intel INTEL CORPORATION, 3065 Bowers Avenue, Santa Clara, California 95051 • (408) 246-7501

MCS-4™
MICRO COMPUTER SET

USERS MANUAL

ANNOUNCING the 4008/4009
Standard Memory and I/O Interface Set
Also includes complete descriptions
of MCS-4 Assemblers
and Simulators

FEBRUARY 1973
REV. 4

MCS-4™

FOUR-BIT PARALLEL MICRO COMPUTER SET

Features

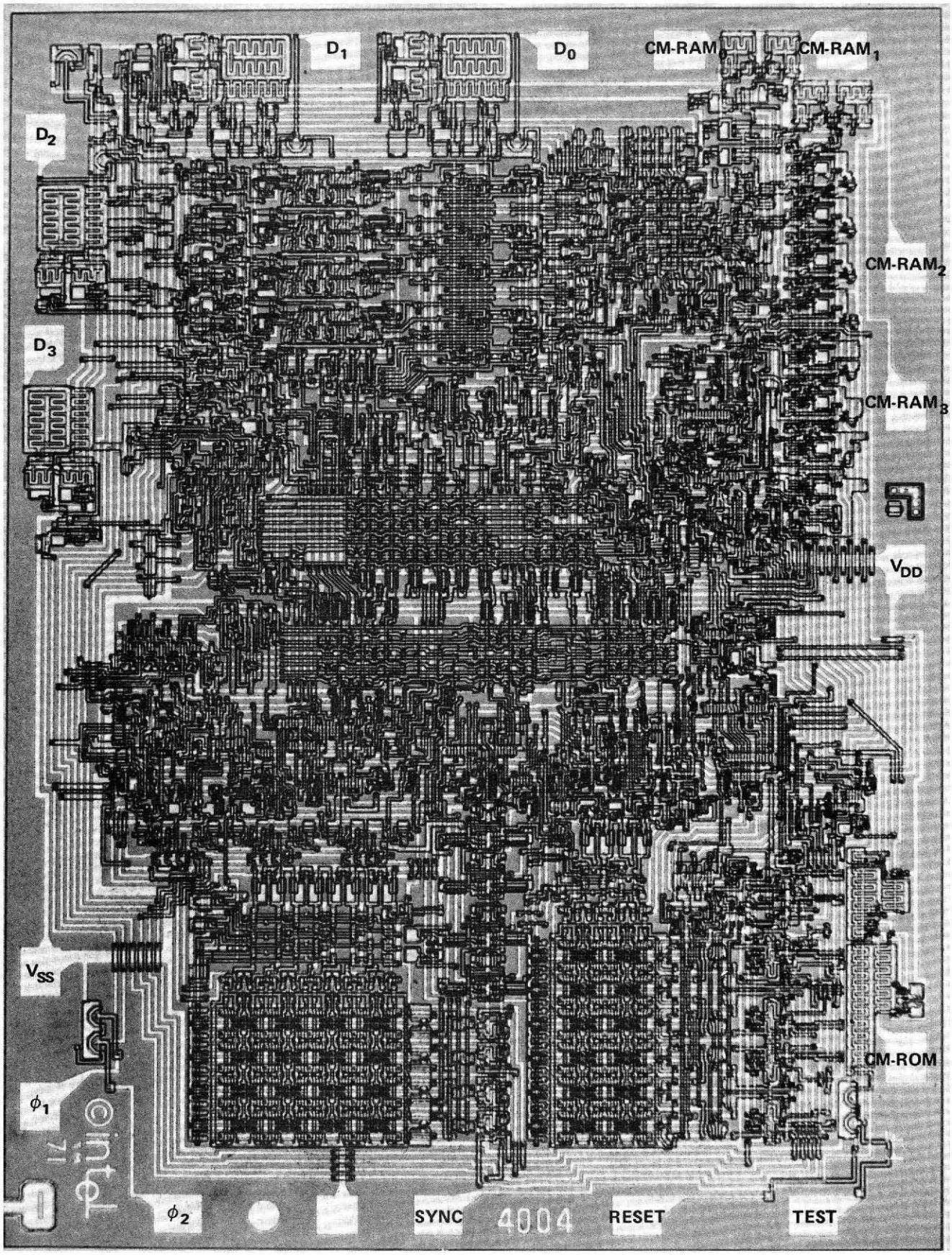
- Microprogrammable General Purpose Computer Set
- 4-Bit Parallel CPU With 45 Instructions
- Instruction Set Includes Conditional Branching, Jump to Subroutine and Indirect Fetching
- Binary and Decimal Arithmetic Modes
- Addition of Two 8-Digit Numbers in 850 Microseconds
- 2-Phase Dynamic Operation
- 10.8 Microsecond Instruction Cycle
- CPU Directly Compatible With MCS-4 ROMs and RAMs
- Easy Expansion — One CPU can Directly Drive up to 32,768 Bits of ROM and up to 5120 Bits of RAM
- Unlimited Number of Output Lines
- Packaged in 16-Pin Dual In-Line Configuration

MCS-4 CAPABILITIES ARE EXPANDED BY THE ADDITION OF THE 4008/4009, THE STANDARD MEMORY AND I/O INTERFACE SET.

Expanded Features

- Directly Compatible With 4004 CPU
- Interface 1702A PROMs Directly to 4004 CPU -- Completely Eliminates TTL Interface
- Permits Program Storage in Alterable Memory
- Execute MCS-4 Programs from any Mix of Standard Intel PROMs, ROMs and RAMs
- Expanded I/O Port Capability
- Each Port May be Both Input and Output -- Up to 16 4-bit Input Ports and 16 4-bit Output Ports
- Number of I/O Ports is Independent of the Size of the Program Memory
- I/O Ports and Control Lines are TTL Compatible
- New Instruction WPM (Write Program Memory) is Used for Loading Alterable Program Storage (RAM)

CONTENTS



4004 Photomicrograph With Pin Designations

I. INTRODUCTION – THE ALTERNATIVE TO RANDOM LOGIC SYSTEMS

A. General Discussion

Since its inception, digital computer applications have evolved from calculation through data processing and into control. The development of the minicomputer has vastly increased the scope of computer usage. In particular, the use of minicomputers in dedicated applications has had a profound effect on systems design.

Many engineers have found having a minicomputer at the heart of a system offers significant advantages. Minicomputer systems are more flexible, can be easily personalized for a particular customer's requirements, and can be more easily changed or updated than fixed-logic design systems. For most designers, the programming of a minicomputer is a much easier and more straightforward procedure than designing a controller with random logic.

Unfortunately, the size and cost of even the smallest minicomputer has limited its use to relatively large and costly systems. This has resulted in many smaller systems being implemented with complicated random logic. INTEL NOW OFFERS ANOTHER ALTERNATIVE. . . THE MCS-4 MICRO COMPUTER SET.

This new concept in LSI technology makes the power of a general purpose computer available to almost every logic designer and represents a strong attack on the dependency of systems manufacturers on complicated random logic systems. This component computer from Intel can provide the same arithmetic, control and computing functions of a minicomputer in as few as two 16 pin DIP's and costs nearly 2 orders of magnitude less.

The set is not designed to compete with the minicomputer, but rather to extend the power of the concept into new ranges of applications. For example, many systems now built of SSI and MSI TTL can now be implemented with a totally self-contained system built around this set of devices.

Heart of each system is a single chip central processor unit (CPU) which performs all control and data processing functions. Auxiliary to the CPU are ROM's which store microprograms and data tables; RAM's which store data and instructions, and Shift Registers which can expand the I/O capacity of the system. The MCS-4 system communicates with circuits and devices outside the family through "ports" provided on each RAM and ROM.

A system using this set of devices will usually consist of one CPU, from one to 16 ROM's, up to 16 RAM's and an arbitrary number of SR's. A minimum system could be designed with just one CPU and one ROM. With these components, you can build distributed computers, dedicated computers, or personalized computers and utilize the almost infinite combinations of microprogramming. The designer buys standard devices, and with microprogramming of the ROM fulfills his own unique circuit requirements.

The three major advantages of Intel microcomputers:

Great system flexibility, with easy program changes, ability to expand or shrink the system, and small size and low power.

Expediency of design, because ROM programming is easier than random circuit design, system checkout is easier using electrically programmable and erasable ROM's, and ability to insert new microprograms helps prevent system obsolescence.

Manufacturing economies come from simple DIP package design, automatic insertion, lower labor costs, lower inventory of parts and boards.

When designing with random logic (logic gates, flip flops, etc.), the designer will usually start with a description of the desired function and attempt to wire counters, gates, etc. to achieve this function. Switches, displays, etc. are also connected to the logic. To correct errors or make changes in a design usually requires significant changes in wiring, often requiring that circuit boards be scrapped and replaced by new ones.

To do the same design with the MCS-4 Micro Computer Set, the designer again starts with the functional description. However, he implements these functions by encoding suitable sequences of instructions in ROM. The MCS-4 instruction set is quite complete and allows a wide variety of functions to be performed: decimal or binary arithmetic, counting, decisions, table-lookup, etc. Switches, displays, etc. are connected to the system via the input and output ports.

As a result of this organization, almost the entire logic, the entire "personality" of the machine is determined by the instructions in ROM. Very significant modifications of machine characteristics can be made by changing or adding ROM's without making any changes in wiring or circuit boards.

Thus the set offers tremendous flexibility of design and allows the user to have many of the desirable features of a custom MOS LSI design--small package count, a set of components which is uniquely his own (for each user's program routines are his proprietary property)--and yet have none of the disadvantages of long development cycle, high development costs, etc. The short design cycle and flexibility associated with ROM programming allows much more rapid response to market demands than is possible with custom LSI and thus provides insurance against obsolescence.

B. Applications for the MCS-4 Micro Computer Set

Heart of the MCS-4 micro computer set is the 4004 CPU. This device has a powerful and versatile instruction set which allows the system to perform a wide variety of arithmetic, control and decision functions. The microprograms stored in the ROM devices give the designer the power of designing custom computers with standard components. You can

use the MCS-4 almost anywhere. Here are a few examples:

Control Functions - Because of low initial cost and flexibility of programming, the MCS-4 can be used in place of random logic in systems such as those in process control, numeric controls, elevator controls, highway and rail traffic controls. By changing ROM microprograms the whole system can easily be modified and updated.

Computer Peripherals - The system can be conveniently used in peripheral equipment to control displays, keyboards, printers, readers, plotters and to give intelligence to terminals.

Computing Systems - The MCS-4 system is ideally suited for such devices as billing machines, cash registers, point of sale terminals and accounting machines. For example, the adding of two 8-digit numbers can be done in 850 microseconds. In addition, the MCS-4 can be efficiently used to decentralize central computer functions.

Other Applications - The elements of the MCS-4 have many applications within transportation, automotive, medical electronics and test systems, where inexpensive dedicated computers can improve system performance.

C. Features of the MCS-4

- 4-bit parallel CPU with 45 instructions
- Decimal and binary arithmetic modes
- 10.8 μs instruction cycle
- Addition of Two 8-digit numbers in 850 μsec.
- Sixteen 4-bit general purpose registers
- Nesting of subroutines up to 3 levels
- Instruction Set includes conditional branching, jump to subroutine, and indirect fetching
- 2-phase dynamic operation
- Synchronous operation with memories
- Direct compatibility with 4001, 4002 and 4003
- No interface circuitry to memory and I/O required
- Directly drives up to: 4K by 8 ROM (16 4001's)
  1280 by 4 RAM (16 4002's)
  128 I/O lines (without 4003)
  Unlimited I/O (with 4003's)
- Memory capacity expandable through bank switching
- 16-pin DIP package
- P-channel Silicon Gate MOS
- Minimum system: CPU and one ROM

II. MCS-4 SYSTEM DESCRIPTION

A. General Description

Each MCS-4 circuit constitues a basic standard building block which allows the design of many different types of systems which can be fabricated using the same parts. The only custom part is the ROM chip which will store a microprogram defined by the user and requires a metal mask option for each new program.

The MCS-4 micro computer set consists of the following 4 chips, each packaged in a 16 pin DIP package:

(1) A Central Processor Unit Chip -CPU - 4004
(2) A Read Only Memory Chip - ROM - 4001
(3) A Random Access Memory Chip - RAM - 4002
(4) A Shift Register Chip - SR - 4003

The CPU contains the control unit and the arithmetic unit of a general purpose microprogrammable computer. The ROM stores microprograms and data tables, the RAM stores data and instructions, and the Shift Register is used in conjunction with I/O devices to effectively increase the number of I/O lines.

The MCS-4 set has been designed for optimum interfaceability; the CPU communicates with the RAM's and ROM's by means of a 4-line data bus (D_0, D_1, D_2, D_3). This single data bus is used for all information flow between the chips except for control signals which are sent to RAM and ROM over 5 additional lines. One CPU controls up to 16 ROM's (4K x 8 words), 16 RAM's (1280 x 4 words), and 128 I/O lines without requiring any interface circuit. With the addition of few gates up to 48 ROMS & RAMS combined and 192 I/O lines can be controlled by one CPU.

The I/O function, although different from the ROM and RAM functions, is physically located in the ROM and RAM chips. Each 4001 and 4002 has 4 I/O lines for communication with I/O devices.

4001-ROM - The 4001 is a 2048 Bit metal mask programmable ROM providing custom microprogramming capability for the MCS-4 micro computer set. Each chip is organized as 256 x 8 bit words which can be used for storing programs or data tables. Each chip also has a 4 bit input-output (I/O) port which is used to route information to and from the data bus lines in and out of the system.

4002-RAM - The 4002 performs two functions. As a RAM it stores 320 bits arranged as 4 registers of twenty 4-bit characters each. As a vehicle of communication with peripheral devices, it is provided with 4 output lines and associated control logic to perform output operations.

4003-SR - The 4003 is a 10 bit Serial-in/parallel-out, serial-out shift register. Its function is to increase the number of output lines to interface with I/O devices such as keyboards, displays, printers, teletypewriters, switches, readers, A-D converters, etc.

4004-CPU - The 4004 is a central processor unit designed to work in conjunction with the other members of the MCS-4 micro computer set to form a completely self-contained system. The CPU communicates with the other members of the set through a four line data bus and with the peripheral devices through the RAM, ROM or SR I/O ports. The CPU chip contains 5 command control lines, four of which are used to control the RAM chips (each line can control up to 4 RAM chips for a total system capacity of 16 RAM's) and one which is used to control a bank of up to 16 ROM's.

