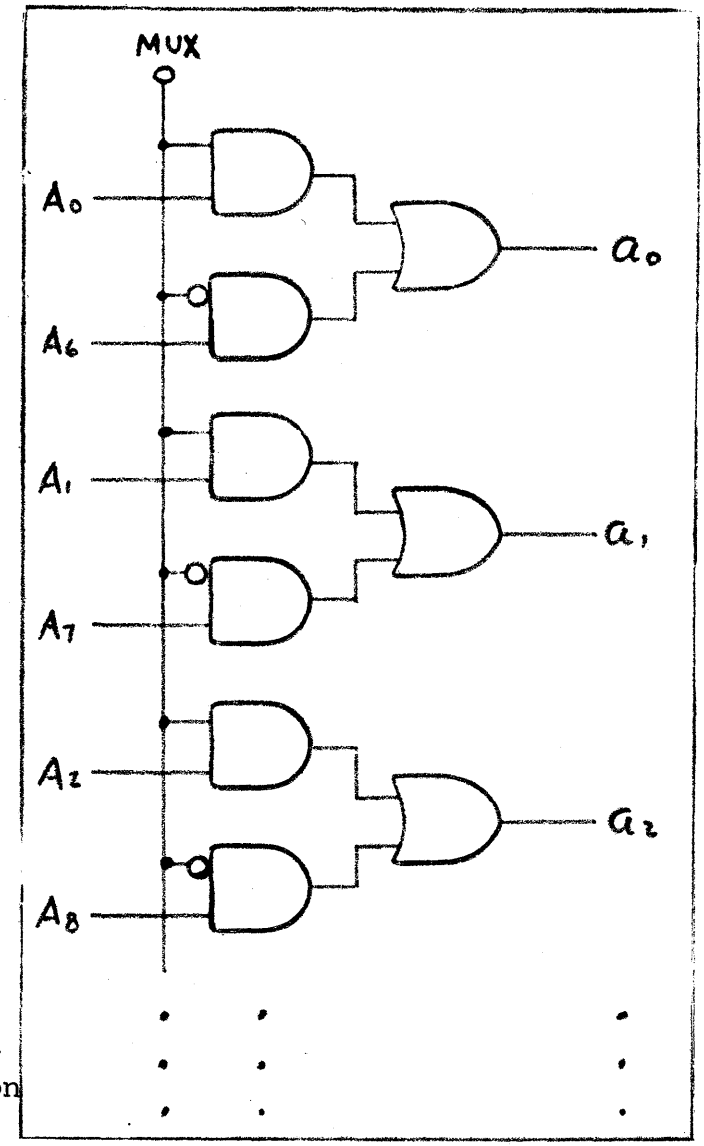


Figure 3 - Address Multiplexing Using AND/OR Function
An = System Address Signal
a_n = Multiplexed Device Address Input