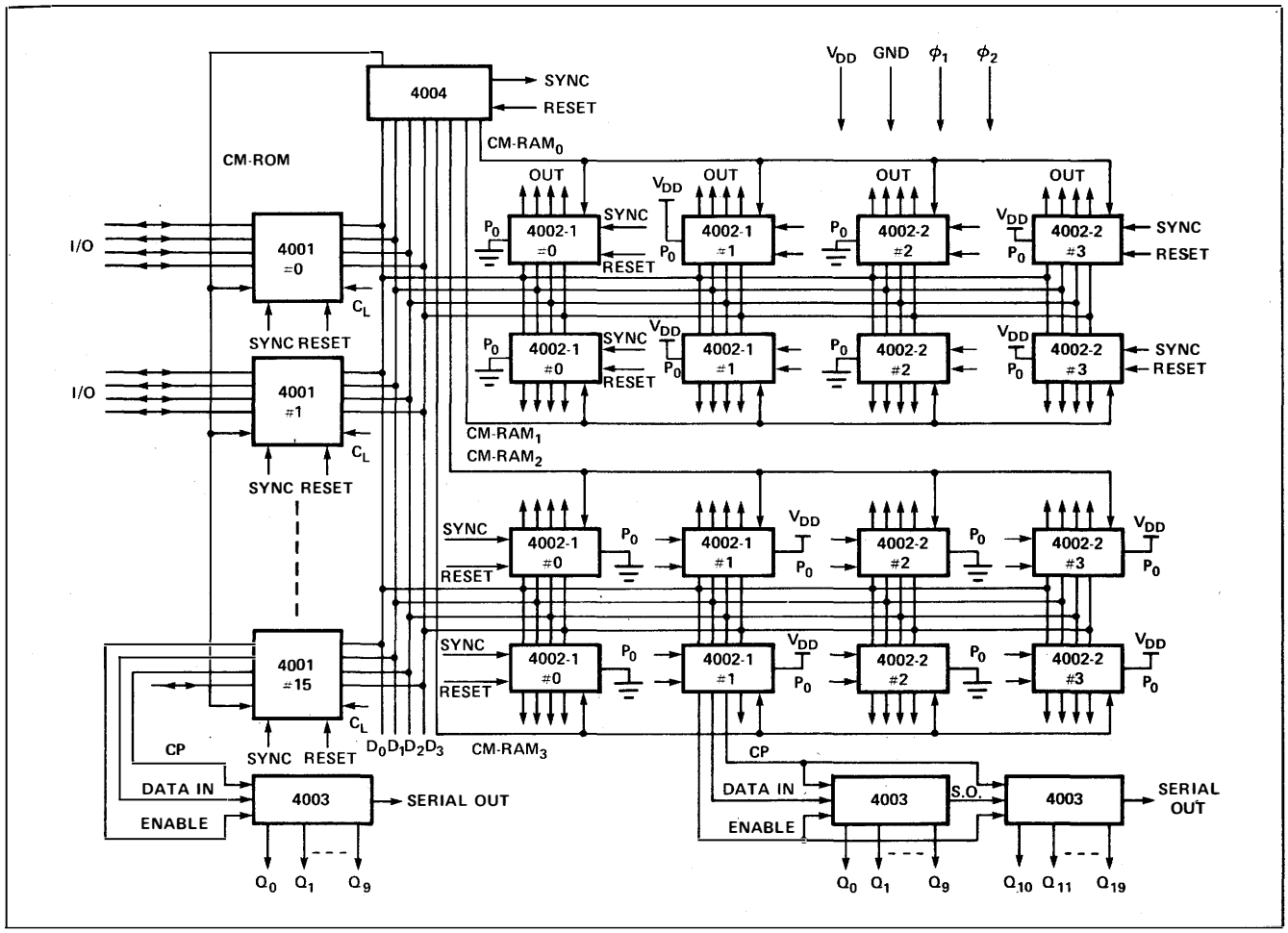


Figure 1. MCS-4 System Interconnection

B. Basic System Operation

The MCS-4 uses a 10.8 μsec instruction cycle. The CPU (4004) generates a synchronizing signal(SYNC), indicating the start of an instruction cycle, and sends it to the ROM's (4001) and RAM's (4002).

Basic instruction execution requires 8 or 16 cycles of a 750 kHz clock. In a typical sequence, the CPU sends 12 bits of address (in three 4 bit bytes on the data bus) to the ROM's in the first three cycles (A_1, A_2, A_3). This address selects 1 out of 16 chips and 1 out of 256 8-bit words in that chip. The selected ROM chip sends back 8 bits of instruction (OPR, OPA) to the CPU in the next two cycles (M_1, M_2). This instruction is sent over the 4 line data bus in two 4 bit bytes. The instruction is then interpreted and executed in the final three cycles (X_1, X_2, X_3). (See Figure 2)

When an I/O instruction is received from the ROM, data is transferred to or from the CPU accumulator on the four ROM I/O lines during X_2 time.

A set of four RAM's is controlled by one of four command control lines from the CPU. The address of a RAM chip, register and character is stored in two index registers in the CPU and is transferred to the RAM during X_2, X_3 time when a RAM instruction is executed. When the RAM output instruction is received by the CPU, the content of the CPU accumulator is transferred to the four RAM output lines.

The CPU, RAM's and ROM's can be controlled by an external RESET line. While RESET is activated the contents of the registers and flip-flops are cleared. After RESET, the CPU will start from address 0 and CM-RAM_0 is selected.

The interconnection of the MCS-4 system is shown in Figure 1. An expanded configureation is shown. The minimum system configuration consists of one CPU (4004) and one ROM (4001).

C. MCS-4 Logic Definitions

The MCS-4 devices operate with negative Logic. Logic "1" is defined as the low voltage (negative voltage) Level and Logic "0" is defined as the high voltage Level (V_{SS}). This definition will be used throughout the manual.

D. Basic System Timing

For the correct operation of the system two non-overlapping clock phases - $\emptyset_1$, $\emptyset_2$ - must be externally supplied to the 4001, 4002 and 4004.(1) The 4004 will generate a SYNC signal every 8 clock periods and will send it to the 4001's and 4002's. The SYNC signal marks the beginning of each instruction cycle. The 4001's and 4002's will then generate internal timing using SYNC and $\emptyset_1$, $\emptyset_2$.

(1) The 4003 is a static shift register and does not use these two clocks for its operation.

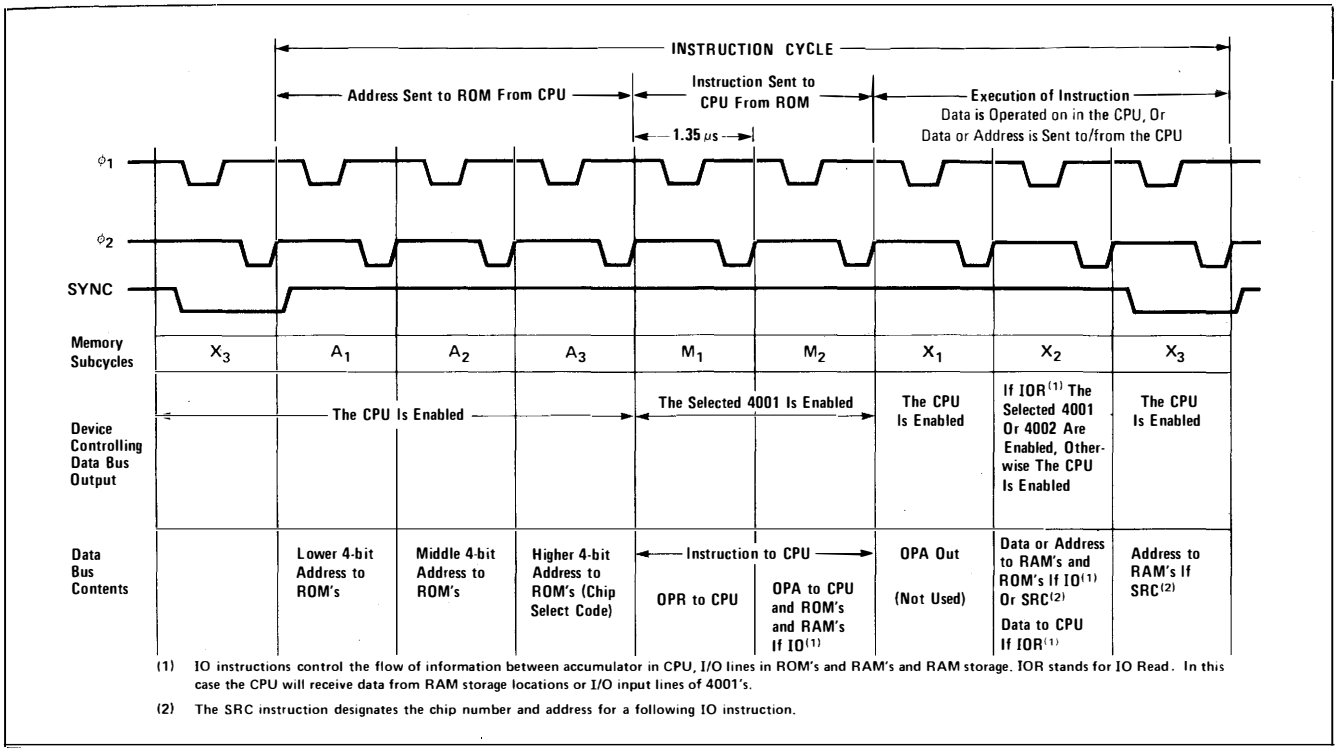


Figure 2. MCS-4 Basic Instruction Cycle

Figure 2 shows how a basic instruction cycle is subdivided and what the activity is on the data bus during each clock period. Each data bus output buffer has three possible states: "1", "0" and floating. At a given time, only 1 output buffer is allowed to drive a data line, therefore all the other buffers must be in a floating condition. However, more than 1 input buffer per data line can receive data at the same time.

III 4 BIT CENTRAL PROCESSOR UNIT (CPU) – 4004

A. Description

The 4004 block diagram shown in Figure 3 contains the following functional blocks:

(1) Address register (program counter and stack organizaed as 4 words of 12 bits each) and address incrementer.
(2) Index register (64 bits organized as 16 words of 4 bits each.
(3) 4-bit adder.
(4) Instruction register (8 bits wide), decoder and control.
(5) Peripheral circuitry.

The functional blocks communicate internally through a 4-line bus and are shown in Figure 3. The function and composition of each block is as follows:

1. Address Register (Program counter & Stack) & Address Incrementer

The address register is a dynamic RAM cell array of 4 x 12 bits. It contains one level used to store the instruction address (program counter) and 3 levels used as a stack for subroutine calls. The stack address is provided by the effective address counter and by the refresh counter, and it is multiplexed to the decoder.

The address when read is stored in an address buffer and is demultiplexed to the internal bus during A_1, A_2, and A_3 in three 4-bit slices (see Figure 2 for basic instruction cycle). The address is incremented by a 4-bit carry look-ahead circuit (address incrementer) after each 4-bit slice is sent out on the data bus. The incremented address is transferred back to the address buffer and finally written back into the address register.

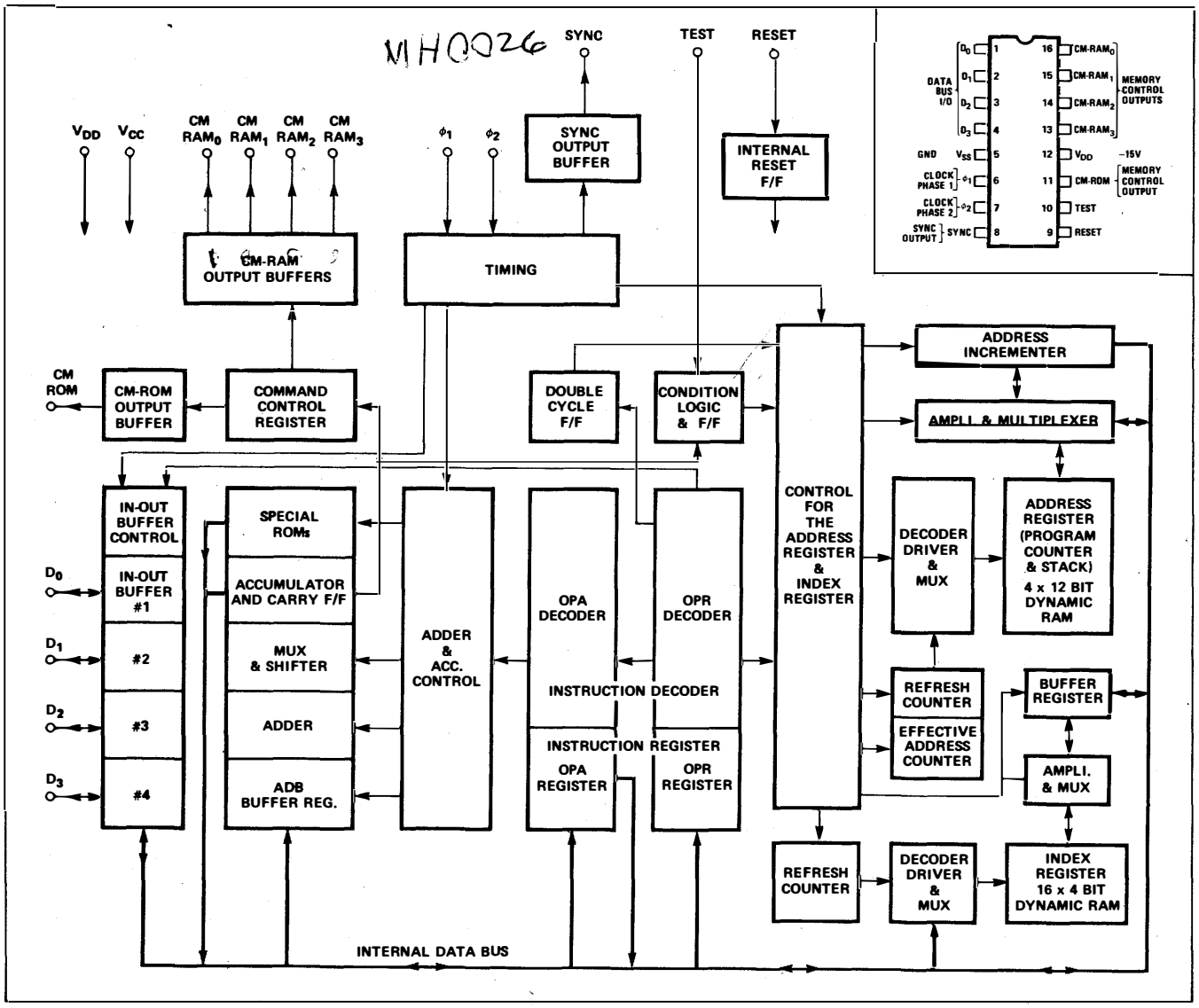


Figure 3. 4004 CPU Block Diagram

2. Index Register

The index register is a dynamic RAM cell array of 16 x 4 bits and has two modes of operation. In one mode of operation the index register provides 16 directly addressable storage locations for intermediate computation and control. In the second mode, the index register provides 8 pairs of addressable storage locations for addressing RAM and ROM as well as for storing data fetched from ROM.

The index register address is provided by the internal bus and by the refresh counter and is multiplexed to the index register decoder.

The content of the index register is transferred to the internal bus through a multiplexer. Writing into the register is accomplished by transferring the content of the internal bus into a temporary register and then to the index register.

3. 4-Bit Adder

The 4-bit adder is of the ripple-through carry type. One term of the addition comes from the "ADB" register which communicates with the internal bus on one side and can transfer data or $\overline{data}$ to the adder. The other term of the addition comes from the accumulator and carry flip-flop. Both data and $\overline{data}$ can be transferred. The output of the adder is transferred to the accumulator and carry FF. The accumulator is provided with a shifter to implement rotate right and rotate left instructions. The accumulator also communicates with the command control register, special ROM's, the condition flip-flop and the internal bus. The command control register holds a 3-bit code used for CM-RAM line switching. The special ROM's perform a code conversion for DAA (decimal adjust accumulator) and KBP (Keyboard Process) instructions. The special ROM's also communicate with the internal bus. The condition logic senses ADD = 0 and ACC = 0 conditions, the state of the carry FF, and the state of an external signal (TEST) to implement JCN (jump on condition) and ISZ (increment index register skip if zero) instructions.

4. Instruction Register Decoder and Control

The instruction register (consisting of the OPR Register and OPA Register each 4 bits wide) is loaded with the contents of the internal bus (at M_1 and M_2 time in the instruction cycle) through a multiplexer and holds the instruction fetched from ROM. The instructions are decoded in the instruction decoder and appropriately gated with timing signals to provide the control signals for the various functional blocks. A double cycle FF is set from any one of 5 double-length instructions. Double-length instructions are instructions whose OP-code is 16 bits wide (instead of 8 bits)and that require two system cycles (16 clock cycles) for their execution. Double length instructions are stored in two successive locations in ROM. A condition FF controls JCN and ISZ instructions and is set by the condition logic. The state of an external pin "test" can control one of the conditions in the JCN instruction.

5. Peripheral Circuitry

This includes:

a. The data bus input-output buffers communicating between data pads and internal bus.

b. Timing and SYNC generator.

c. 1 ROM command control (CM-ROM) and the 4 RAM command control ($CM\text{-}RAM_i$) output buffers.

d. Reset flip-flop.

During reset (Reset pin low), all RAM's and static FF's are cleared, and the data bus is set to "0". After reset, program control will start from "0" step and $CM\text{-}RAM_0$ is selected. To completely clear all registers and RAM locations in the CPU the reset signal must be applied for at least 8 full instruction cycles (64 clock cycles) to allow the index register refresh counter to scan all locations in memory. (256 clock cycles for the 4002 RAM).

6. Instruction Repertoire

The instruction repertoire of the 4004 consists of:

a. 16 machine instructions (5 of which are double length)

b. 14 accumulator group instructions

c. 15 input/output and RAM instructions

The instruction set and its format will be briefly described in the next section. Section VII will then describe each instruction in detail.

B. CPU Instruction Set Format, Index Register Organization, and Operation of the Address Register and Command Lines

1. Instruction Set Format

a. Machine Instructions

- 1-word instructions - 8 bits wide and requiring 8 clock periods (1 instruction cycle)
- 2-word instructions -16 bits wide and requiring 16 clock periods (2 instruction cycles) for execution

A 1-word instruction occupies one location in ROM (each location can hold one 8-bit word) and a 2-word instruction occupies two successive locations in ROM. Each instruction word is divided into two 4-bit fields. The upper 4 bits is called the OPR and contains the operation code. The lower 4 bits is called the OPA and contains the modifier. For a single word machine instruction the operation code (OPR) contains the code of the operation that is to be performed (add, subtract, load, etc.). The modifier (OPA) contains one of 4 things:

(1) A register address
(2) A register pair address
(3) 4 bits of data
(4) An instruction modifier

For a 2-word machine instruction the 1st word is similar to a 1-word instruction, however, the modifier (OPA) contains one of 4 things:

(1) A register address
(2) A register pair address
(3) The upper portion of another ROM address
(4) A condition for jumping

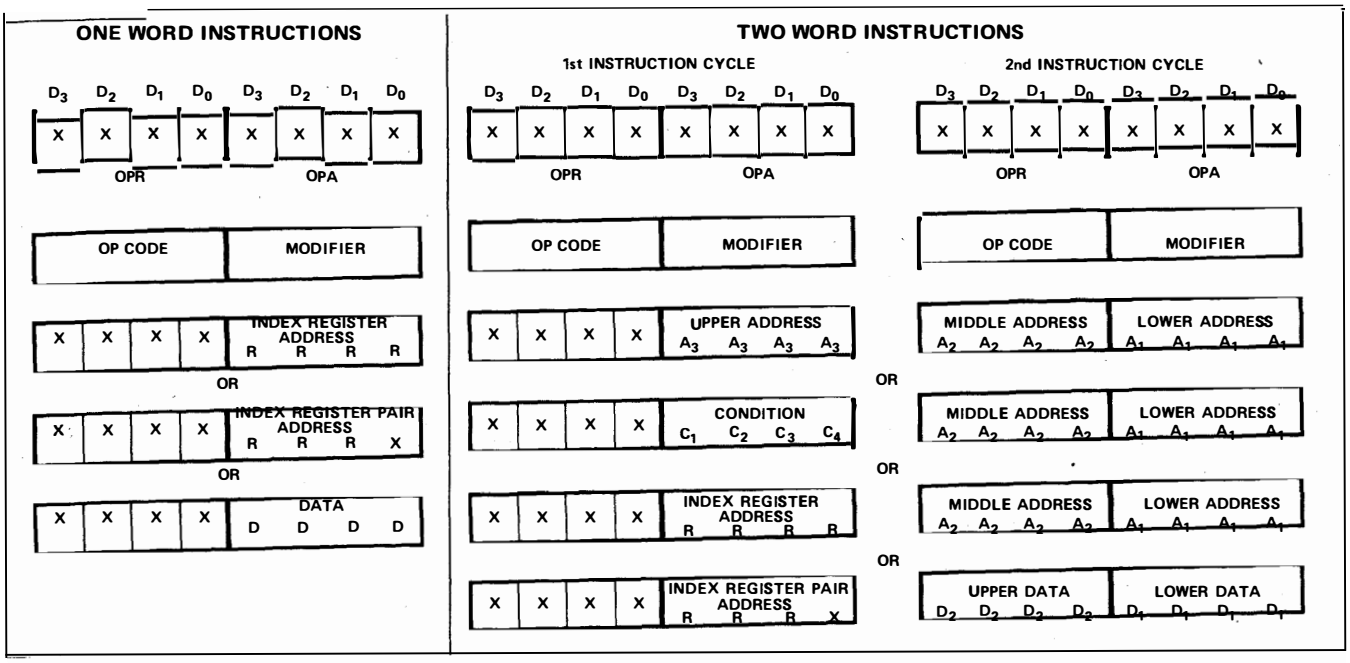


Table I - Machine Instruction Format

The 2nd word contains either the middle portion (in OPR) and lower portion (in OPA) of another ROM address or 8 bits of data (the upper 4 bits in OPR and the lower 4 bits in OPA).

The upper 4 bits of instruction (OPR) will always be fetched before the lower 4 bits of instruction (OPA) during M_1 and M_2 times respectively. Table I illustrates the contents of each 4-bit field in the machine instructions.

b. Input/Output & RAM Instructions and Accumulator Group Instructions

In these instructions (which are all single word) the OPR contains a 4-bit code which identifies either the I/O instruction or the accumulator group instruction and the OPA contains a 4-bit code which identifies the operation to be performed. Table II illustrates the contents of each 4-bit field.

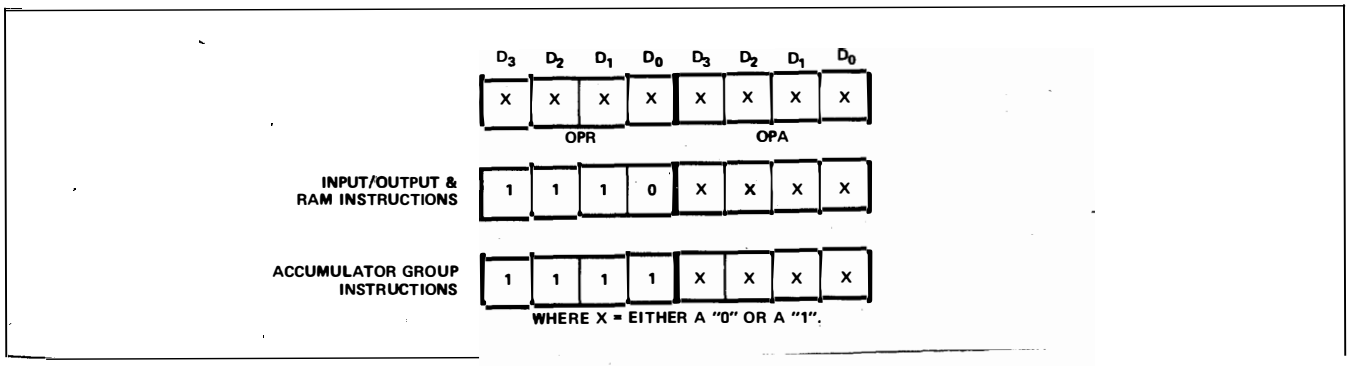


Table II - I/O and Accumulator Group Instruction Formats

2. Index Register Organization

The index register can be addressed in two modes

a. By specifying 1 out of 16 possible locations with an OPA code of the form RRRR(1) (See Table III).

b. By specifying 1 out of 8 pairs with an OPA code of the form RRRX(2) (See Table III).

When the index register is used as a pair register, the even number register (RRR0) is used as the location of the middle address or the upper data fetched from the ROM, the odd number register (RRR1) is used as the location of the lower address or the lower data fetched from the ROM.

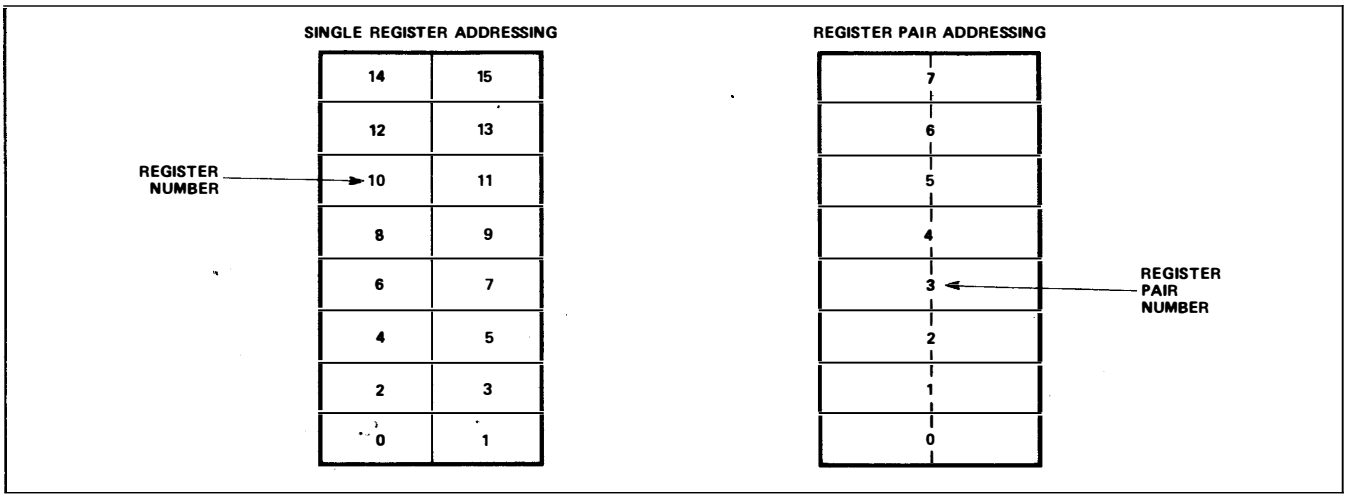


Table III - Index Register Organization

3. Operation of the Address Register (Program Counter and Stack)

The address register contains four 12-bit registers; one register is used as the program counter and stores the instruction address. the other 3 registers make up the push down stack.

Initially any one of the 4 registers can be used as the program counter to store the instruction address. In a typical sequence the program counter is incremented by 1 after the last address is sent out. This new address then becomes the effective address. If a JMS (Jump to Subroutine) instruction is received by the CPU, the program control is transferred to the address called out in JMS instruction. This address is stored in the register just above the old program counter which now saves the address of the next instruction to be executed following the last JMS.(3) This return address becomes the effective address following the BBL(Branch back and load) instruction at the end of the subroutine.

(1) In this case the instruction is executed on the 4-bit content addressed by RRRR.

(2) In this case the instruction is executed on the 8-bit content addressed by RRRX, where X is specified for each instruction.

(3) Since the JMS instruction is a 2-word instruction the old effective address is incremented by 2 to correctly give the address of the next instruction to be executed after the return from JMS.

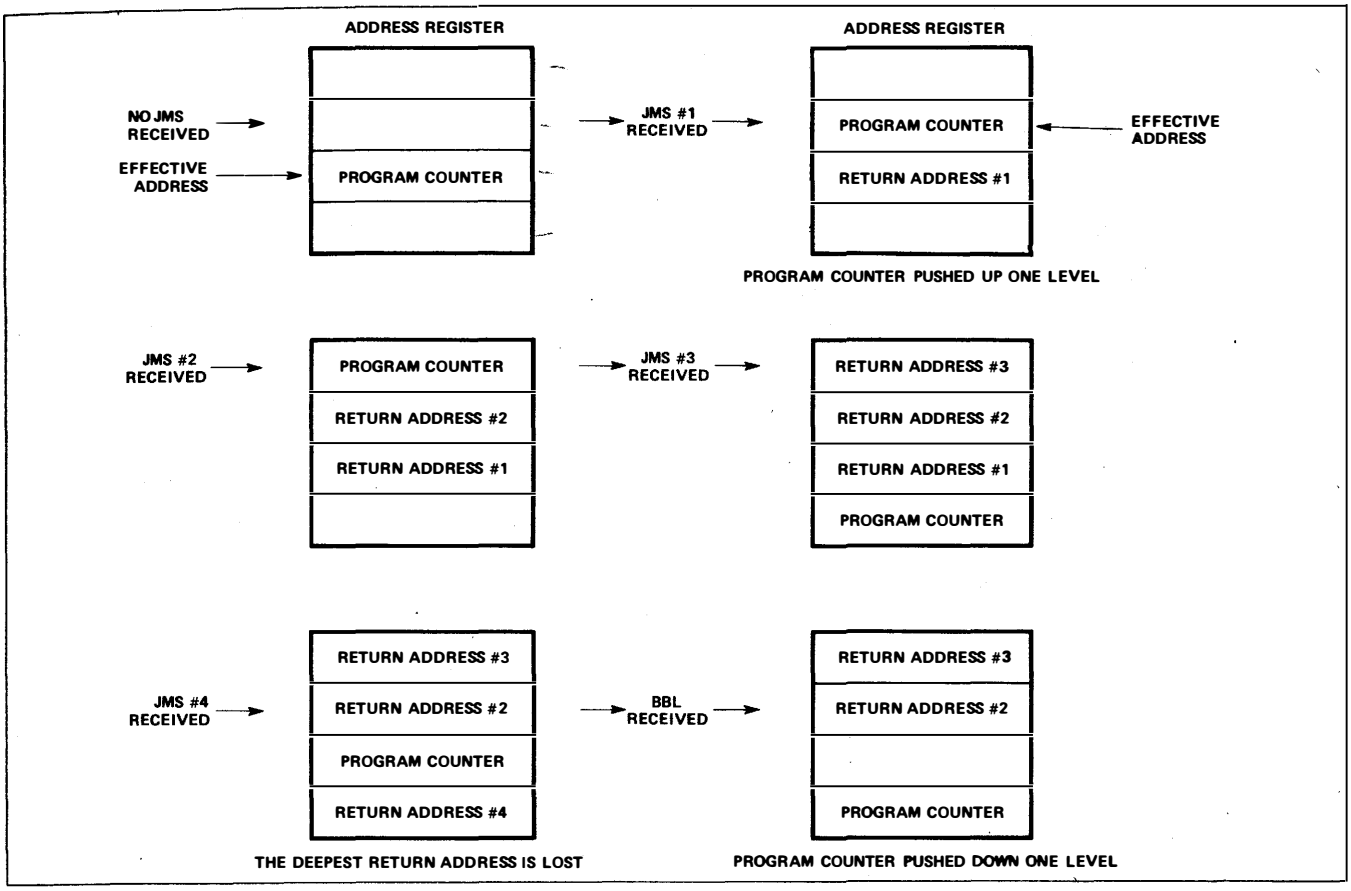


Table IV - Operation of the Address Register on a Jump to Subroutine Instruction

In summary, then, a JMS instruction pushes the program counter up one level and a BBL instruction pushes the program counter down one level. Since there are 3 registers in the push down stack, 3 return addresses may be saved. If a fourth JMS occurs, the deepest return address (the first one stored) is lost.