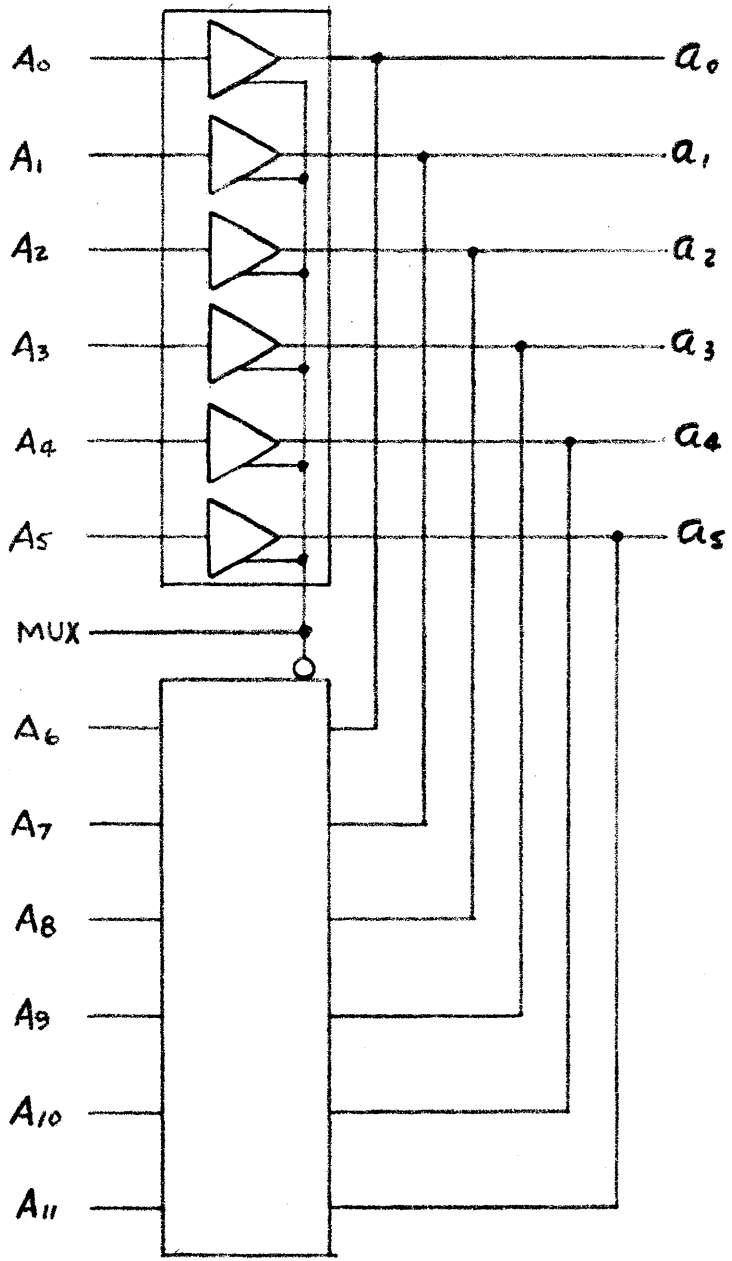


FIGURE 4 - Address Multiplexing Using 3-state Buffers

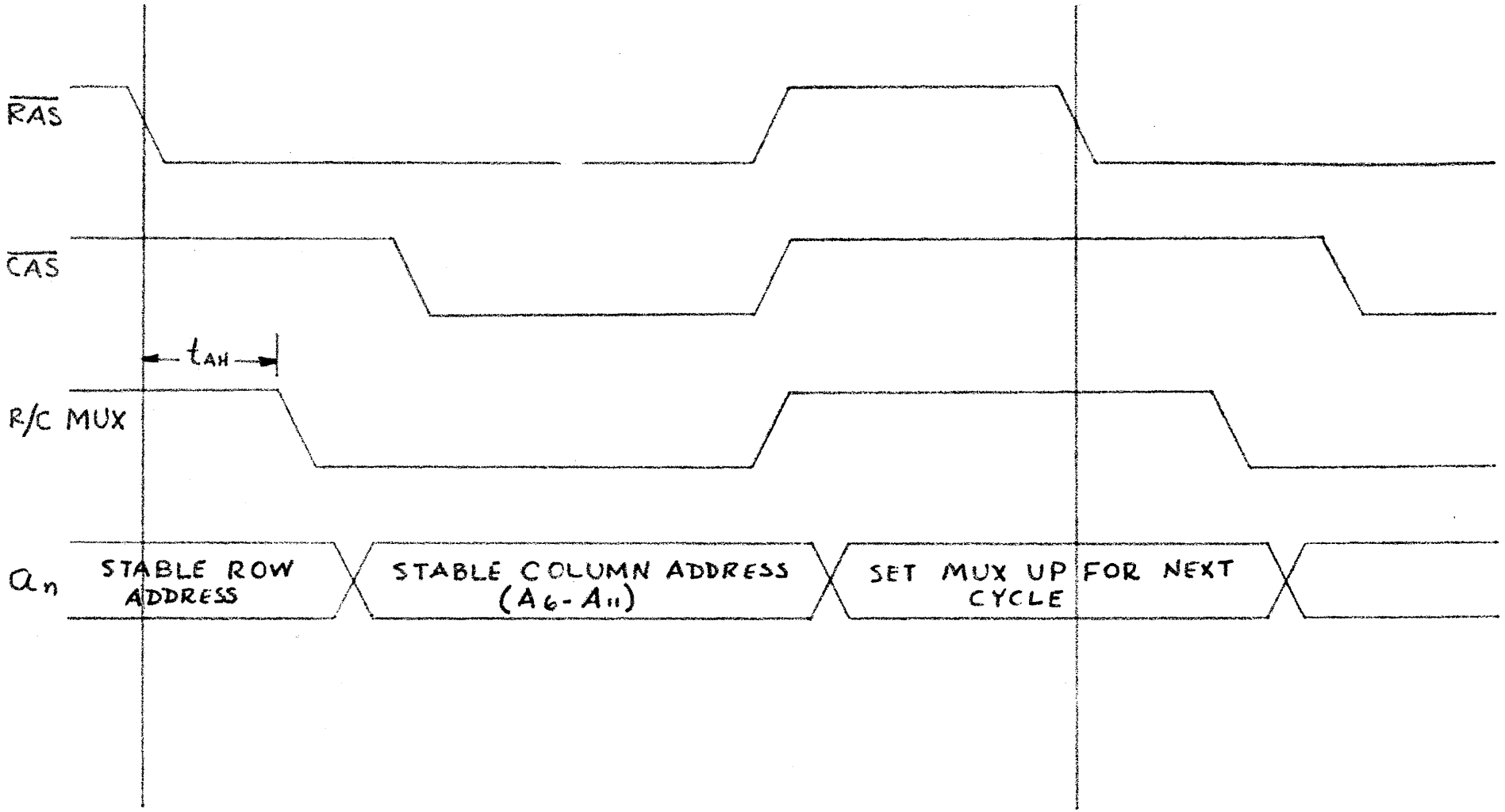


FIGURE 5- 4096 Address Timing

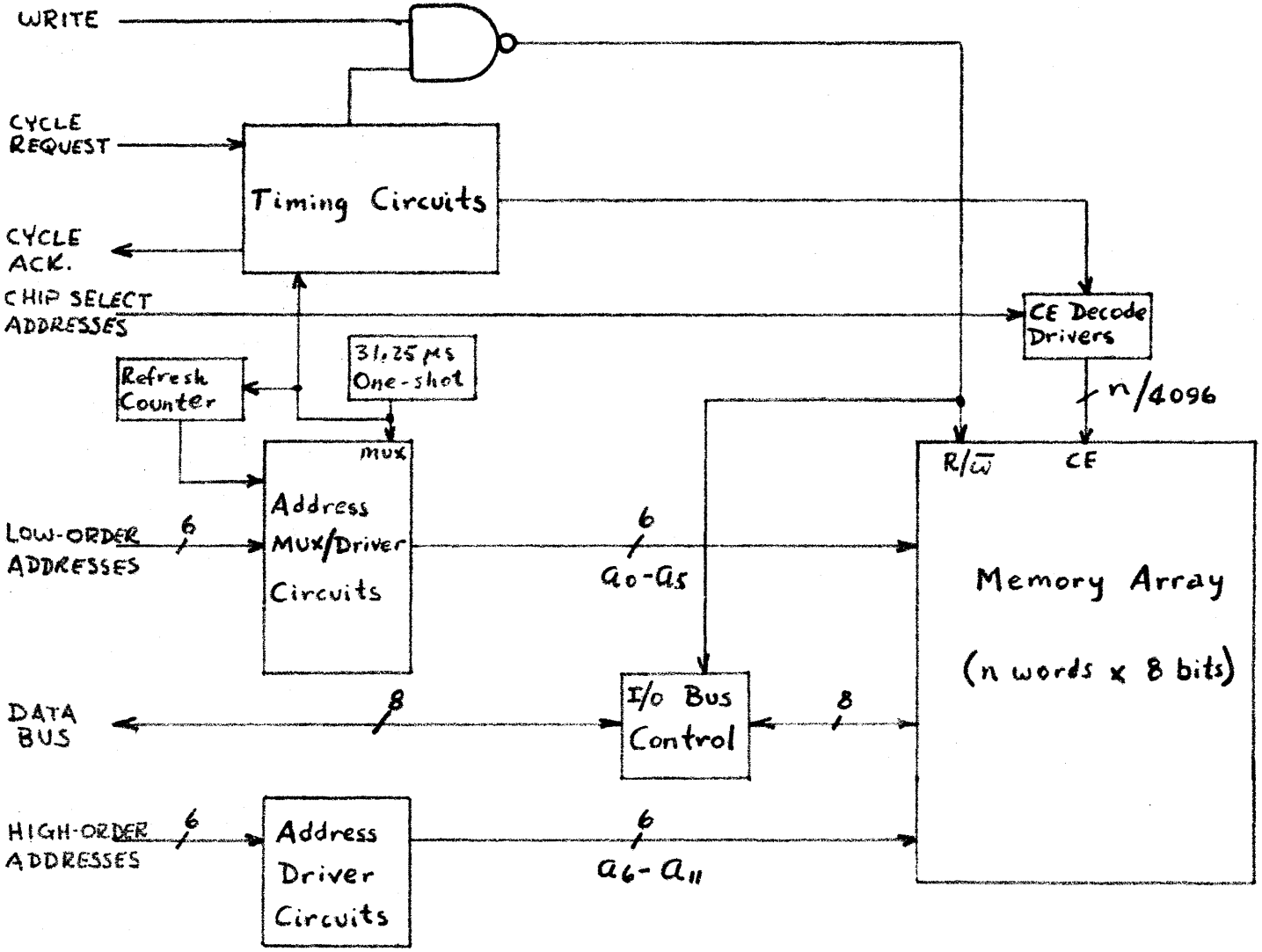


FIGURE 6a - 18-PIN 4K RAM Memory System Block Diagram

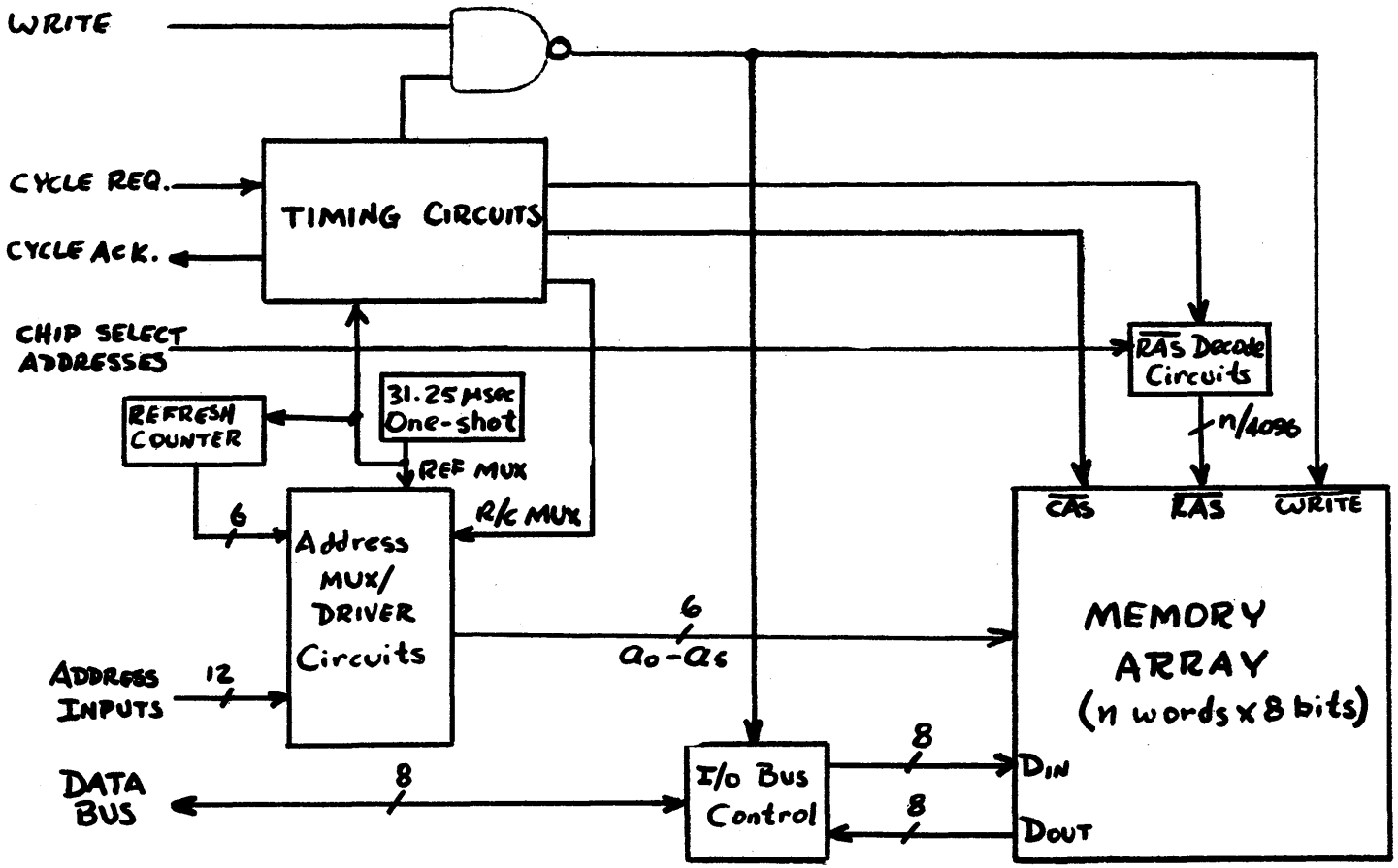
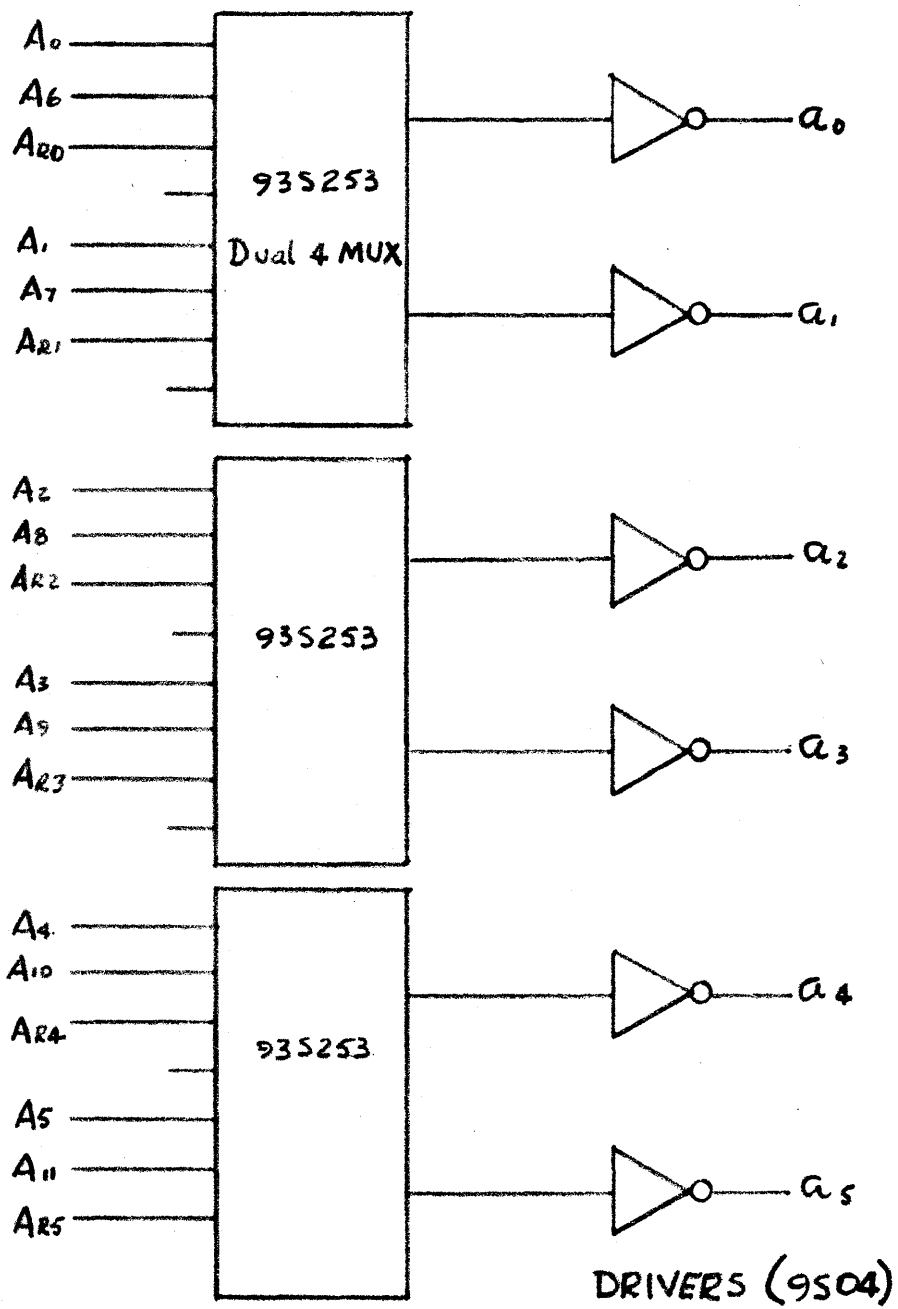