Table IV shows the operation of the address stack.

4. Operation of The Command Lines and the SRC Command

The CPU command lines (CM-ROM, CM-RAM_i) are used to control the ROM's and RAM's by indicating to them how to interpret the data bus content at any given time.

The command lines allow the implementation of RAM bank, chip, register and character addressing, ROM chip addressing, as well as activating the instruction control in each ROM and RAM chip at the time the CPU receives an I/O and RAM group instruction.

In a typical system configuration the CM-ROM line can control up to sixteen 4001's and each CM-RAM_i line can control up to four 4002's.

Each CM-RAM_i line can be selected by the execution of the DCL (Designate Command Line) instruction. The CM-ROM line, however, is always enabled.(1)

(1) If the number of ROM's in the system needs to be more than 16, external circuitry can be used to route CM-ROM to two ROM banks. The same comment applies to the CM-RAM_i lines if more than 16 RAM's need to be used.

For the execution of an I/O and RAM group instruction the following steps are necessary:

(1) The appropriate command line must be selected (by DCL)
(2) The ROM chip and RAM chip, register and character must be selected using the SRC (Send Register Control) instruction.
(3) An I/O and RAM instruction must be fetched (WRM, RDM, WRR, ...)

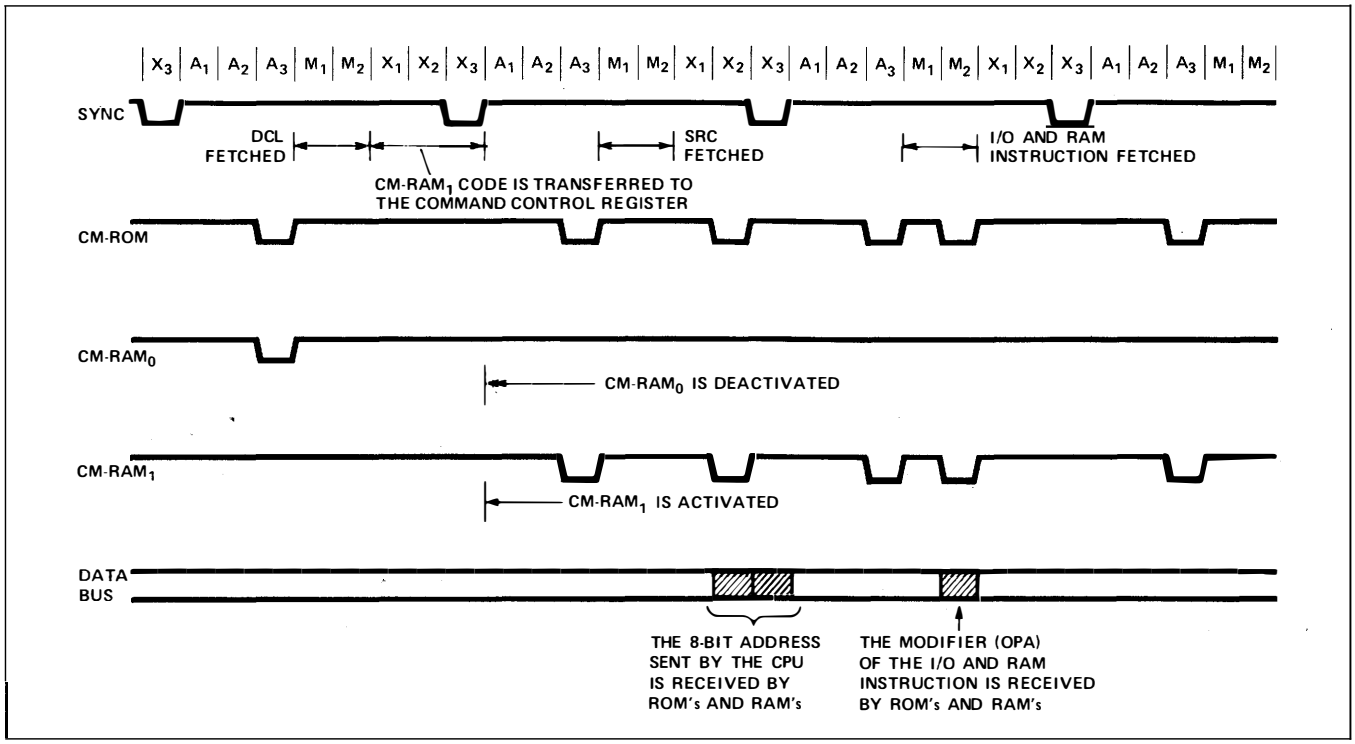


Figure 4. Operation of the Command Control Lines

Following is a detailed explanation of each step.

(1) Prior to execution of the DCL instruction the desired CM-RAM$_i$ code must be stored in the accumulator (for example through an LDM instruction).

(2) During DCL the CM-RAM$_i$ code is transferred from the accumulator to the command control register in the CPU. One CM-RAM$_i$ line is then activated (selecting one RAM bank) during the next instruction which would be an SRC.

The CM-RAM$_i$ code remains in the command control register until a new DCL instruction is received. Each time a new SRC instruction is executed it will operate on the same RAM bank. This allows all RAM and I/O instructions to be executed within the same RAM bank without the necessity of executing another DCL instruction each time. DCL does not affect CM-ROM. **Only the RAM on the designated command line will latch the SRC.**

If up to 4 RAM chips are used in a system, it is convenient to arrange them in a bank controlled by CM-RAM$_0$. This is because CM-RAM$_0$ is automatically selected after the application of at least one RESET (usually at start-up time.) In this case DCL is unnecessary and Step 1 & 2 are omitted).

(3) The SRC instruction specified an index register pair in the CPU, whose content is an 8-bit address (this 8-bit address has previously been stored in the register pair) used to select a RAM chip, register and character and a ROM chip. This address is sent to the data bus during X_2 and X_3 time of the SRC instruction cycle. At X_2 time the CM-ROM line and the selected CM-RAM$_i$ line are in a logic true state to indicate which bank of RAMs and ROMs are to respond to the 8-bit address that is now on the data bus. The 8-bit address is interpreted in the following way:

by the ROM's
- a) The first 4-bits (X_2 time) select one chip out of 16; a flip-flop is set in the selected chip.
- b) The second 4-bits (X_3 time) are ignored.

by the RAM's
- a) The first four bits sent out at X_2 time select one out of four chips and one out of four registers. The two higher order bits (D_3, D_2) select the chip and the two lower order bits (D_1, D_0) select the register.
- b) The second 4-bits (X_3 time) select one 4-bit character out of 16; The address is stored in the address register of the selected chip.

(See Section V for a detailed description of the RAM chip)

(4) At this time one ROM chip and one RAM chip, register and character, have been selected. If the CPU fetches an I/O and RAM instruction, it will cause the CM-ROM and the selected CM-RAM$_i$ line to be logical true at M_2 time. This allows the previously selected ROM's and RAM's to receive the modifier of the instruction. The selected ROM and RAM will decode the instruction (as well as the CPU) and appropriately execute it during the execution time of the same instruction cycle.

It should be added that the CM-ROM and the selected CM-RAM$_i$ lines are always in a logical true state at A_3 time of any instruction cycle.

CM-ROM equals "1" at A3 time indicates to ROM's that the code at A3 time is the chip number of a ROM within their bank. This feature allows the user to expand the system to more than 16 ROM chips.

CM-RAM$_i$ equals "1" at A3 time has no meaning for the RAM chips, however, it could be meaningful if ROM's and RAM's were controlled by a CM-RAM$_i$ line.

Figure 4 summarizes the operation of the command lines in the various instruction cycles.

C. Basic Instruction Set

Table V shows the basic instruction set of the 4004 (CPU). Section VII will describe each instruction in detail.

[Those instructions preceded by an asterisk (*) are 2 word instructions that occupy 2 successive locations in ROM]

MACHINE INSTRUCTIONS (Logic 1 = Low Voltage = Negative Voltage; Logic 0 = High Voltage = Ground)

MNEMONIC	OPR $D_3 D_2 D_1 D_0$	OPA $D_3 D_2 D_1 D_0$	DESCRIPTION OF OPERATION
NOP	0 0 0 0	0 0 0 0	No operation.
*JCN	0 0 0 1 $A_2 A_2 A_2 A_2$	$C_1 C_2 C_3 C_4$ $A_1 A_1 A_1 A_1$	Jump to ROM address $A_2 A_2 A_2 A_2$, $A_1 A_1 A_1 A_1$ (within the same ROM that contains this JCN instruction) if condition $C_1 C_2 C_3 C_4$ [1] is true, otherwise skip (go to the next instruction in sequence).
*FIM	0 0 1 0 $D_2 D_2 D_2 D_2$	R R R 0 $D_1 D_1 D_1 D_1$	Fetch immediate (direct) from ROM Data D_2, D_1 to index register pair location RRR.[2]
SRC	0 0 1 0	R R R 1	Send register control. Send the address (contents of index register pair RRR) to ROM and RAM at X_2 and X_3 time in the Instruction Cycle.
FIN	0 0 1 1	R R R 0	Fetch indirect from ROM. Send contents of index register pair location 0 out as an address. Data fetched is placed into register pair location RRR.
JIN	0 0 1 1	R R R 1	Jump indirect. Send contents of register pair RRR out as an address at A_1 and A_2 time in the Instruction Cycle.
*JUN	0 1 0 0 $A_2 A_2 A_2 A_2$	$A_3 A_3 A_3 A_3$ $A_1 A_1 A_1 A_1$	Jump unconditional to ROM address A_3, A_2, A_1.
*JMS	0 1 0 1 $A_2 A_2 A_2 A_2$	$A_3 A_3 A_3 A_3$ $A_1 A_1 A_1 A_1$	Jump to subroutine ROM address A_3, A_2, A_1, save old address. (Up 1 level in stack.)
INC	0 1 1 0	R R R R	Increment contents of register RRRR. [3]
*ISZ	0 1 1 1 $A_2 A_2 A_2 A_2$	R R R R $A_1 A_1 A_1 A_1$	Increment contents of register RRRR. Go to ROM address A_2, A_1 (within the same ROM that contains this ISZ instruction) if result $\neq 0$, otherwise skip (go to the next instruction in sequence).
ADD	1 0 0 0	R R R R	Add contents of register RRRR to accumulator with carry.
SUB	1 0 0 1	R R R R	Subtract contents of register RRRR to accumulator with borrow.
LD	1 0 1 0	R R R R	Load contents of register RRRR to accumulator.
XCH	1 0 1 1	R R R R	Exchange contents of index register RRRR and accumulator.
BBL	1 1 0 0	D D D D	Branch back (down 1 level in stack) and load data DDDD to accumulator.
LDM	1 1 0 1	D D D D	Load data DDDD to accumulator.

Table V - Basic CPU Instruction Set

INPUT/OUTPUT AND RAM INSTRUCTIONS

(The RAM's and ROM's operated on in the I/O and RAM instructions have been previously selected by the last SRC instruction executed.)

MNEMONIC	OPR $D_3 D_2 D_1 D_0$	OPA $D_3 D_2 D_1 D_0$	DESCRIPTION OF OPERATION
WRM	1 1 1 0	0 0 0 0	Write the contents of the accumulator into the previously selected RAM main memory character.
WMP	1 1 1 0	0 0 0 1	Write the contents of the accumulator into the previously selected RAM output port. (Output Lines)
WRR	1 1 1 0	0 0 1 0	Write the contents of the accumulator into the previously selected ROM output port. (I/O Lines)
WPM	1 1 1 0	0 0 1 1	Write the contents of the accumulator into the previously selected half byte of read/write program memory (for use with 4008/4009 only)
WRϕ [4]	1 1 1 0	0 1 0 0	Write the contents of the accumulator into the previously selected RAM status character 0.
WR1 [4]	1 1 1 0	0 1 0 1	Write the contents of the accumulator into the previously selected RAM status character 1.
WR2 [4]	1 1 1 0	0 1 1 0	Write the contents of the accumulator into the previously selected RAM status character 2.
WR3 [4]	1 1 1 0	0 1 1 1	Write the contents of the accumulator into the previously selected RAM status character 3.
SBM	1 1 1 0	1 0 0 0	Subtract the previously selected RAM main memory character from accumulator with borrow.
RDM	1 1 1 0	1 0 0 1	Read the previously selected RAM main memory character into the accumulator.
RDR	1 1 1 0	1 0 1 0	Read the contents of the previously selected ROM input port into the accumulator. (I/O Lines)
ADM	1 1 1 0	1 0 1 1	Add the previously selected RAM main memory character to accumulator with carry.
RDϕ [4]	1 1 1 0	1 1 0 0	Read the previously selected RAM status character 0 into accumulator.
RD1 [4]	1 1 1 0	1 1 0 1	Read the previously selected RAM status character 1 into accumulator.
RD2 [4]	1 1 1 0	1 1 1 0	Read the previously selected RAM status character 2 into accumulator.
RD3 [4]	1 1 1 0	1 1 1 1	Read the previously selected RAM status character 3 into accumulator.

ACCUMULATOR GROUP INSTRUCTIONS

CLB	1 1 1 1	0 0 0 0	Clear both. (Accumulator and carry)
CLC	1 1 1 1	0 0 0 1	Clear carry.
IAC	1 1 1 1	0 0 1 0	Increment accumulator.
CMC	1 1 1 1	0 0 1 1	Complement carry.
CMA	1 1 1 1	0 1 0 0	Complement accumulator.
RAL	1 1 1 1	0 1 0 1	Rotate left. (Accumulator and carry)
RAR	1 1 1 1	0 1 1 0	Rotate right. (Accumulator and carry)
TCC	1 1 1 1	0 1 1 1	Transmit carry to accumulator and clear carry.
DAC	1 1 1 1	1 0 0 0	Decrement accumulator.
TCS	1 1 1 1	1 0 0 1	Transfer carry subtract and clear carry.
STC	1 1 1 1	1 0 1 0	Set carry.
DAA	1 1 1 1	1 0 1 1	Decimal adjust accumulator.
KBP	1 1 1 1	1 1 0 0	Keyboard process. Converts the contents of the accumulator from a one out of four code to a binary code.
DCL	1 1 1 1	1 1 0 1	Designate command line.

NOTES: [1] The condition code is assigned as follows:

$C_1 = 1$ Invert jump condition $C_2 = 1$ Jump if accumulator is zero $C_4 = 1$ Jump if test signal is a 0

$C_1 = 0$ Not invert jump condition $C_3 = 1$ Jump if carry/link is a 1

[2] RRR is the address of 1 of 8 index register pairs in the CPU.

[3] RRRR is the address of 1 of 16 index registers in the CPU.

[4] Each RAM chip has 4 registers, each with twenty 4-bit characters subdivided into 16 main memory characters and 4 status characters. Chip number, RAM register and main memory character are addressed by an SRC instruction. For the selected chip and register, however, status character locations are selected by the instruction code (OPA).

Table V - Basic CPU Instruction Set (Continued)

IV. 4001 - 256 x 8 MASK PROGRAMMABLE ROM AND 4 BIT I/O PORT

The 4001 performs two basic and distinct functions: As a ROM it stores 256 x 8 words of program or data tables; as a vehicle of communication with peripheral devices it is provided with 4 I/O pins and associated control logic to perform input and output operations. (The block diagram is shown in Figure 5.)

In the ROM mode of operation the 4001 will receive an 8-bit address during A_1 and A_2 time (see Figure 2) and a chip number, together with CM-ROM during A_3 time. When CM-ROM is present, only the chip whose metal option code matches the chip number code sent during A_3 (CSE = "1") is allowed to send data out during the following two cycles: M_1 and M_2. The activity of the 4001 in the ROM mode ends at M_2. Before going into the I/O mode of operation we must first review two basic instructions used in conjunction with it.

1. SRC Instruction (Send address to ROM and RAM)

   When the CPU executes an SRC instruction it will send out 8 bits of data during X_2 and X_3 and will activate the CM-ROM and one CM-RAM(1) line at X_2. Data at X_2, with simultaneous presence of CM-ROM, is interpreted by the 4001 as the chip number of the unit that should later perform an I/O operation. Data at X_3 is ignored. In the case of the 4002, data at X_2 will designate the chip number (one out of 4 chips) and the register number (one out of 4 registers); data at X_3 will designate the 4-bit character (one out of 16) to be operated upon. After SRC only one 4001 and one 4002 will be ready to execute a following I/O instruction.

2. I/O and RAM Instructions

   I/O and RAM instructions allow the CPU to communicate with the I/O ports of the 4001's and 4002's. When the CPU receives an I/O instruction it will activate the CM-ROM and one CM-RAM line during M_2, in time for 4001's and 4002's to receive the second part (OPA) of the I/O instruction. The OPA portion of the I/O instruction is a code specifying which I/O operation should be performed. There are 15 different operations possible. The only ones affecting the 4001 operation are RDR - read ROM port, and WRR - write ROM port.

   In the I/O mode of operation, the selected 4001 (by SRC) after receiving RDR will transfer the information present at its I/O pins to the data bus at X_2. If the instruction received was WRR, the data present on the data bus at $X_2.\emptyset_2$ will be latched on the output flip-flops associated with the I/O lines.

(1) Only one out of four CM-RAM lines is allowed to be activated at any given time. CM-RAM line selection (RAM bank switching) is accomplished by the CPU when a "designate command line" (DCL) instruction is executed. If no DCL is executed prior to SRC, the $CM\text{-}RAM_0$ will automatically be activated at X_2 provided that RESET was applied at least once to the System (most likely at the start-up time). See detailed definition of system instruction in Section VII.

Figure 5 shows the block organization of the 4001. The ROM array has a dynamic mode of operation and is divided into two blocks of 16 x 64 cells each. Multiplexing is needed for both address to address register and data to data bus output buffer operations.

The MTC flip-flop controls the outputting of data. It is set at A_3, (see Figure 2), if CM-ROM and CSE (chip select) are "1". CSE is a single 4-input AND gate of the 4 data bus lines, using Di or $\overline{Di}$ according to the chip number that the user wants to assign to the chip. This is accomplished by metal mask option.

The SRC flip-flop is set by CM-ROM and CSE at X_2, (see Figure 2), and presets the I/O logic for a following input or output operation.

TIMING generates all internal timing signals for the ROM and I/O control using SYNC, ϕ_1 and ϕ_2. A RESET(1) signal will clear all static flip-flops and will inhibit data out.

The output flip-flops associated with I/O pins can also be cleared using an external CL pin.

(1) RESET is used for the start-up of the system.

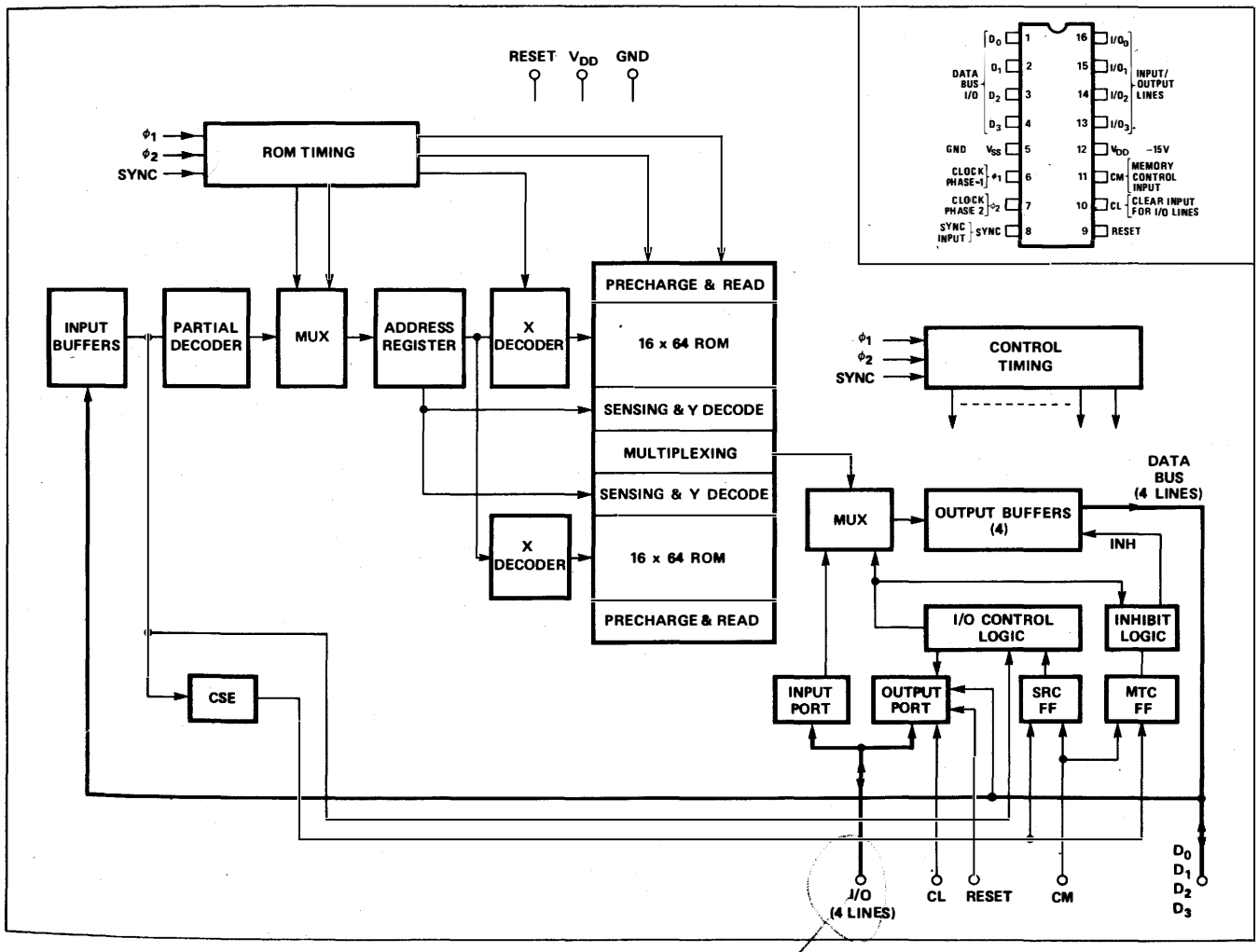


Figure 5. 4001 ROM Block Diagram

ROM Options and Ordering the ROM

Each I/O pin on each ROM can be uniquely chosen to be either an input or output line by metal option. Also each input or output can either be inverted or direct. When the pin is chosen as an input it may have an on-chip resistor connected to either VDD or VSS. Figure 6 shows the available options for each I/O pin.

When ordering a 4001 the following information must be specified:

1. Chip number
2. All the metal options for each I/O pin
3. ROM pattern to be stored in each of the 256 locations.

A blank customer truth table is available upon request from Intel. A copy of this table is shown in the appendix.

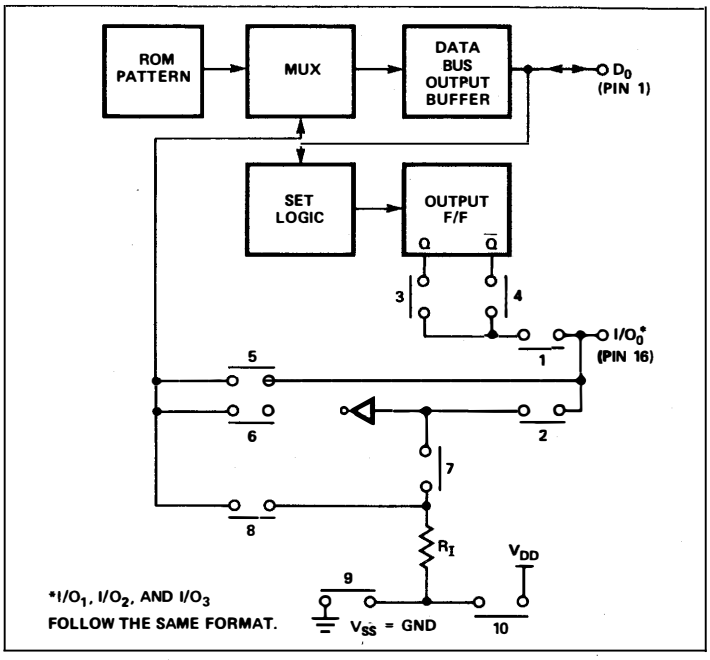


Figure 6. 4001 Available Metal Options for Each I/O Pin

V. 4002 - 320 BIT RAM AND 4 BIT OUTPUT PORT

The 4002 performs two distinct functions. As a RAM it stores 320 bits arranged in 4 registers of twenty 4-bit characters each (16 main memory characters and 4 status characters). As a vehicle of communication with peripheral devices, it is provided with 4 output lines and associated control logic to perform output operations. (The block diagram is shown in Figure 7).

In the RAM mode, the operation is as follows: When the CPU receives an SRC instruction it will send out the content of the designated index register pair during X_2 and X_3 and will activate one CM-RAM line at X_2 for the previously (1) selected RAM bank.

The data at X_2 and X_3 is interpreted as shown below:

X_2		X_3
D_3 D_2 Chip No.	D_1 D_0 Register No.	D_3 D_2 D_1 D_0 Main Memory Character No.
(0 through 3)	(0 through 3)	(0 through 15)

The status character location (0 through 3) as well as the operation to be performed on it are selected by the OPA portion of the I/O and RAM instructions.

---

(1) Bank switching is accomplished by the CPU after receiving a "DCL" (designate command line) instruction. Prior to execution of the DCL instruction the desired CM-RAM code has been stored in the accumulator (for example through an LDM instruction.) During DCL the CM-RAM code is transferred from the accumulator to the CM-RAM register. The RAM bank is then selected starting with the next instruction.

For chip selection, the 4002 is available in two metal options, 4002-1 and 4002-2. An external pin, P_0, is also available for chip selection. The chip number is assigned as follows:

Chip No.	4002 Option	P_0	D_3 D_2 @ X_2
0	4002-1	GND	0 0
1	4002-1	V_{DD}	0 1
2	4002-2	GND	1 0
3	4002-2	V_{DD}	1 1

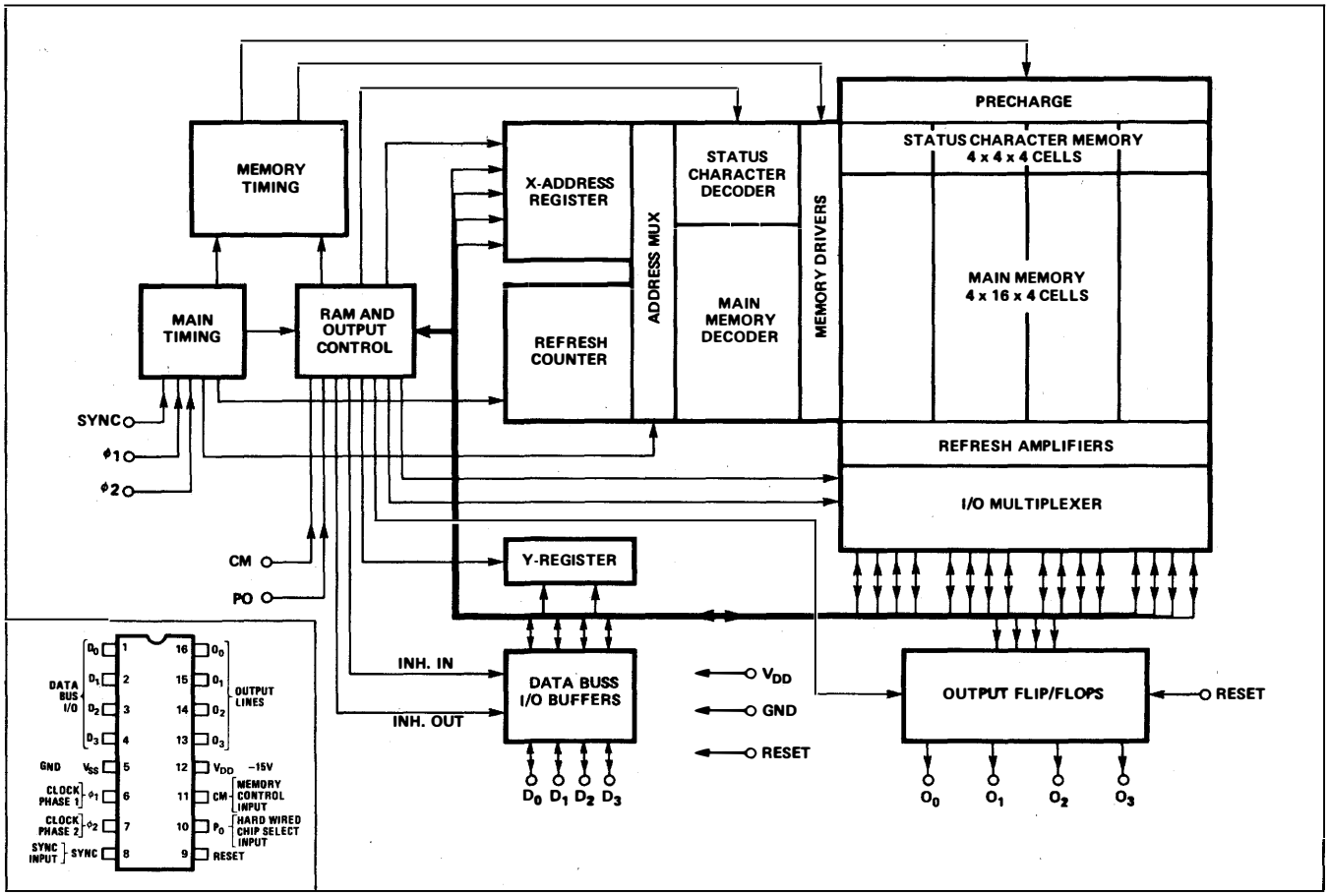


Figure 7. 4002 RAM Block Diagram

Presence of CM-RAM during X_2 tells 4002's that an SRC instruction was received. For a given combination of data at X_2 on D_2, D_3, only the chip with the proper metal option and P_0 state will be ready for the I/O or RAM operation that follows.