FIGURE 6b - 16-Pin 4K RAM Memory System Block Diagram



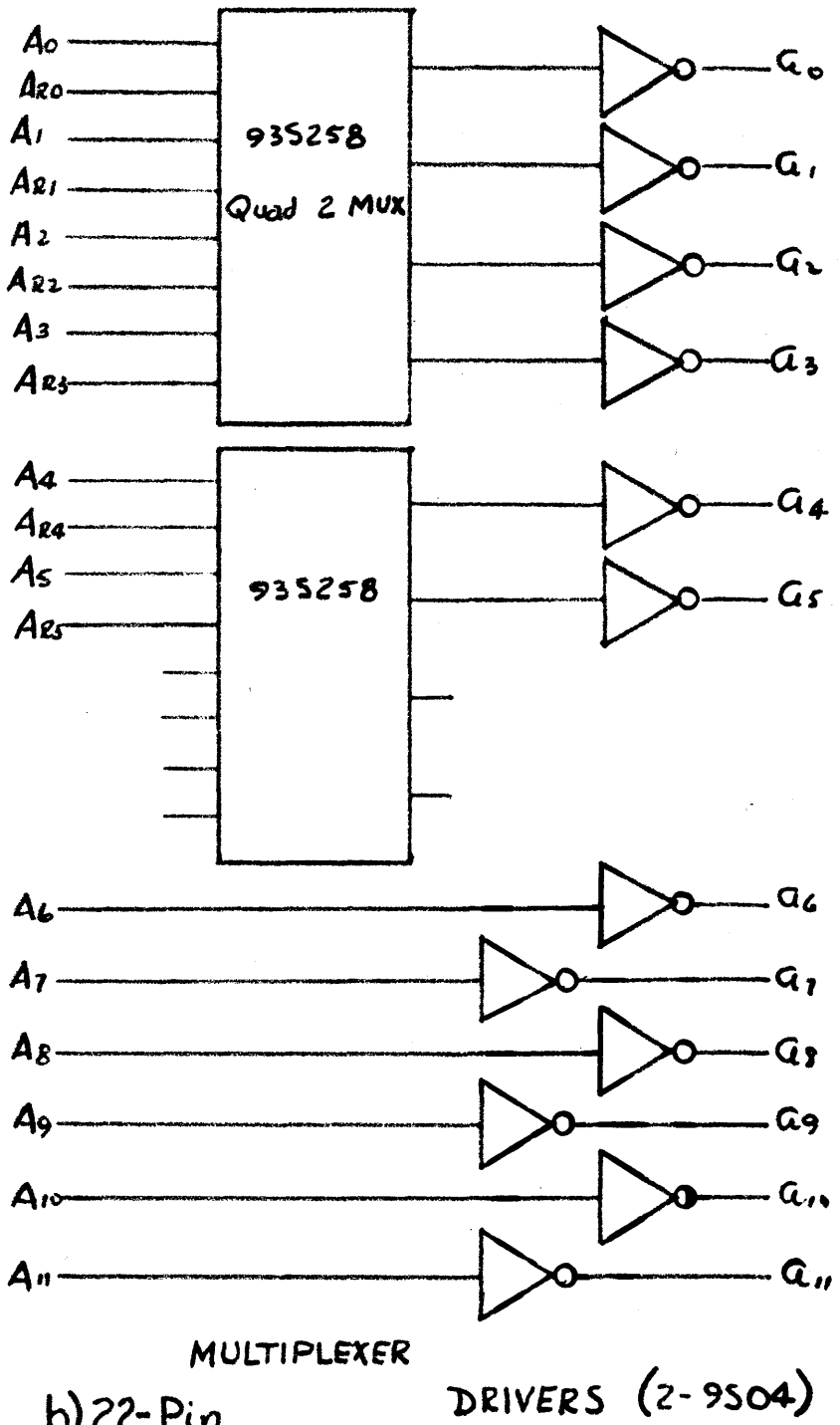


FIGURE 7-Address Mux Circuitry

17. USING THE 8700 SERIES CMOS A-TO-D CONVERTERS

SKIP OSGOOD
Linear Products Marketing Manager
Teledyne Semiconductor
Mountain View, California

In recent years the use of computers and other digital techniques has grown dramatically in systems which must interface with "real world" physical variables. Examples include industrial control, instrumentation, and data transmission and recording equipment. Since the effectiveness of any digital system is dependent upon the input it receives, accurate and reliable data converters are becoming increasingly needed to translate analog information into digital signals for subsequent processing and analysis.

Recognizing both this need and the interest of designers in reducing system cost and complexity, Teledyne Semiconductor has introduced the 8700 series of monolithic CMOS analog-to-digital converters. In this paper, we shall discuss the features of these new devices and how to use them to advantage in system designs.

Teledyne 8700 Series A/D Converters

Teledyne Semiconductor's 8700 Series A/D converters are available in three versions with 8-bit (8700), 10-bit (8701) and 12-bit (8702) resolution. Each device features latched, binary-coded outputs and is constructed as a single-chip complementary metal oxide semiconductor (CMOS) integrated circuit. Packaging is 24 pin ceramic DIP, and operation is specified over $-40^{\circ}C$ to $+85^{\circ}C$. Conversion is performed using an integrating, incremental charge balancing technique which has inherently high accuracy, linearity and noise immunity.

The 8700 chip includes all of the active devices required to perform the analog-to-digital conversion function. As few as six passive components--four resistors and two capacitors--are all that is needed to complete the circuit, and only two of these (the input and reference resistors) have critical tolerance requirements. Unlike other IC type converters, the 8700 does not require an external clock, comparator or precision resistor network.

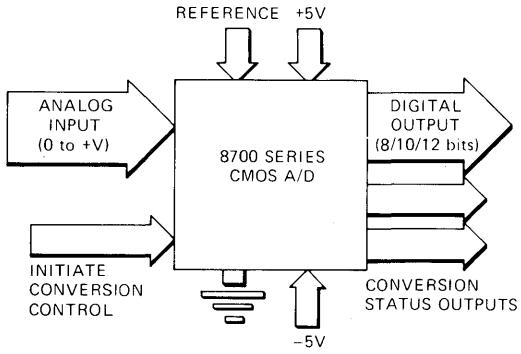


Fig. 1 8700 Series
FUNCTIONAL DIAGRAM

Functional Description

Figure 1 shows the basic functional configuration of the 8700 series devices. Inputs include the analog signal (which may be any positive voltage) and a reference. An "initiate conversion control" is also provided; this may be used to operate the device under system control or it may be held high (e.g. connected to V_{DD}), allowing the converter to free-run. Supply requirements are +5V and -5V, with a wide 3.5V to 7V tolerance.

The digital data outputs are in the form of an 8, 10 or 12-bit parallel binary word. Two conversion status outputs are also available. The "busy" line indicates when a conversion is being performed; this output normally would be monitored in the system-controlled mode of operation. The "data valid" output goes low during the approximately 5μs interval between the end of conversion and the updating of data in the output latches; the system thus is warned that data read during this interval may be invalid.

All outputs and control inputs are compatible with CMOS and LPTTL.

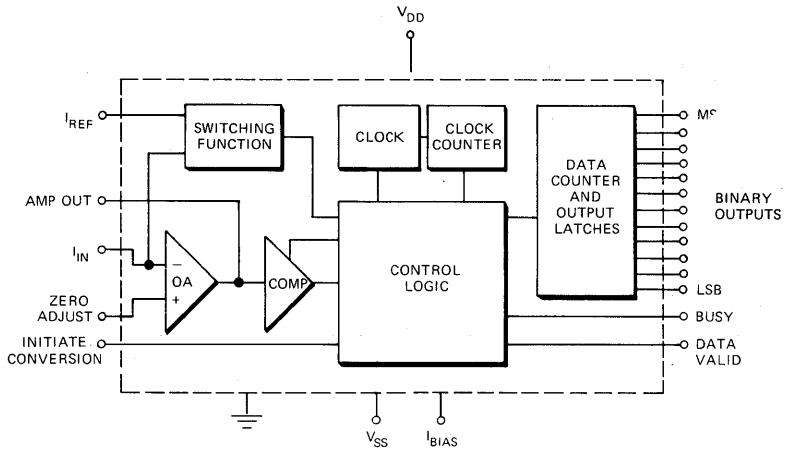


FIG. 2 8700 SERIES INTERNAL ELEMENTS

Incremental Charge-Balancing Conversion Principle

An integrating conversion principle known as incremental charge-balancing is used in the 8700 series. (See Figure 2) The unknown analog signal (changed to current by a scaling resistor) flows into the summing input and is integrated by the op amp. Fixed increments of a reference current (opposite in sign to the input) are switched into this input just often enough to keep the integral of the total input (positive unknown plus negative reference) at or near zero. The comparator and control logic determine when a reference charge increment must be added, and the total number of reference increments added during the conversion cycle is counted in the data counter. This number is related to the input voltage by the transfer equation for the device and is latched into the outputs as a binary word at the end of the cycle, which is defined by the clock and clock counter.

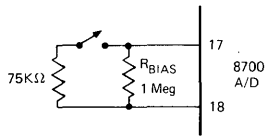
High Accuracy, Linearity, and Stability

All devices in the family are linear to within ½LSB (least significant bit), including differential non-linearity error of less than ½LSB. In the 12-bit 8702 version, this is equivalent to accuracy of ±0.01%. The integrating conversion principle gives inherently monotonic operation, with no missing output codes, as well as excellent thermal stability. Gain temperature coefficient is typically less than 10ppm/°C, and zero drift is typically less than 30μV/°C.

Low Power Consumption

Because of their monolithic CMOS construction, the 8700 Series devices require less than 20mW of power during operation, allowing the designer to use them in remote-located and battery-operated systems where low power consumption is a must.

In applications where only intermittent duty is required from the converter, a simple addition can be made to the bias circuit, for even lower power usage during standby. (See Fig. 3) The R_{BIAS} resistor between pins 17 and 18 sets the current through the integrating amplifier and regulates the conversion rate; recommended value for this resistor in normal operation is 75KΩ. Where standby power reduction is desired, a 1 megohm resistor can be connected instead across these pins, shutting down the op amp and internal clock and reducing quiescent current from 2mA to about 200μA. (Data in the output latches is unaffected, however, and continues to be available for use by the system.) The 75KΩ resistor then is switched in only when a conversion is to be performed; turn-on time is negligible.



REDUCES POWER CONSUMPTION
DURING STANDBY BY 10X
(Approx. 200μA)

FIG. 3 BIAS CONNECTION FOR STANDBY POWER REDUCTION

Conversion Speed

Typical of integrating converters, the 8700 Series offers conversion speeds in the 1 to 20 millisecond range. Specifications are as follows:

Model	Resolution	Conversion Time (typ.)	Conversions/Second (typ.)
8700	8 Bit	1.25 ms	800
8701	10 Bit	5.00 ms	200
8702	12 Bit	20.00 ms	50

Choosing External Components

The relationship between the binary output word (V_{OUT}) and the input voltage is given by the transfer equation

$$V_{OUT} = \frac{V_{IN} \times A \times R_{REF}}{R_{IN} \times V_{REF}}$$

where A is a constant equal to 528 for 8 bits, 2064 for 10 bits and 8208 for 12 bits. Typically, the reference voltage (V_{REF}) and resistor (R_{REF}) would be chosen first to provide a reference current of approximately -20μA. The input resistor then would be selected using the above equation, and the resulting input current will be approximately 10μA at the full scale input voltage.

Recommended values for other external components are as follows:

R_{BIAS} = Bias Resistor = 75KΩ ±10%

C_{INT} = Integrating Capacitor = 68pF ±10%

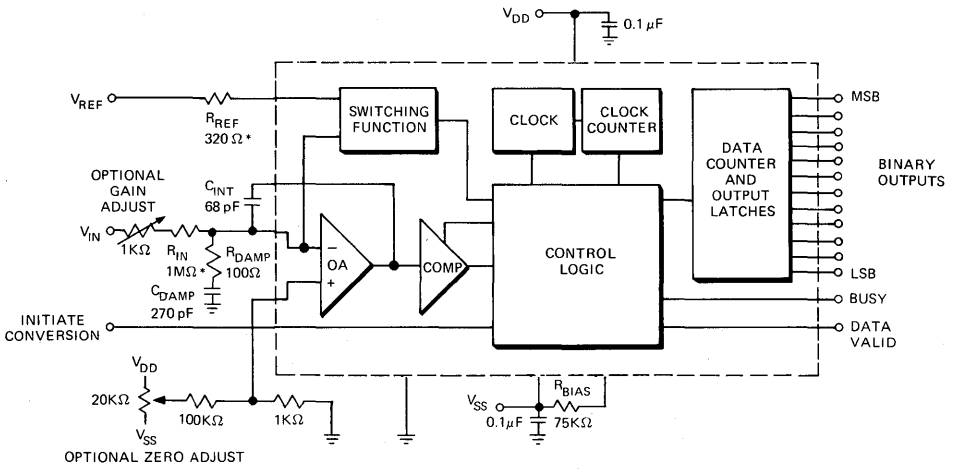
R_{DAMP} = Damping Resistor = 100Ω ±20%

C_{DAMP} = Damping Capacitor = 270pF ±20%

C_{BYPASS} = Bypass Capacitors (2- Optional) ≥0.1μF

Note that the integrating capacitor is shown at 68pF. Earlier data on the 8700 Series shows a value of 33pF for C_{INT}, but recent testing has indicated that 68pF gives improved stability and linearity, especially on the 12-bit device. A glass capacitor is recommended for this application to assure satisfactory dielectric absorption characteristics.

The R_{DAMP} and C_{DAMP} filter circuit is recommended to minimize any spikes generated by the reference current switch. Other time constants can be used, but the response time will be affected.



*COMPONENTS CHOSEN FOR V_{IN} (FULL SCALE) = 10V, V_{REF} = 6.4V

FIG. 4 EXTERNAL COMPONENTS AND ADJUSTMENT CIRCUITS

External Adjustment Circuits

Figure 4 shows a typical selection of components for 10V full scale input and 6.4V reference. Also shown are optional auxiliary circuits for trimming the zero location and full scale gain. Because the digital outputs remain constant outside of the normal operating range (i.e. below zero and above full scale), it is recommended that transition points be used in setting the zero and full scale values. Recommended procedure, using the circuits shown, is as follows:

1. Set the initiate conversion control high to provide free-run operation and verify that converter is operating.

2. Set V_{IN} to +½LSB and trim the zero adjust circuit to obtain a 000...000 to 000...001 transition. This will correctly locate the zero end.

3. For full scale adjustment, set V_{IN} to the full scale value less 1½LSB and trim the gain adjust circuit for a 111...110 to 111...111 transition.

As there is no interaction between these adjustments, they should not have to be repeated.

The R_{BIAS} resistor may also be adjusted to obtain better than the specified temperature stability. However, this adjustment is difficult as it must be performed over temperature to attain optimum results; use of the specified 75KΩ value for R_{BIAS} is recommended unless extremely severe temperature variations are to be encountered.

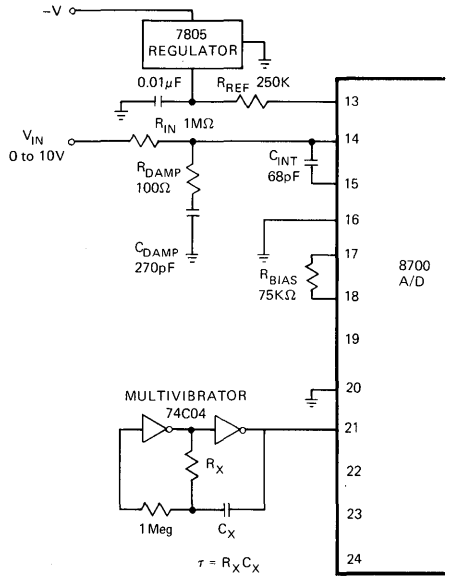


FIG. 5 TYPICAL INPUT CIRCUITRY

Typical Input Circuitry

Another typical hook-up for 8700 Series inputs is shown in Figure 5. For this application, a 5-volt 7805 spot regulator is used

as the reference voltage, establishing R_{REF} at 250KΩ for a -20μA reference current. While virtually any full scale input range can be used, we have shown a range of 0V to +10V and a 1MΩ value for R_{IN}, giving a full scale input current of +10μA. The exact values of the input and reference currents are not critical; more important is their ratio and how it varies with temperature, as any deviation will add to the full scale gain temperature coefficient of the circuit.

The multivibrator circuit shown at pin 21 (initiate conversion control) can be used for system synchronization to initiate the converter at rates slower than the normal free-run mode.

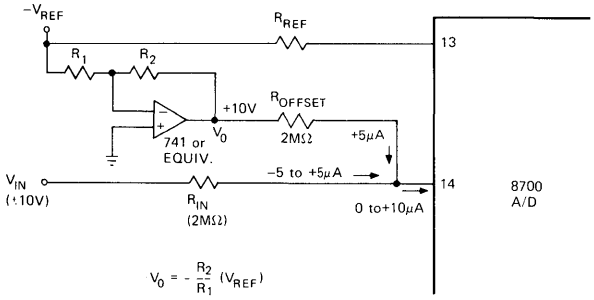


FIG. 6 BIPOLAR OPERATION

Bipolar Operation

In addition to unipolar straight binary operation, the 8700 Series devices can also be connected for bipolar offset binary. (See Fig. 6) An external inverting amplifier, such as a 741 or Teledyne's 844, is all that is required. This offsets the input by half scale; for example, a -10V input will correspond to 0μA and +10V (minus 1LSB) will correspond to +10μA. Two's complement coding can also be generated by inverting the most significant bit (MSB) output.