The twenty 4-bit characters for each 4002 register are arranged as follows:

1. 16 characters addressable by an SRC instruction: Four 16-character registers constitute the "main" memory.

2. 4 characters addressable by the OPA of an I/O instruction: Four 4-character registers constitute the "status character" memory.

Two separate X decoders switch between main and status character memories.

When an I/O or RAM instruction is received by the CPU, the CPU will activate one CM-RAM line during M_2 , in time for the 4002's to receive the OPA (2nd part of the instruction), which will specify the I/O or RAM operation to be performed. Shown below is a list of the 15 possible I/O and RAM operations.

The I/O and RAM operations are divided into Read operations (IOR) and Write operations (IOW). The state of D_3 will determine if the operation is a read or a write. $D_3 = 1$ for IOR, $D_3 = 0$ for IOW (see Basic Instruction Set, shown in Section IIIc).

For each I/O instruction the action is as shown in the following table:

Instr. Mnem.	4001 I/O Oper.	4002 I/O Oper.	4002 RAM Op.	4001 Data Bus Output Buffer Enabled	4002 Data Bus Output Buffer Enabled	4004 Data Bus Output Buffer Enabled
WRM			x			x
WMP		x				x
WRR	x					x
WRØ			x			x
WR1			x			x
WR2			x			x
WR3			x			x
SBM			x		x	
RDM			x		x	
RDR	x			x		
ADM			x		x	
RDØ			x		x	
RD1			x		x	
RD2			x		x	
RD3			x		x	

In the I/O mode of operation, the selected 4002 chip (by SRC), after receiving the OPA of an I/O instruction (CM-RAM activated at M_2), will decode the instruction.

If the instruction is WMP, the data present on the data bus during $X_2.Ø_2$ will set the output flip-flops associated with the I/O pins. That information will be available until next WMP for peripheral devices control.

An external signal - RESET - when applied to the chip, will cause a clear of all output and control static flip-flops and will clear the RAM array. To completely clear the memory, RESET must be applied for at least 32 instruction cycles (256 clock periods) to allow the internal refresh counter to scan the memory. During RESET the data bus output buffers are inhibited (floating condition).

Figure 7 shows the block organization of the 4002. The RAM array uses a dynamic cell, therefore it must be periodically refreshed. A refresh counter scans the memory array and the memory content is refreshed during an idle portion of the system cycle (M_1 and M_2). An address multiplexer allows loading the content of either the refresh counter or the address register into the decoder.

The RAM control is composed of an SRC flip-flop, chip selection logic, an instruction register, instruction decoder and I/O control logic. This block controls the loading of the address register, the status and main memory decoder switching, the generation of memory timing, the enable of the data bus input-output buffers, the RAM read/write operations, and the loading of the output flip-flops.

VI. 4003 10-BIT SERIAL-IN/PARALLEL-OUT, SERIAL-OUT SHIFT REGISTER

The 4003 is a 10-bit serial-in, parallel-out, serial-out shift register with enable logic. The 4003 is used to expand the number of ROM and RAM I/O ports to communicate with peripheral devices such as keyboards, printers, displays, readers, teletypewriters, etc.

Data is loaded serially and is available in parallel on 10 output lines which are accessed through enable logic. When enabled (E = low), the shift register contents is read out; when not enabled (E = high), the parallel-out lines are at V_{SS}. The serial-out line is not affected by the enable logic.

Data is also available serially permitting an indefinite number of similar devices to be cascaded together to provide shift register length multiples of 10.

The data shifting is controlled by the CP signal. An internal power-on-clear circuit will clear the shift register ($Q_i = V_{SS}$) between the application of the supply voltage and the first CP signal.

The 4003 output buffers are push-pull ratio type, useful for multiple key depression rejection when a 4003 is used in conjunction with a keyboard. In this mode if up to three output lines are connected together, the state of the output is high (Logic "0") if at least one line is high.

The 4003 is a single phase static shift register; however, the clock pulse (CP) maximum width is limited to 10 msec. Data-in and CP can be simultaneous . To avoid race conditions, CP is internally delayed.

Fig. 8 shows the block organization of the 4003.

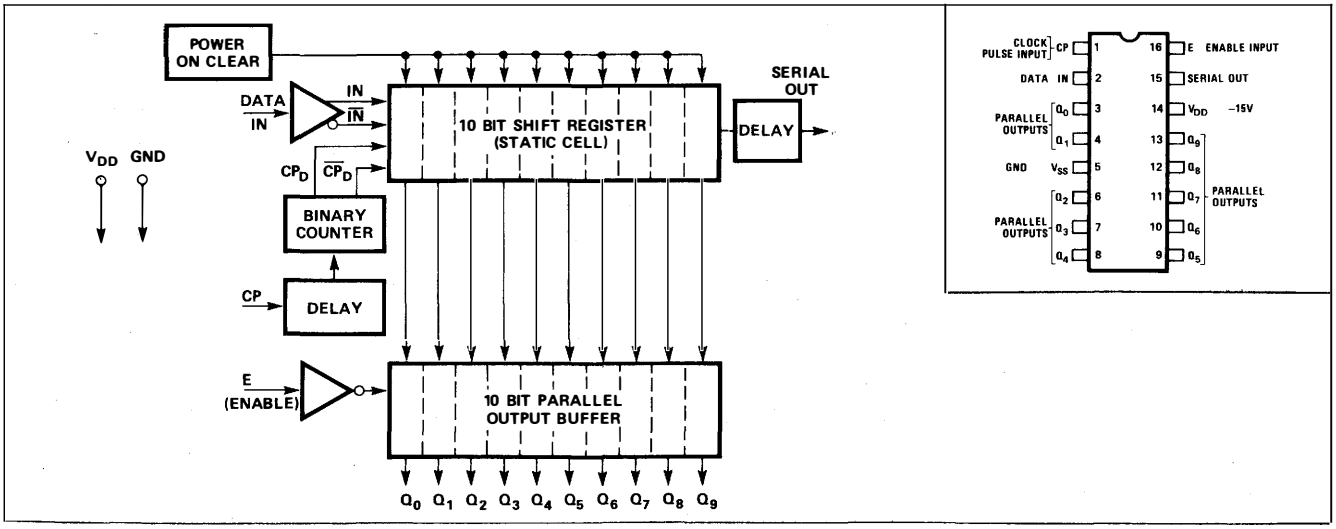


Figure 8. 4003 Shift Register Block Diagram

VII. DETAILED INSTRUCTION REPERTOIRE OF THE MCS-4

A. Instruction Format

As previously discussed, the MCS-4 micro computer set has two types of instruction.

a) 1 word instruction with an 8-bit code and an execution time of 10.8 usec.

b) 2 word instruction with a 16-bit code and an execution time of 21.6 μsec.

Due to the time multiplexed operation of the system, the 8-bit instruction is fetched 4-bits at a time on two successive clock periods. The first 4-bit code is called OPR, the second 4-bit code is called OPA.

The instruction formats were illustrated in Tables I and II

B. Symbols and Abbreviations

The following Symbols and abbreviations will be used thorughout the next few sections:

( )	the content of
$\longrightarrow$	is transferred to
ACC	Accumulator (4-bit)
CY	Carry/link Flip-Flop
ACBR	Accumulator Buffer Register (4-bit)
RRRR	Index register address
RRR	Index register pair address
P_L	Low order program counter Field (4-bit)
P_M	Middle order program counter Field (4-bit)
P_H	High order program counter Field (4-bit)
a_i	Order i content of the accumulator
CM_i	Order i content of the command register
M	RAM main character location
M_{si}	RAM status character i
DB (T)	Data bus content at time T
Stack	The 3 registers in the address register other than the program counter.

Throughout the text "page" means a block of 256 instructions whose address differs only on the most significant 4 bits (all of the instructions on one page are all stored in one ROM).

Example: page 7 means all locations having addresses between 0111 0000 0000 and 0111 1111 1111

C. Format for Describing Each Instruction

Each instruction will be described as follows:

(1) Mnemonic symbol and meaning
(2) OPR and OPA code
(3) Symbolic representation of the instruction
(4) Description of the instruction (if necessary)
(5) Example and/or exceptions (if necessary)

D. One Word Machine Instructions

Mnemonic: **NOP** (No Operation)
OPR OPA: 0000 0000
Symbolic: Not applicable
Description: No operation performed

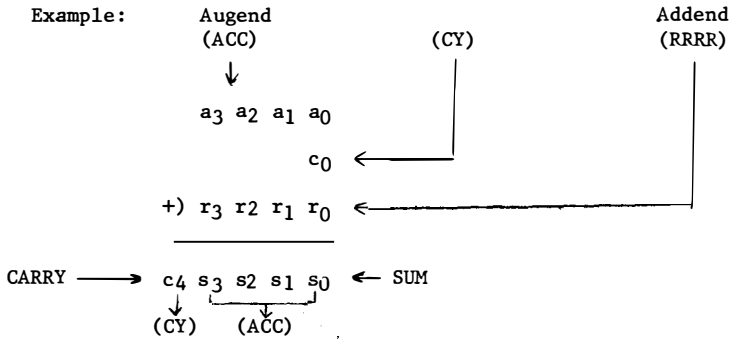
Mnemonic: **LDM** (Load Data to Accumlator)
OPR OPA: 1101 DDDD
Symbolic: DDDD → ACC
Description: The 4 bits of data, DDDD stored in the OPA field of instruction word are loaded into the accumulator. The previous contents of the accumlator are lost. The carry/link bit is unaffected.

Mnemonic: **LD** (Load index register to Accumulator)
OPR OPA: 1010 RRRR
Symbolic: (RRRR)→ACC
Description: The 4 bit content of the designated index register (RRRR) is loaded into the accumulator. The previous contents of the accumulator are lost. The 4 bit content of the index register and the carry/link bit are unaffected.

Mnemonic: **XCH** (Exchange index register and accumlator)
OPR OPA: 1011 RRRR
Symbolic: (ACC)→ ACBR, (RRRR)→ ACC, (ACBR)→ RRRR
Description: The 4 bit content of the designated index register is loaded into the accumulator. The prior content of the accumulator is loaded into the designated register. The carry/link bit is unaffected.

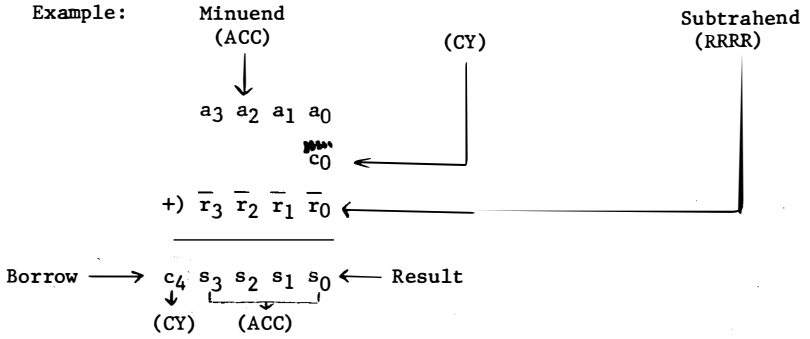
Mnemonic: **ADD** (Add index register to accumulator with carry)
OPR OPA: 1000 RRRR
Symbolic: (RRRR) + (ACC) + (CY) → ACC, CY
Description: The 4 bit content of the designated index register is added to the content of the accumulator with carry. The result is stored in the accumlator. The carry/link is set to 1 if a sum greater than 15_{10} was generated to indicate a carry out; otherwise, the carry/link is set to 0. The 4 bit content of the index register is unaffected.

Example:



Mnemonic: **SUB** (Subtract index register from accumulator with borrow)
OPR OPA: 1001 RRRR
Symbolic: (ACC) + $(\overline{RRRR})$ + $(\overline{CY})$ ⟶ ACC, CY
Description: The 4 bit content of the designated index register is complemented (ones complement) and added to content of the accumulator with borrow and the result is stored in the accumulator. If a borrow is generated, the carry bit is set to 0; otherwise, it is set to 1. The 4 bit content of the index register is unaffected.

Example:



Mnemonic: **INC** (Increment index register)
OPR OPA: 0110 RRRR
Symbolic: (RRRR) +1 ⟶ RRRR
Description: The 4 bit content of the designated index register is incremented by 1. The index register is set to zero in case of overflow. The carry/link is unaffected.

Mnemonic: **BBL** (Branch back and load data to the accumulator)
OPR OPA: 1100 DDDD
Symbolic: (Stack) ⟶ P_L, P_M, P_H; DDDD ⟶ ACC
Description: The program counter(address stack)is pushed down one level. Program control transfers to the next instruction following the last jump to subroutine (JMS) instruction.

The 4 bits of data DDDD stored in the OPA portion of the instruction are loaded to the accumulator.

BBL is used to return from subroutine to main program.

Mnemonic: **JIN** (Jump indirect)
OPR OPA: 0011 RRR1
Symbolic: (RRR0) ⟶ P_M
(RRR1) ⟶ P_L; P_H unchanged
Description: The 8 bit content of the designated index register pair is loaded into the low order 8 positions of the program counter. Program control is transferred to the instruction at that address on the same page (same ROM) where the JIN instruction is located. The 8 bit content of the index register is unaffected.

EXCEPTIONS: When JIN is located at the address (P_H) 1111 1111 program control is transferred to the next page in sequence and not to the same page where the JIN instruction is located. That is, the next address is (P_H + 1) (RRR0) (RRR1) and not (P_H) (RRR0) (RRR1)

Mnemonic: **SRC** (Send register control)
OPR OPA: 0010 RRR1
Symbolic: (RRR0) ⟶ DB (X_2)
(RRR1) ⟶ DB (X_3)
Description: The 8 bit content of the designated index register pair is sent to the RAM address register at X_2 and X_3. A subsequent read, write, or I/O operation of the RAM will utilize this address. Specifically, the first 2 bits of the address designate a RAM chip; the second 2 bits designate 1 out of 4 registers within the chip; the last 4 bits designate 1 out of 16 4-bit main memory characters within the register. This command is also used to designate a ROM for a subsequent ROM I/O port operation. The first 4 bits designate the ROM chip number to be selected. The address in ROM or RAM is not cleared until the next SRC instruction is executed. The 8 bit content of the index register is unaffected.