An Alternate Approach to System Design

Figure 7 shows the conventional approach to a process monitoring and control system. Physical variables are monitored by analog sensors, and the information is transmitted to the computer via an analog multiplexer and A/D converter. The computer then effects feedback control of the process through the D/A converter.

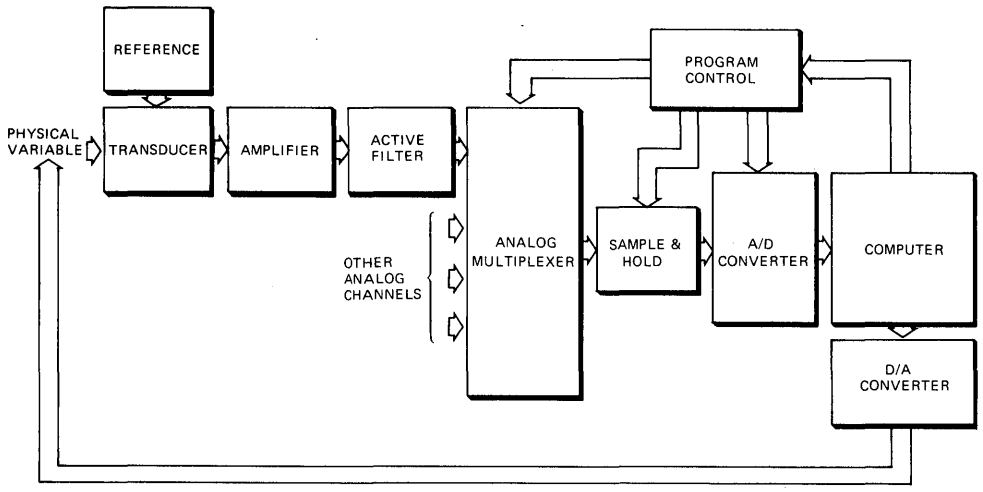


FIG. 7 PROCESS MONITOR/CONTROL SYSTEM
(Conventional Approach)

Note that in this arrangement, the analog input channels typically require an active filter to suppress input noise, that signals are transmitted as analog voltages, and that a sample-and-hold circuit is required to match the multiplexer output to the A/D input. If many analog channels are involved, the A/D must be a high-speed type, requiring some sacrifice in accuracy, cost or both.

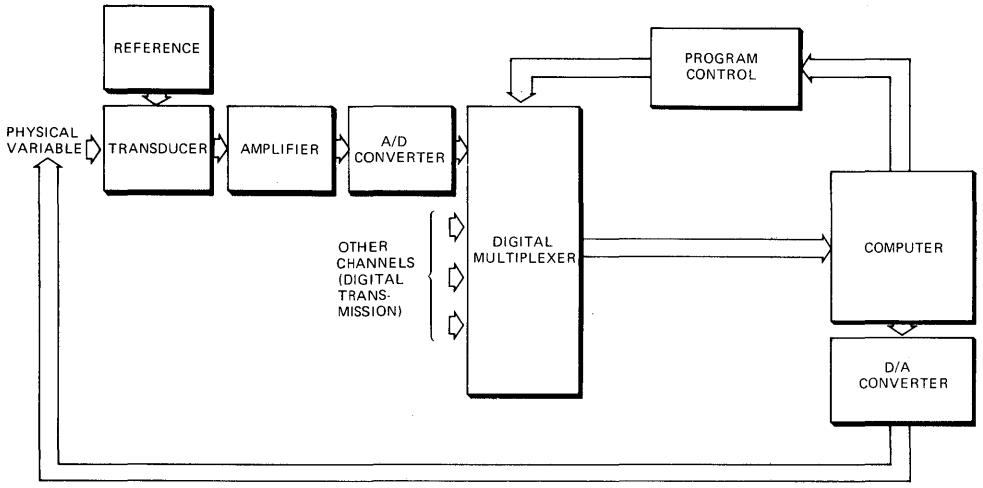


FIG. 8 PROCESS MONITOR/CONTROL SYSTEM
(With Low Cost A/D)

With the availability of high accuracy low cost A/D's such as the Teledyne 8700 Series, the simpler approach shown in Fig. 8 is becoming more attractive to system designers. Because most process variables tend to change fairly slowly, high sampling rates are not needed. The designer can now afford to locate a low-speed integrating converter at each analog sensor, allowing him to transmit data as digital

bits instead of analog voltages, with a resulting improvement in accuracy. The inherently high noise suppression of an integrating device allows him to eliminate the active filters, and the sample-and-hold likewise is unnecessary if the converter includes latched outputs. Use of a digital, rather than analog, multiplexer offers further opportunity for cost saving. The result is a system which will provide better accuracy and performance at a lower overall cost.

18. ANALOG-TO-DIGITAL CONVERSION TECHNIQUES WITH THE M6800 MICROPROCESSOR

DON ALDRIDGE
Applications Engineering
Motorola Semiconductor Products Incorporated
Phoenix, Arizona

The MPU (microprocessing unit) is rapidly replacing both digital and analog circuitry in the industrial control environment. It provides a convenient and efficient method of handling data; controlling valves, motors and relays; and in general, supervising a complete processing machine. However, much of the information required by the MPU for the various computations necessary in the processing system may be available as analog input signals instead of digitally formatted data. These analog signals may be from a pressure transducer, thermistor or other type of sensor. Therefore, for analog data an A/D (analog-to-digital) converter must be added to the MPU system.

Although there are various methods of A/D conversion, each system can usually be divided into two sections - an analog subsystem containing the various analog functions for the A/D and a digital subsystem containing the digital functions. To add an A/D to the MPU, both of the sections may be added externally to the microprocessor in the form of a PC card, hybrid module or monolithic chip. However, only the analog subsystem of the A/D need be added to the microprocessor, since by adding a few instructions to the software, the MPU can perform the function of the digital section of the A/D converter in addition to its other tasks. Therefore, a system design that already contains a MPU and requires analog information needs only one or two additional inexpensive analog components to provide the A/D. The microprocessor software can control the analog section of the A/D, determine the digital value of the analog input from the analog section and perform various calculations with the resulting data. In addition, the MPU can control several analog A/D sections in a timeshare mode, thus multiplexing the analog information at a digital level.

Using the MPU to perform the tasks of the digital section provides a lower cost approach to the A/D function than adding a complete A/D external to the MPU. The information presented in this note describes this technique as applied to both successive approximation (SA) A/D and dual ramp A/D. With the addition of a DAC (digital-to-analog converter), a couple of operational amplifiers, and the appropriate MPU software, an 8- of 10-bit successive approximation A/D is available. Expansion to greater accuracies is possible by modifying the software and adding the appropriate D/A converter. The technique of successive approximation A/D provides medium speed with accuracies compatible with many systems. The second technique adds an MC1405 dual ramp analog sybsystem to the MPU system and, if desired, a digital display to produce a 12-15 bit binary or a 3½-or 4½-digit BCD A/D conversion with 7-segment display readout. This A/D technique has a relatively slow conversion rate but produces a converter of very high accuracy. In addition to

the longer conversion time, the MPU must be totally devoted to the A/D function during the conversion period. However, if maximum speed is not required this technique of A/D allows an inexpensive and practical method of handling analog information.

Figure 1 shows the relative merits of each A/D conversion technique. Listed in this table are conversion time, accuracy and whether interrupts to the MPU are allowed during the conversion cycle.

MPU

The Motorola microprocessor system devices used are the MC6800 MPU, MCM6810 RAM, MCM6830 ROM and MC6820 PIA (peripheral interface adapter). The following is a brief description of the basic MPU system as it pertains to the A/D systems presented later in this application note.

The Motorola MPU system uses a 16-bit address bus and an 8-bit data bus. The 16-bit address bus provides 65,536 possible memory locations which may be either storage devices (RAM, ROM, etc.) or interface devices (PIA, etc.). The basic MPU contains two 8-bit accumulators, one 16-bit index register, a 16-bit counter, a 16-bit stack pointer, and an 8-bit condition code register. The condition code register indicates carry, half carry, interrupt, zero, minus, and 2's complement overflow. Figure 2 shows a functional block of the MC6800 MPU.

The MPU uses 72 instructions with six addressing modes which provide 197 different operations in the MPU. A summary of each instruction and function with the appropriate addressing mode is shown in Appendix A of this note.

The RAMs used in the system are static and contain 128 8-bit words for scratch pad memory while the ROM is mask programmable and contains 1024 8-bit words. The ROM and RAM, along with the remainder of the MPU system components, operate from a single +5 volt power supply; the address bus, data bus and PIAs are TTL compatible.

The MPU system requires a 2ϕ non-overlapping clock with a lower frequency limit of 100 kHz and an upper limit of 1 MHz.

The PIA is the interface device used between the address and data buses and the analog sections of the A/D. Each PIA contains two essentially identical 8-bit interface ports. These ports (A side, B side) each contain three internal registers that include the data register which is the interface from the data bus to the A/D, the data direction register which programs each of the eight lines of the data register as either an input or an output, and the control register which, in addition to other functions, switches the data bus between the data register and the data direction register. Each port to the PIA contains two addition pins, CA1 and CA2, for interrupt capability and extra I/O lines. The functions of these lines are programmable with the remaining bits in the control register. Figure 3 shows a functional block of the MC6820 PIA.

Each PIA requires four address locations in memory. Two addresses access either of the two (A or B sides) data/data direction registers while the remaining two addresses access either of the two control registers. These addresses are decoded by the chip select and register select lines of the PIA which are connected to the MPU address bus. Selection between the data register and data direction register is made by programming a "1" or "0" in the third least significant bit of each control register. A logic "0" accesses the data direction register while a logic "1" accesses the data register.

By programming "0"s in the data direction register each corresponding line performs as an input, while "1"s in the data direction register make corresponding lines act as outputs. The eight lines may be intermixed between inputs and outputs by programming different combinations of "1"s and "0"s into the data direction register. At the beginning of the program the I/O configuration is programmed into the data direction register, after which the control register is programmed to select the data register for I/O operation.

The printouts shown for each A/D program are the source instructions for the cross assembler from the Motorola timeshare. Since the MPU contains a 16-bit address bus and an 8-bit data bus, the hexadecimal number system provides a convenient representation of these numbers. Although the assembler output is in hexadecimal, the source input may be either binary, octal, decimal or hexadecimal. A dollar sign ($) preceding a number in the source instructions indicates hexadecimal, a percent sign (%) indicates binary and an at sign (@) indicates octal. No prefix indicates the decimal number system.

Only the beginning addresses of the program and labels are shown in the source programs. These beginning addresses may be changed prior to assembling the total system program or the programs may be relocated after assembly with little or no modification.

Successive Approximation Techniques

One of the more popular methods of A/D conversion is that of successive approximation. The technique uses a DAC (digital-to-analog converter) in a feedback loop to generate a known analog signal to which the unknown analog input is compared. In addition to medium speed conversion rates, it has the advantages of providing not only a parallel digital output after the conversion is completed but also the serial output during the conversion.

Figure 4 shows the block diagram and waveform of the SA-A/D. The DAC inputs are controlled by the successive approximation register (SAR) which is, as presented here, the microprocessor. The DAC output is compared to the analog input (V_{in}) by the analog comparator and its output controls the SAR. At the start

of a conversion the MSB of the DAC is turned on by the SAR, producing an output from the DAC equal to half of the full scale value. This output is compared to the analog input and if the DAC output is greater than the input unknown, the SAR turns the MSB off. However, if the DAC output is less than the input unknown, the MSB remains on. Following the trial of the MSB the next most significant bit is turned on and again the comparison is made between the DAC output and the input unknown. The same criteria exists as before and this bit is either left on or turned off. This procedure of testing each bit continues for the total number of DAC inputs (bits) in the system.

After the comparison of each bit the digital output is available immediately thus providing both the serial output as well as the parallel output at the end of the conversion. The serial output provides the MSB first, followed by the remaining bits in order. The total conversion time for the SA-A/D is the time required to turn on a bit, compare the DAC output with the input unknown and, if required, turn the bit off, multiplied by the total number of bits in the A/D system. The conversion time is hence constant and unaffected by the analog input value.

One SA-A/D discussed in this paper uses an 8-bit DAC (MC1408) to produce an 8-bit A/D; a second version uses a 10-bit DAC (MC3410) to produce a 10-bit A/D. Both of these are used in conjunction with the MPU as an SAR. In addition, the MC1408 is shown with the MC14549 CMOS SAR as a convert-on-command system under control of the MPU. All of these A/Ds produce a binary output. However, by adding the appropriate software a BCD output or 7-segment-display outputs are available. Also by using a BCD-weighted DAC, the BCD output can be produced directly.

8-Bit SA Program

The flow chart for the 8-bit MPU A/D system is shown in Figure 5; Figures 6 and 7 show the software and the hardware external to the microprocessor. The DAC used is the MC1408L-8, which has active high inputs and a current sink output. An uncompensated MLM301A operational amplifier is used as a comparator while an externally compensated MLM301A or internally compensated MC1741 operational amplifier is used as a buffer amplifier for the input voltage. The output voltage compliance of the DAC is $\pm$0.5 volt; if the current required by the D/A does not match that produced from the output of the buffer amplifier through R1 and R2, then the DAC output will saturate at 0.5 volt above or below ground, thus toggling the comparator. The system is calibrated by adjusting R1 for 1 volt full scale, and zero calibration is set by adjusting R3.

The first MPU instruction for the 8-bit A/D is in line 45 of Figure 6. After assembly, this instruction will be placed in memory location $0A00 as defined in the assembler directive of line 42. The assembled code for this program is relocatable in memory as long as the PIA addresses and storage addresses are unchanged. The program as shown requires 106 memory bytes. Source program lines 45 through 53 configure the PIAs for the proper input/

output configuration. PIA1BD is used for various control functions between the MPU system and the external hardware. The exact configuration of this PIA is shown in lines 28 through 33 of Figure 6. PIA1AD provides the 8-bit output needed for the DAC. Lines 51 through 53 set bit 3 of the PIA control register to access the data register for the actual A/D program.

Lines 55 and 56 set the conversion finished flag, which consists of a LED on the hardware schematic, after which the program enters a loop in lines 63-65 which causes the MPU to wait until the cycle input line goes high. (This feature could be eliminated if the program was a subroutine of a larger control program.) In this case, when a conversion was to be made the control program would go to the A/D subroutine and return with the digital results. Lines 68 and 69 clear the PIA-A which is connected to the DAC inputs and an internal memory location. This memory location is used as a pointer to keep track of which bit of the DAC is currently being tested. Next, the conversion finished line is reset indicating a conversion is in process and the carry bit of the condition code register is set. The memory location POINTR is then rotated right in line 79, moving the carry bit of the condition code register into the MSB of that memory location. Line 80 is a conditional branch that determines if all 8 bits of the DAC have been tested. After nine rotations of POINTR the carry bit will again be set indicating all 8 bits have been compared.

Program lines 81 through 83 load the previous DAC value into an accumulator and the next DAC bit is turned on for the comparator test. An 8 μs delay produced by the NOP instruction of lines 87 through 90 allows the DAC and comparator to settle to a final value before the comparator test of lines 91 and 92. At this point, if the comparator was high, the Yes loop is executed, which generates a simulated clock pulse and a serial output "1". If the comparator was low, lines 95 through 101 are executed, resetting the bit under test and generating a simulated clock pulse and a serial output of "0". The three NOP instructions of the Yes loop equalize the execution time between the high and low comparator loops. After completion of either the high or low comparator loop, the A accumulator which contains the new digital number is stored in PIA1AD and in a RAM memory location labeled ANS. Then the next bit of the DAC is tested in the same manner and this procedure is continued until all eight DAC inputs have been tested. When this has occurred, the program returns to line 55 where the conversion finished flag is "set" and the MPU awaits the next cycle input from PIA1BD.

The total conversion time is 700 μs for the 8-bit converter assuming a 1 MHz MPU clock frequency. The simulated clock pulse is 7 μs wide and can be used to indicate when to sample the serial output.

Characteristic	Successive Approximation			Dual Ramp			
	8-Bit Software	10-Bit Software	8-Bit Hardware	12-Bit Software	3½-Digit Software	4½-Digit Software	3½-Digit Hardware
External Hardware	8-Bit DAC Op Amp Comparator	10-Bit DAC Op Amp Comparator	8-Bit DAC SAR* Op Amp Comparator	MC1405	MC1405	MC1405	MC1405 MC14435 MC14558 (for 7-segment display)
Conversion Rate	700 μs Constant	1.25 ms Constant	60 μs for MPU, plus A/D Conversion Time	165 ms (max) Variable	60 ms (max) Variable	600 ms (max) Variable	183 μs (min) for MPU, plus A/D Conversion Time
Interrupt Capability	Allowed	Allowed	Allowed	Not Allowed	Not Allowed	Not Allowed	Allowed
Number of Memory Locations Required (Including PIA Configuration)	106	145	42	84	296	328	58
Serial Output Available	Yes	Yes	Yes	No	No	No	No