Mnemonic: **FIN** (Fetch indirect from ROM)
OPR OPA: 0011 RRR0
Symbolic: (P_H) (0000) (0001) $\longrightarrow$ ROM address
(OPR) $\longrightarrow$ RRR0
(OPA) $\longrightarrow$ RRR1

Description: The 8 bit content of the 0 index register pair (0000) (0001) is sent out as an address in the same page where the FIN instruction is located. The 8 bit word at that location is loaded into the designated index register pair. The program counter is unaffected; after FIN has been executed the next instruction in sequence will be addressed. The content of the 0 index register pair is unaltered unless index register 0 was designated.

EXCEPTIONS:
a) Although FIN is a 1-word instruction, its execution requires two memory cycles (21.6 μsec).

b) When FIN is located at address (P_H) 1111 1111 data will be fetched from the next page(ROM) in sequence and not from the same page(ROM) where the FIN instruction is located. That is, next address is $(P_H + 1)$ (0000) (0001) and not (P_H) (0000) (0001).

E. Two Word Machine Instruction

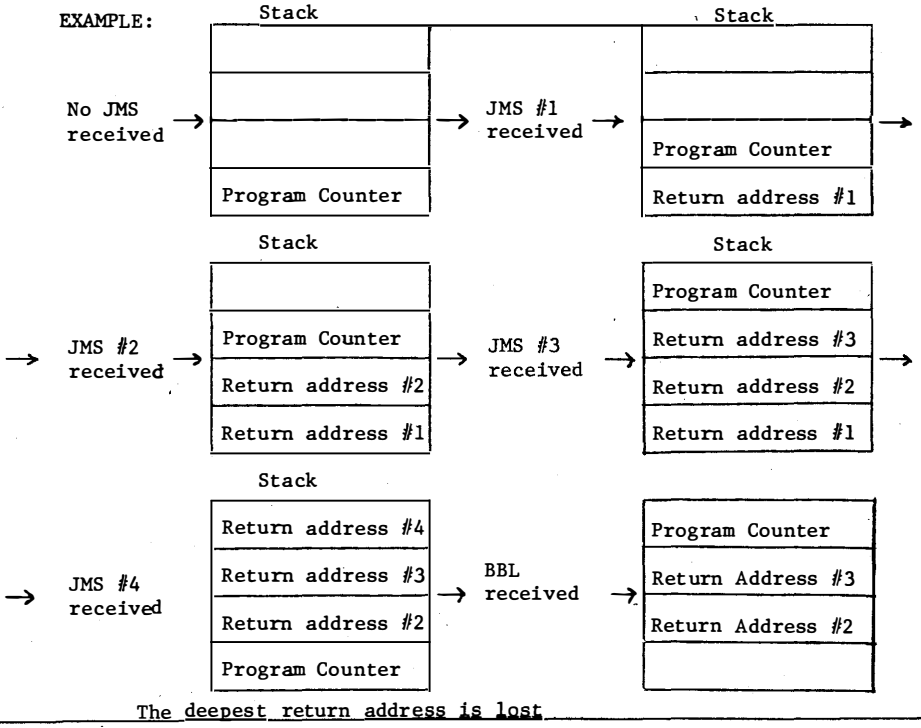
Mnemonic: **JUN** (Jump unconditional)
1st word OPR OPA: 0100 $A_3\ A_3\ A_3\ A_3$
2nd word OPR OPA: $A_2\ A_2\ A_2\ A_2$ $A_1\ A_1\ A_1\ A_1$
Symbolic: $A_1\ A_1\ A_1\ A_1 \longrightarrow P_L$, $A_2\ A_2\ A_2\ A_2 \longrightarrow P_M$, $A_3\ A_3\ A_3\ A_3 \longrightarrow P_H$
Description: Program control is unconditionally transferred to the instruction locater at the address $A_3\ A_3\ A_3\ A_3$, $A_2\ A_2\ A_2\ A_2$, $A_1\ A_1\ A_1\ A_1$.

Mnemonic: **JMS** (Jump to Subroutine)
1st word OPR OPA: 0101 $A_3\ A_3\ A_3\ A_3$
2nd word OPR OPA: $A_2\ A_2\ A_2\ A_2$ $A_1\ A_1\ A_1$
Symbolic: $(P_H, P_M, P_L + 2) \longrightarrow$ Stack
$A_1\ A_1\ A_1\ A_1 \longrightarrow P_L$, $A_2\ A_2\ A_2\ A_2 \longrightarrow P_M$,
$A_3\ A_3\ A_3\ A_3 \longrightarrow P_H$

Description: The address of the next instruction in sequence following JMS (return address) is saved in the push down stack. Program control is transferred to the instruction located at the 12 bit address ($A_3A_3A_3A_3A_2A_2A_2A_2A_1A_1A_1A_1$). Execution of a return instruction (BBL) will cause the saved address to be pulled out of the stack, therefore, program control is transferred to the next sequential instruction after the last JMS.

The push down stack has 4 registers. One of them is used as the program counter, therefore nesting of JMS can occur up to 3 levels.

EXAMPLE:



The deepest return address is lost

Mnemonic: **JCN** (Jump conditional)
1st word OPR OPA: 0001 $C_1C_2C_3C_4$
2nd word OPR OPA: $A_2A_2A_2A_2$ $A_1A_1A_1A_1$
Symbolic: If $C_1C_2C_3C_4$ is true, $A_2A_2A_2A_2 \longrightarrow P_M$
$A_1A_1A_1A_1 \rightarrow P_L$, P_H unchanged
if $C_1C_2C_3C_4$ is false,
$(P_H) \longrightarrow P_H$, $(P_M) \longrightarrow P_M$, $(P_L + 2) \longrightarrow P_L$
Description: If the designated condition code is true, program control is transferred to the instruction located at the 8 bit address $A_2A_2A_2A_2$, $A_1A_1A_1A_1$ on the same page(ROM) where JCN is located.
If the condition is not true the next instruction in sequence after JCN is executed.
The condition bits are assigned as follows:
$C_1 = 0$ Do not invert jump condition
$C_1 = 1$ Invert jump condition
$C_2 = 1$ Jump if the accumulator content is zero
$C_3 = 1$ Jump if the carry/link content is 1
$C_4 = 1$ Jump if test signal (pin 10 on 4004) is zero.

Example:

OPR	OPA	
0001	0110	Jump if accumulator is zero or carry = 1

Several conditions can be tested simutaneously.

The logic equation describing the condition for a jump is give below:

$$JUMP = \overline{C_1} \cdot ((ACC = 0) \cdot C_2 + (CY = 1) \cdot C_3 + \overline{TEST} \cdot C_4) +$$

$$C_1 \cdot \overline{((ACC = 0) \cdot C_2 + (CY = 1) \cdot C_3 + \overline{TEST} \cdot C_4)}$$

EXCEPTIONS: If JCN is located on words 254 and 255 of a ROM page, when JCN is executed and the condition is true, program control is transferred to the 8-bit address on the next page where JCN is located.

Mnemonic: **ISZ** (Increment index register skip if zero)
1st word OPR OPA: 0111 RRRR
2nd word OPR OPA: $A_2A_2A_2A_2$ $A_1A_1A_1A_1$
Symbolic: $(RRRR) + 1 \longrightarrow RRRR$, if result = 0
$(P_H) \longrightarrow P_H$, $(P_M) \longrightarrow P_M$, $(P_L + 2) \longrightarrow P_L$:
if result $\neq 0$ $(P_H) \rightarrow P_H$,
$A2A2A2A2 \rightarrow P_M$, $A1A1A1A1 \rightarrow P_L$
Description: The content of the designated index register is incremented by 1. The accumulator and carry/link are unaffected. If the result is zero, the next instruction after ISZ is executed. If the result is different from 0, program control is transferred to the instruction located at the 8 bit address A2A2A2A2, A1A1A1A1 on the same page (ROM) where the ISZ instruction is located.

EXCEPTIONS: If ISZ is located on words 254 and 255 of a ROM page, when ISZ is executed and the result is not zero, program control is transferred to the 8-bit address located on the next page in sequence and not on the same page where ISZ is located.

Mnemonic: **FIM** (Fetched immediate from ROM)
1st word OPR OPA: 0010 RRR0
2nd word OPR OPA: $D_2D_2D_2D_2$ $D_1D_1D_1D_1$
Symbolic: $D_2D_2D_2D_2 \rightarrow$ RRR0
$D_1D_1D_1D_1 \rightarrow$ RRR1
Description: The 2nd word represents 8-bits of data which are loaded into the designated index register pair.

F. Input/Output and RAM Instructions

(The RAM's and ROM's operated on in the I/O and RAM instructions have been previously selected by the last SRC instruction executed.)

Mnemonic: **RDM** (Read RAM character)
OPR OPA: 1110 1001
Symbolic: (M) $\longrightarrow$ ACC
Description: The content of the previously selected RAM main memory character is transferred to the accumulator. The carry/link is unaffected. The 4-bit data in memory is unaffected.

Mnemonic: **RDO** (Read RAM status character 0)
OPR OPA: 1110 1100
Symbolic: (M_{S0}) $\longrightarrow$ ACC
Description: The 4-bits of status character 0 for the previously selected RAM register are transferred to the accumulator. The carry/link and the status character are unaffected.

Mnemonic: **RD1** (Read RAM status character 1)
OPR OPA: 1110 1101
Symbolic: (M_{S1}) $\longrightarrow$ ACC

Mnemonic: **RD2** (Read RAM status characer 2)
OPR OPA: 1110 1110
Symbolic: (M_{S2}) $\longrightarrow$ ACC

Mnemonic: **RD3** (Read RAM status character 3)
OPR OPA: 1110 1111
Symbolic: (M_{S3}) $\longrightarrow$ ACC

Mnemonic: **RDR** (Read ROM port)
OPR OPA: 1110 1010
Symbolic: (ROM input lines) $\longrightarrow$ ACC
Description: The data present at the input lines of the previously selected ROM chip is transferred to the accumulator. The carry/link is unaffected.
If the I/O option has both inputs and outputs within the same 4 I/O lines, the user can choose to have either "0" or "1" transferred to the accumulator for those I/O pins coded as outputs, when an RDR instruction is executed.

EXAMPLE: Given a 4001 with I/O coded with 2 inputs and 2 outputs, when RDR is executed the transfer is as shown below:

$I_3\ O_2\ O_1\ I_0$ (ACC)

1 X X 0 $\Rightarrow$ 1 (1 or 0) (1 or 0) 0

↖ Input Data ↗ ↖ User can choose ↗

Mnemonic: **WRM** (Write accumulator into RAM character)
OPR OPA: 1110 0000
Symbolic: (ACC) $\longrightarrow$ M
Description: The accumulator content is written into the previously selected RAM main memory character location. The accumulator and carry/link are unaffected.

Mnemonic: **WRO** (Write accumulator into RAM status character 0)
OPR OPA: 1110 0100
Symbolic: (ACC) $\longrightarrow$ M_{S0}
Description: The content of the accumulator is written into the RAM status character 0 of the previously selected RAM register. The accumulator and the carry/link are unaffected.

Mnemonic: **WR1** (Write accumulator into RAM status character 1)
OPR OPA: 1110 0101
Symbolic: (ACC) $\longrightarrow$ M_{S1}

Mnemonic: **WR2** (Write accumulator into RAM status character 2)
OPR OPA: 1110 0110
Symbolic: (ACC) $\longrightarrow$ M_{S2}

Mnemonic: **WR3** (Write accumulator into RAM status character 3)
OPR OPA: 1110 0111
Symbolic: (ACC) $\longrightarrow$ M_{S3}

Mnemonic: **WRR** (Write ROM port)
OPR OPA: 1110 0010
Symbolic: (ACC) $\longrightarrow$ ROM output lines
Description: The content of the accumulator is transferred to the ROM output port of the previously selected ROM chip. The data is available on the output pins until a new WRR is executed on the same chip. The ACC content and carry/link are unaffected. (The LSB bit of the accumulator appears on I/O_0, pin 16, of the 4001). No operation is performed on I/O lines coded as inputs.

Mnemonic: **WMP** (Write memory port)
OPR OPA: 1110 0001
Symbolic: (ACC) $\longrightarrow$ RAM output register
Description: The content of the accumulator is transferred to the RAM output port of the previously selected RAM chip. The data is available on the output pins until a new WMP is executed on the same RAM chip. The content of the ACC and the carry/link are unaffected. (The LSB bit of the accumultor appears on O_0, Pin 16, of the 4002.)

Mnemonic: **ADM** (Add from memory with carry)
OPR OPA: 1110 1011
Symbolic: (M) + (ACC) + (CY) $\longrightarrow$ ACC, CY
Description: The content of the previously selected RAM main memory character is added to the accumulator with carry. The RAM character is unaffected.

Mnemonic: **SBM** (Subtract from memory with borrow)
OPR OPA: 1110 1000
Symbolic: $(\overline{M})$ + (ACC) + $(\overline{CY})$ $\longrightarrow$ ACC, CY
Description: The content of the previously selected RAM character is subtracted from the accumulator with borrow. The RAM character is unaffected.

G. Accumulator Group Instructions

Mnemonic: **CLB** (Clear both)
OPR OPA: 1111 0000
Symbolic: 0 $\longrightarrow$ ACC, 0 $\longrightarrow$ CY
Description: Set accumulator and carry/link to 0.

Mnemonic: **CLC** (Clear carry)
OPR OPA: 1111 0001
Symbolic: 0 $\longrightarrow$ CY
Description: Set carry/link to 0

Mnemonic: **CMC** (Complement carry)
OPR OPA: 1111 0011
Symbolic: $(\overline{CY})$ $\longrightarrow$ CY
Description: The carry/link content is complemented

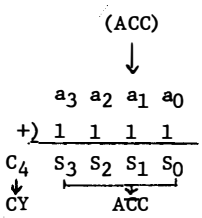
Mnemonic: **STC** (Set carry)
OPR OPA: 1111 1010
Symbolic: 1 $\longrightarrow$ CY
Description: Set carry/link to a 1

Mnemonic: **CMA** (Complement Accumulator)
OPR OPA: 1111 0100
Symbolic: $\overline{a_3}\overline{a_2}\overline{a_1}\overline{a_0}$ $\longrightarrow$ ACC
Description: The content of the accumulator is complemented. The carry/link is unaffected.

Mnemonic: **IAC** (Increment accumulator)
OPR OPA: 1111 0010
Symbolic: (ACC) + 1 $\longrightarrow$ ACC
Description: The content of the accumulator is incremented by 1. No overflow sets the carry/link to 0; overflow sets the carry/link to a 1.

Mnemonic: **DAC** (decrement accumulator)
OPR OPA: 1111 1000
Symbolic: (ACC) - 1 $\longrightarrow$ ACC
Description: The content of the accumulator is decremented by 1. A borrow sets the carry/link to 0; no borrow sets the carry/link to a 1.

EXAMPLE:



Mnemonic: **RAL** (Rotate left)
OPR OPA: 1111 0101
Symbolic: $C_0 \longrightarrow a_0$, $a_i \longrightarrow a_{i+1}$, $a_3 \longrightarrow$ CY
Description: The content of the accumulator and carry/link are rotated left.

Mnemonic: **RAR** (Rotate right)
OPR OPA: 1111 0110
Symbolic: $a_0 \longrightarrow$ CY, $a_i \longrightarrow a_{i-1}$, $C_0 \longrightarrow a_3$
Description: The content of the accumulator and carry/link are rotated right.