*Successive Approximation Register

FIGURE 1 – Relative Merits of A/D Conversion Techniques

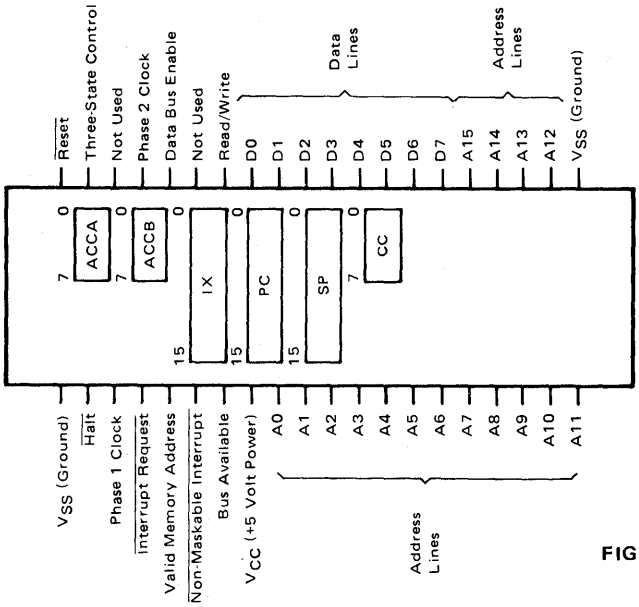


FIGURE 2 – MPU Pin Functions

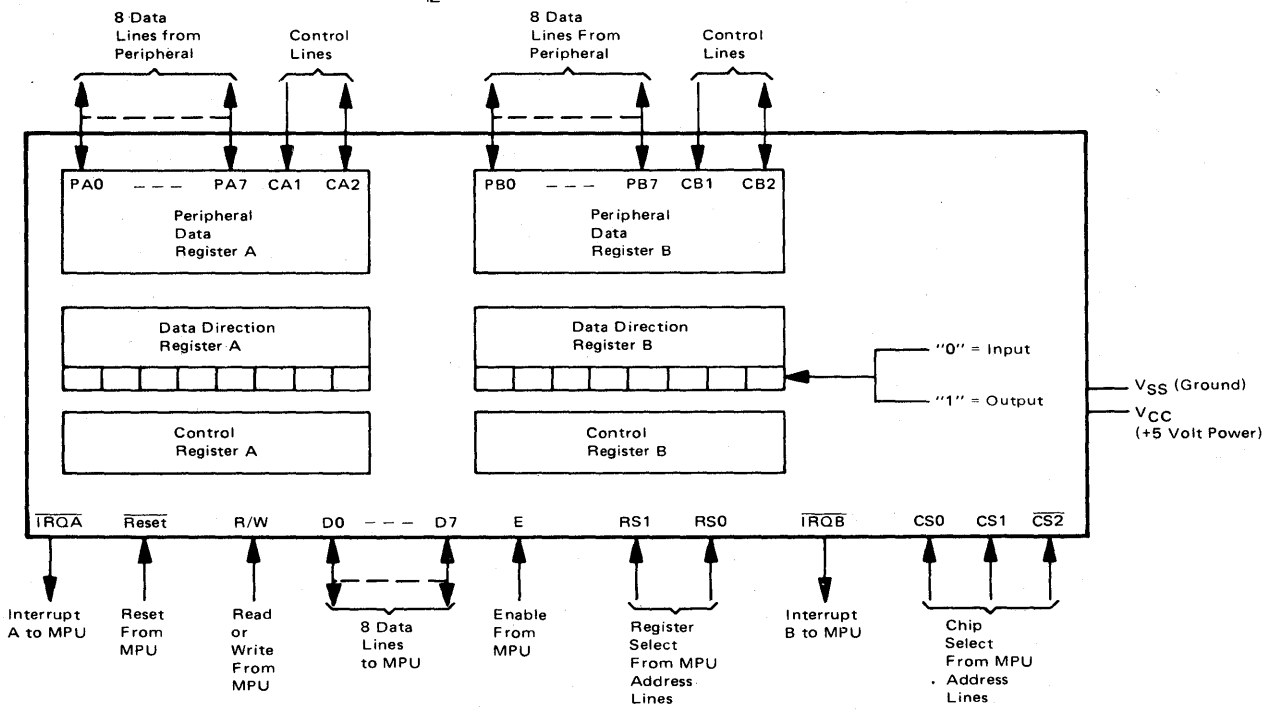
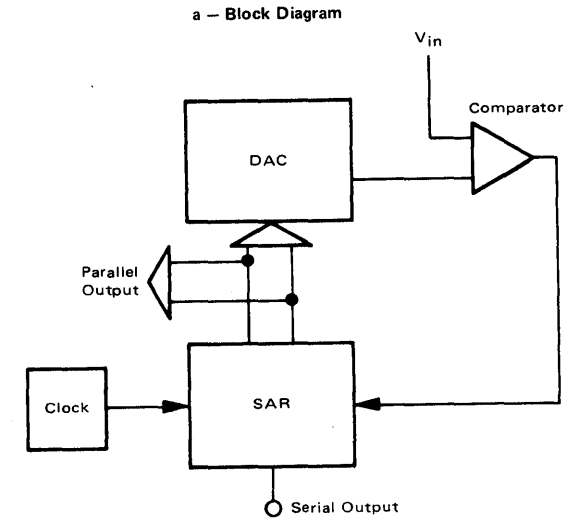


FIGURE 3 – PIA Functions



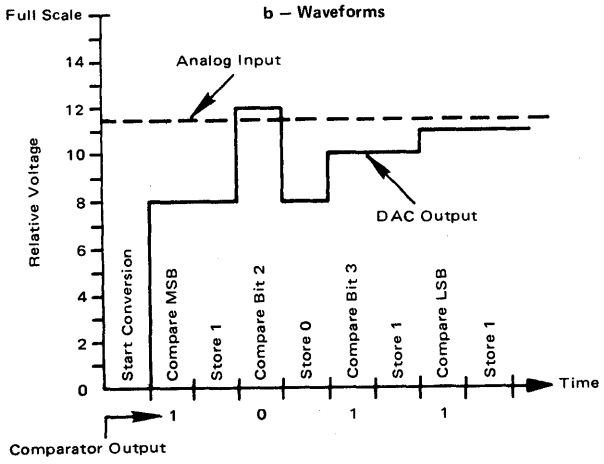


FIGURE 4 – 4-Bit Successive Approximation Converter

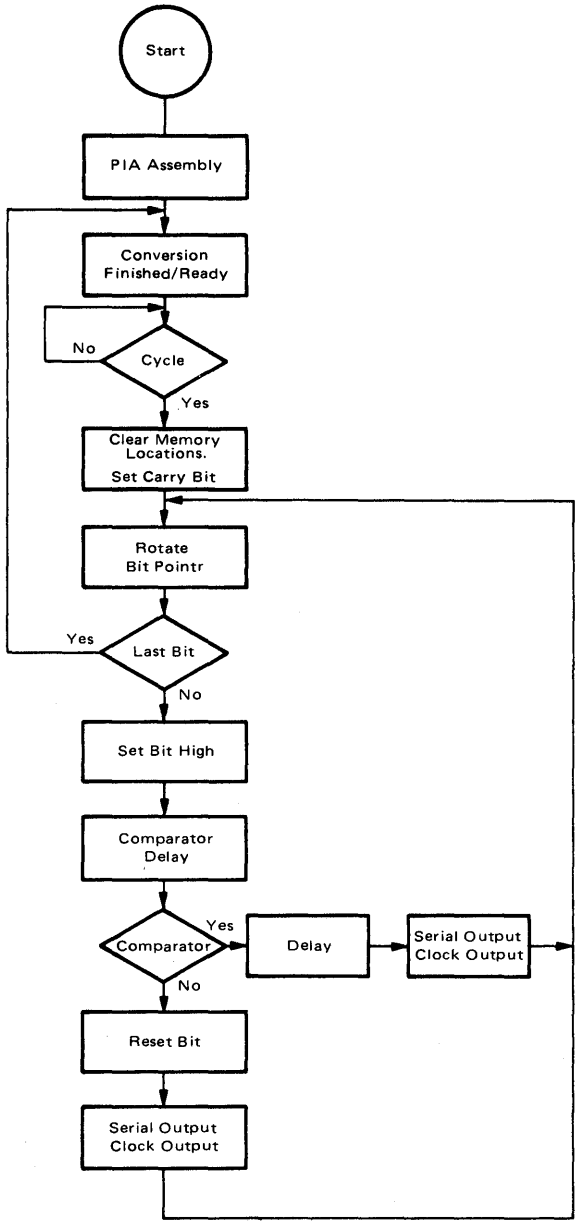


FIGURE 5 – 8-Bit Successive Approximation A/D Flow Diagram

FIGURE 6 – 8-Bit SA Software (Page 1 of 3)

```
 1.000  NAM DWA12
 2.000  OPT MEM
 3.000 *
 4.000 ************************************************************************
 5.000 *                                                                      *
 6.000 *          8 BIT SUCCESSIVE APPROXIMATION A/D                          *
 7.000 *                                                                      *
 8.000 ************************************************************************
 9.000 *
10.000 *
11.000  ORG 0
12.000 ANS RMB 1          FINAL ANSWER MEMORY LOCATION
13.000 POINTR RMB 1       TEMP MEMORY LOCATION
14.000 *
15.000 *
16.000 *
17.000  ORG $4004         PIA MEMORY ADDRESSES
18.000 PIA1AD RMB 1       A SIDE, DATA REGISTER
19.000 PIA1AC RMB 1       A SIDE, CONTROL REGISTER
20.000 PIA1BD RMB 1       B SIDE, DATA REGISTER
21.000 PIA1BC RMB 1       B SIDE, CONTROL REGISTER
22.000 *
23.000 *                  PIA1AD USED FOR DIGITAL OUTPUT TO DAC
24.000 *                  PIA1BD USED FOR A/D CONTROL
25.000 *
26.000 *
27.000 *
28.000 ******************PIA1BD PIN CONNECTIONS******************
29.000 ************************************************************
30.000 * PB7   *  PB6 *  PB5 *  PB4 *  PB3 *  PB2 *  PB1 *  PB0 *
31.000 ************************************************************
32.000 * COMP  *  NC  *  SC  *  CF  *  SD  *  NC  * CYCLE * NC  *
33.000 ************************************************************
34.000 *                           *
35.000 *
```

FIGURE 6 – 8-Bit SA Software (Page 2 of 3)

```
36.000 *
37.000 * COMP-COMPARATOR,SC-SIMULATED CLOCK,SD-SERIAL OUTPUT
38.000 * CF-CONVERSION FINISHED, NC-NO CONNECTION
39.000 *
40.000 *
41.000 *
42.000  ORG $0A00          BEGINNING ADDRESS
43.000 *
44.000 *                                **PIA ASSEMBLY**
45.000  CLR PIA1AC
46.000  CLR PIA1BC
47.000  LDA A #$7C
48.000  STA A PIA1BD
49.000  LDA A #$0FF
50.000  STA A PIA1AD       A SIDE ALL OUTPUTS
51.000  LDA A #$04
52.000  STA A PIA1AC
53.000  STA A PIA1BC
54.000 *
55.000 RSTART LDA A #$10
56.000  STA A PIA1BD       SET CONVERSION FINISHED
57.000 *
58.000 *
59.000 *
60.000 *                                **CYCLE TEST**
61.000 *
62.000 *
63.000 CYCLE LDA A PIA1BD
64.000  AND A #$02
65.000  BEQ CYCLE
66.000 *
67.000 *
68.000  CLR PIA1AD
69.000  CLR POINTR
70.000 *
71.000 *
72.000 *
73.000  CLR PIA1BD         RESET CONVERSION FINISHED
74.000  SEC
75.000 *
76.000 *
77.000 *
78.000 *
79.000 CONVRT ROR POINTR
80.000  BCS RSTART
81.000  LDA A PIA1AD       RECALL PREVIOUS DIGITAL OUTPUT
82.000  ADD A POINTR
83.000  STA A PIA1AD       SET NEW DIGITAL OUTPUT
84.000 *
85.000 *                                **DELAY FOR COMPARATOR**
86.000 *
87.000  NOP
88.000  NOP
89.000  NOP
90.000  NOP
91.000  LDA A PIA1BD       COMPARATOR TEST
92.000  BMI YES
93.000 *
94.000 *                                **LOW COMPARATOR LOOP**
95.000  LDA A PIA1AD
96.000  SUB A POINTR
```

```
 97.000   LDA B #$20          SERIAL OUT OF "0", CLOCK SET
 98.000   STA B PIA1BD
 99.000   CLR B               CLOCK RESET
100.000   STA B PIA1BD
101.000   BRA END
102.000 *
103.000 *                          **HIGH COMPARATOR LOOP**
104.000 YES LDA A PIA1AD
105.000   NOP
106.000   NOP                 DELAY
107.000   NOP
108.000   LDA B #$28          SERIAL OUTPUT OF "1", CLOCK SET
109.000   STA B PIA1BD
110.000   LDA B #$08          CLOCK RESET
111.000   STA B PIA1BD
112.000 *
113.000 END STA A PIA1AD
114.000   STA A ANS
115.000   BRA CONVRT
116.000 *
117.000 *
118.000 *
119.000 *
120.000 *
121.000 *
122.000   MON
```

FIGURE 6 – 8-Bit SA Software (Page 3 of 3)

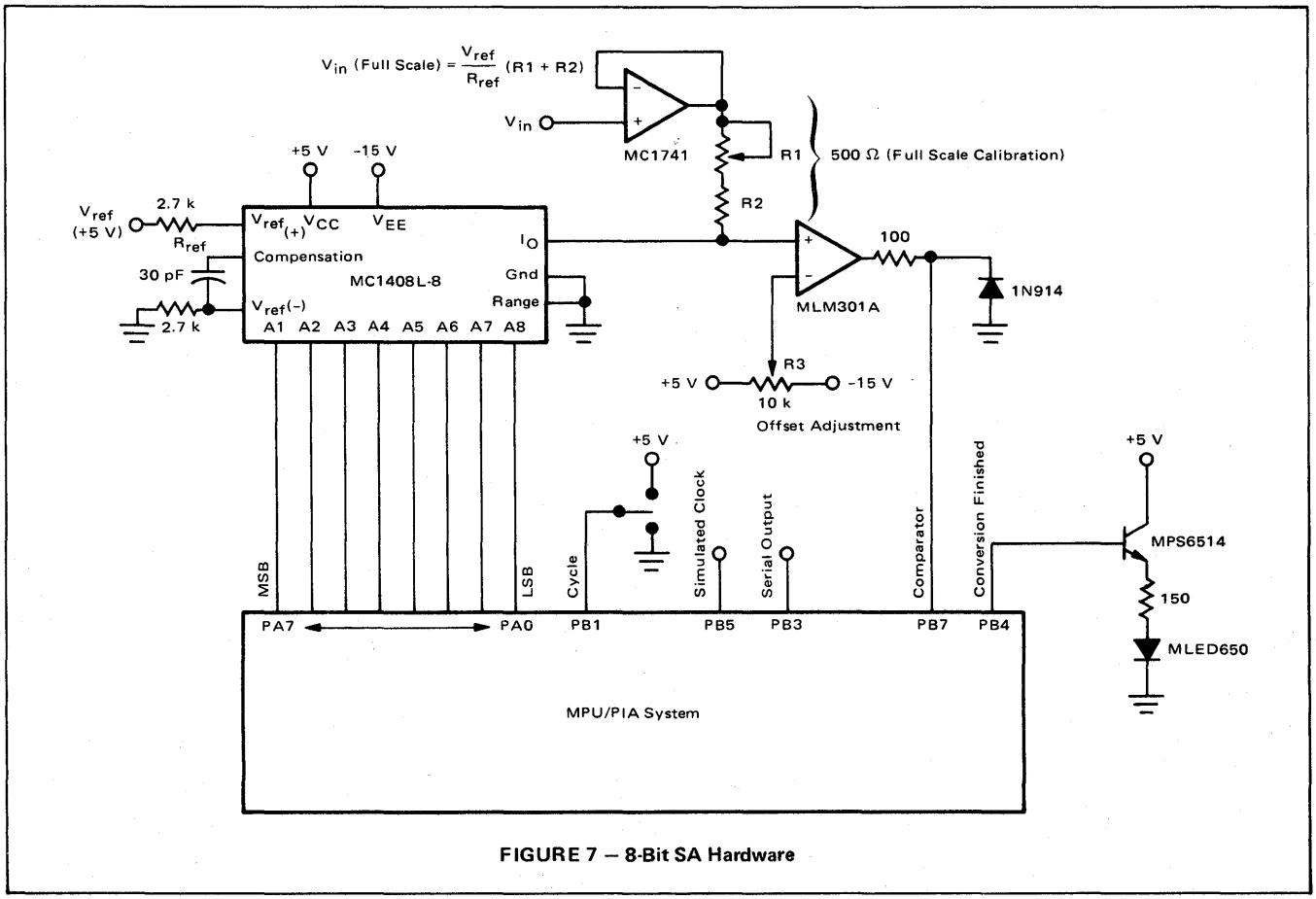
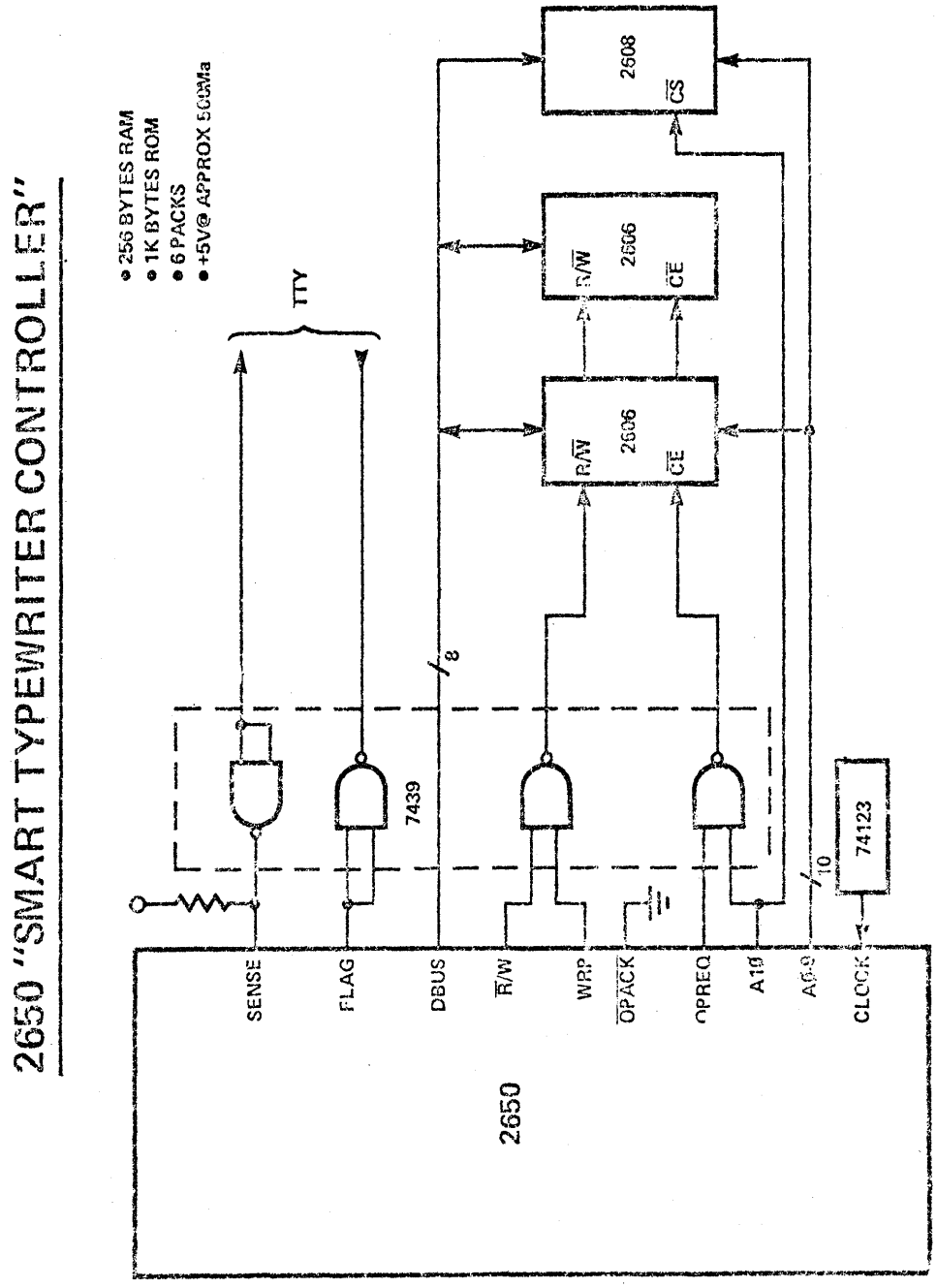


FIGURE 7 – 8-Bit SA Hardware


2650 "SMART TYPEWRITER CONTROLLER"
• 256 BYTES RAM
• 1K BYTES ROM
• 6 PACKS
• +5V@ APPROX 500Ma
TTY
2608
CS
2606
R/W
CE
2606
R/W
CE
7439
8
10
74123
SENSE
FLAG
DBUS
R/W
WRP
OPACK
OPREQ
A10
A09
CLOCK
2650

APPENDIX TWO