Mnemonic: **TCC** (Transmit carry and clear)
OPR OPA: 1111 0111
Symbolic: $0 \longrightarrow$ ACC, (CY) $\longrightarrow a_0$, $0 \longrightarrow$ CY
Description: The accumulator is cleared. The least significant position of the accumulator is set to the value of the carry/link. The carry/link is set to 0.

Mnemonic: **DAA** (Decimal adjust accumulator)
OPR OPA: 1111 1011
Symbolic: (ACC) + 0000 $\longrightarrow$ ACC
or
0110
Description: The accumulator is incremented by 6 if either the carry/link is 1 or if the accumulator content is greater than 9. The carry/link is set to a 1 if the result generates a carry, otherwise it is unaffected.

Mnemonic: **TCS** (Transfer carry subtract)
OPR OPA: 1111 1001
Symbolic: 1001 $\longrightarrow$ ACC if (CY) = 0
1010 $\longrightarrow$ ACC if (CY) = 1
$0 \longrightarrow$ CY
Description: The accumulator is set to 9 if the carry/link is 0.
The accumulator is set to 10 if the carry/link is a 1.
The carry/link is set to 0.

Mnemonic: **KBP** (Keyboard process)
OPR OPA: 1111 1100
Symbolic: (ACC) → KBP ROM → ACC
Description: A code conversion is performed on the accumulator content, from 1 out of n to binary code. If the accumulator content has more than one bit on, the accumulator will be set to 15 (to indicate error). The carry/link is unaffected. The conversion table is shown below

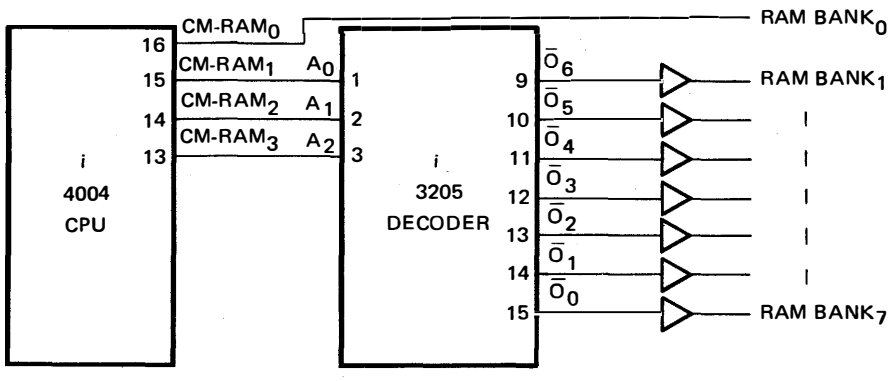
(ACC) before KBP		(ACC) after KBP
0 0 0 0	→	0 0 0 0
0 0 0 1	→	0 0 0 1
0 0 1 0	→	0 0 1 0
0 1 0 0	→	0 0 1 1
1 0 0 0	→	0 1 0 0
0 0 1 1	→	1 1 1 1
0 1 0 1	→	1 1 1 1
0 1 1 0	→	1 1 1 1
0 1 1 1	→	1 1 1 1
1 0 0 1	→	1 1 1 1
1 0 1 0	→	1 1 1 1
1 0 1 1	→	1 1 1 1
1 1 0 0	→	1 1 1 1
1 1 0 1	→	1 1 1 1
1 1 1 0	→	1 1 1 1
1 1 1 1	→	1 1 1 1

Mnemonic: **DCL** (Designate command line)
OPR OPA: 1111 1101
Symbolic: $a_0 \rightarrow CM_0$, $a_1 \rightarrow CM_1$, $a_2 \rightarrow CM_2$
Description: The content of the three least significant accumulator bits is transferred to the comand control register within the CPU.
This instruction provides RAM bank selection when multiple RAM banks are used.(If no DCL instruction is sent out, RAM Bank number zero is automatically selected after application of at lease one RESET) .DCL remains latched until it is changed.

The selection is made according to the following truth table.

(ACC)	CM - RAM_i Enabled	Bank No.
X 0 0 0	CM - RAM_0	Bank 0
X 0 0 1	CM - RAM_1	Bank 1
X 0 1 0	CM - RAM_2	Bank 2
X 1 0 0	CM - RAM_3	Bank 3
X 0 1 1	CM - RAM_1, CM - RAM_2	Bank 4
X 1 0 1	CM - RAM_1, CM - RAM_3	Bank 5
X 1 1 0	CM - RAM_2, CM - RAM_3	Bank 6
X 1 1 1	CM - RAM_1, CM - RAM_2, CM - RAM_3	Bank 7

A 3205 (3 of 8 decoder) or low power TTL equivalent may be tied to the CM-RAM_1, CM-RAM_2, and CM-RAM_3 lines to expand the number of RAM banks to 8. Note that the command lines must be buffered for MOS compatibility. See below.



VIII. AN INTRODUCTION TO PROGRAMMING THE MCS-4

A. Introduction

Writing sequences of instructions for a computer is known as programming. To be able to program a computer effectively, the programmer must understand the action of each of the machine instructions. (The instruction set of the MCS-4 is described in detail in the last section.)

Each machine instruction manipulates data in some way. The data may be the contents of the program counter which indicates where the next instruction is to be found, the contents of one of the CPU registers, accumulator, or carry flip-flop, the contents of RAM or ROM, or the signals at a port.

Programming is probably most easily learned by use of examples. In the pages that follow, a number of sample program segments are described. In general, the examples are shown in order of increasing complexity. These examples have been chosen to illustrate the use of the I/O ports, basic program loops, multiple precision arithmetic, and the use of subroutines.

EXAMPLE #1

Consider the case where it is desired to test the status of a single switch connected to the CPU (4004 chip) on the test input (pin 10). A jump on condition instruction (JCN) can be used to perform this test. Suppose the JCN instruction: JCN TEST, 16 (2 word instruction) is stored at ROM memory locations 2 and 3. The instruction would look as follows:

	OPR	OPA $C_1C_2C_3C_4$
Location #2	0001 (JCN)	0 0 0 1 (Jump if test signal = Logic "0")
Location #3	0001 (Jump to ROM memory Location # 16)	0 0 0 0

When this instruction is executed , if the switch connects a logic "0" (ground) to the test pin of the CPU, the program counter in the address register in the CPU will jump to 16. (That is,the next instruction to be executed would be fetched from ROM Memory location 16). If the switch had been connected to a logic "1" (negative voltage) the program counter would not jump but would be incremented by 1 and hence the instruction in ROM memory location 4 would be executed next. Thus the switch status can be tested simply with one instruction. Furthermore, if it were desired to jump if a test signal equalled a logic "1", the JCN instruction could be coded

	OPR	OPA $C_1C_2C_3C_4$
Location #2	0001	1 0 0 1 Inverted jump condition
Location #3	0001	0 0 0 0

In this case the invert condition bit C_1 is used to indicate a jump is to be made on a logic "1" on the test signal.

If more switches are required a ROM port may be used as shown in the next example.

EXAMPLE #2

Consider the case where it is desired to test the status of a switch connected to the port of ROM #2. To make access to the port, it is necessary to execute and SRC instruction. The SRC instruction utilizes the contents of a pair of registers, which must contain the proper numbers to select the desired port. Register pairs may be most easily loaded using the FIM instruction.

Thus the sequence

Mnemonic	Description
FIM 0 2 , 0	/Fetch immediate (direct) from ROM data (0010, 0000), to index regiser pair 0.
SRC 0	/Send the contents of index register pair 0 to select a ROM. The first 4 bits of data sent out at X_2 time (0010) select ROM #2.
RDR	/Read to contents of the previously selected ROM (ROM #2) input port into the accumulator

has the effect of loading the accumulator with the values appearing at ROM port #2. Individual bits may be tested by shifting them into the carry flip-flop and using a jump on condition instruction. In this manner up to 4 switches can be interrogated from one set of ROM input ports (4 of them).

EXAMPLE #3

Suppose a series of 10 clock pulses must be generated, perhaps to drive the clock line of a 4003 port expander. Let us assume that RAM #3 is to be used. The high order 2 bits of data sent out at X_2 time during an SRC instruction selects the RAM chip. Hence 1100 (binary equivalent of 12) is required at X_2 to select RAM #3.

Since we must select the port on RAM #3 we will require

```
FIM 0
 12, 0
SRC 0
```

This pair of instructions sets up the desired port for use. To generate the clock pulses, we must alternately write a 1 and an 0 into the appropriate port bit. Let us assume that we will only use the high order bit of the port on RAM #3 and that it is initially set at zero (so that the program does not have to reset it). Furthermore, let us assume that we do not care about the other three bits of the port.

First let us set the accumulator to 0

```
LDM  0      /Set accumulator to 0
```

We may then complement the high order bit of the accumulator by the sequence

```
RAL         /Rotate left (accumulator and carry)

CMC         /Complement carry

RAR         /Rotate right (accumulator and carry)
```

which achieves the operation by shifting the bit into the carry flip-flop, complementing it, and shifting it back.

An alternate way to complement the high order bit is to add 8 (binary 1000) to the accumulator. We may set the contents of one register, say register 15, to 8 by the sequence:

```
LDM   8     /Load data DDDD (1000) to the accumulator.

XCH  15     /Exchange contents of index register 15 and accumulator

LDM   0     /Load (0000) to accumulator
```

The first instruction loads the binary number 1000 into the accumulator and the second places the contents of the accumulator into register 15. Since the prior contents of register 15 are also placed in the accumulator, an LDM instruction is then executed to clear the accumulator.

Now the operation ADD 15 will add the binary value 1000 to the accumulator, because Register 15 contains the value 8.

Note the difference in how the LDM and the XCH and ADD instructions utilize the second half of the instruction. The LDM loads the accumulator with the value carried by the instruction i.e. in binary code LDM 8 appears as 1101 1000 and loads the accumulator with 1000. However, the ADD and XCH select a register, and the contents of the register are used as data. That is, ADD 8 would add the contents of register 8 to the accumulator, not the value 8.

To generate the sequence of 10 clock pulses, one could repeat the following 4 instructions 10 times.

```
ADD  15     /Add contents of register 15 (1000
             previously stored in the register)
             to accumulator

WMP         /Write the contents of the accumu-
             lator into the previously selected
             RAM output port

ADD  15

WMP
```

} one clock pulse generated

However, this would take some 40 instructions. The indexing operation available with the ISZ instruction allows a program loop to be repeated 10 times.

The ISZ instruction increments a selected register. If the register initially contained any value other than the value 15 (binary 1111) the instruction performs a JUMP to an address specified by the instruction. This address must be on the same page (within the same ROM) as the instruction immediately following the ISZ.

If however, the register originally contained 15, the CPU will proceed to execute the next instruction in sequence.

By loading a register, say register 14, with the value 6, on the 10th execution of an ISZ, the processor will proceed to the next instruction in sequence rather than jump.

Execution of the ISZ does not affect the accumulator, so that the accumulator does not have to be "saved" prior to its execution.

The program sequence which performs the desired action is then

Instruction #	Address Name	Mnemonic	OPA	Description
(1)		LDM	8	/Load 1000 to accumulator
(2)		XCH	15	/Exchange contents of index register 15 and accumulator
(3)		LDM	6	/Load 0110 to accumulator
(4)		XCH	14	/Exchange contents of index register 14 and accumulator
(5)		FIM	0	/Fetch immediate from ROM, Data (1100 0000)
		12,	0	to index register pair location 0
(6)		SRC	0	/Send address (contents of index register pair 0) to RAM
(7)		LDM	0	/Set accumulator to 0
(8)	→ LOOP	ADD	15	/Add contents of register 15 to accumulator
(9)		WMP		/Write contents of accumulator into RAM output ports
(10)		ADD	15	/Add contents of Register 15 to accumulator
(11)		WMP		/Write contents of accumulator into RAM output ports
(12)		ISZ	14	/Increment contents of register 14. Go
		LOOP		to ROM address A_2, A_1 (called Loop) if result ≠0, otherwise skip.

Explanation of Program

(a) Instruction #1 and #2 - Loads the number 8 (1000) into index register number 15 (1111)

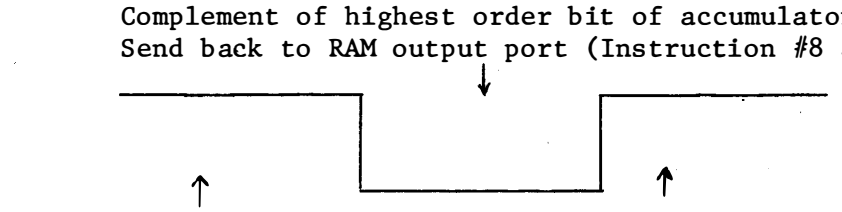
(b) Instruction #3 and #4 - Loads the number 6 (0110) into index register number 14 (1110)

(c) Instruction # 5 - Fetches the address of the desired RAM and stores it in an index register pair

(d) Instruction #6 - Sends the stored address to the RAM bank and selects the desired RAM

(e) Instruction #7 - Initializes the accumulator to 0000.

(f) Instruction #8, 9, 10, and 11 - Generates one clock pulse as follows:



(g) Instruction #12 - The contents of Register 14 is incremented by 1 (0001). The number 7 (0111) is now stored in register 14. Since this result is not equal to zero, program control jumps to the address specified in the 2nd word of this instruction. In this case the address stored in the 2nd word is the address of instruction #8. The program then executes the next 4 instructions in sequence and generates a 2nd clock pulse. This sequence is repeated a total of 10 times, thus generating 10 clock pulses. On the 10th time when the contents of register 14 is incremented it goes to the value 0000 and the program skips to the next instruction in sequence and gets out of the loop.

Example #4

Clock pulse streams of the type derived above are often used to drive groups of 4003 shift registers. It may often be desirable to transfer the contents of a RAM register to a group of 4 shift registers via two output ports. Fig. 9 shows the connection used.

To operate this system, it is necessary to fetch a character from RAM and present it at port #2, then issue the clock pulse at port #1. This sequence requires three SRC commands, one for the RAM selection, one for port #1 selection, and one for port # 2 selection.

In addition, the location in RAM must be incremented each time to provide selection of the next character.

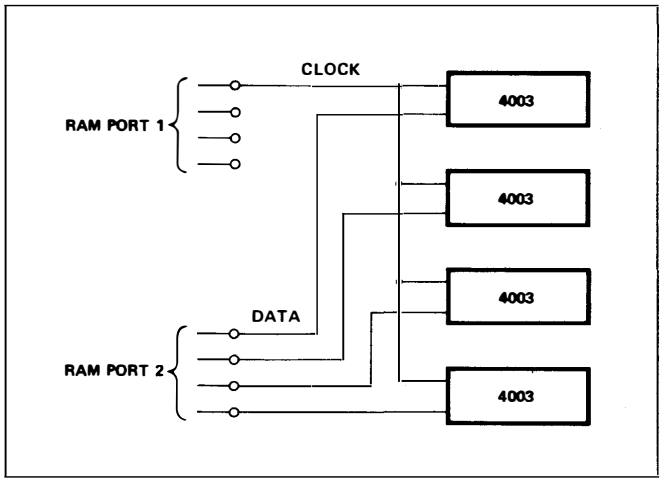


Figure 9. RAM Output Ports Driving Groups of Shift Registers

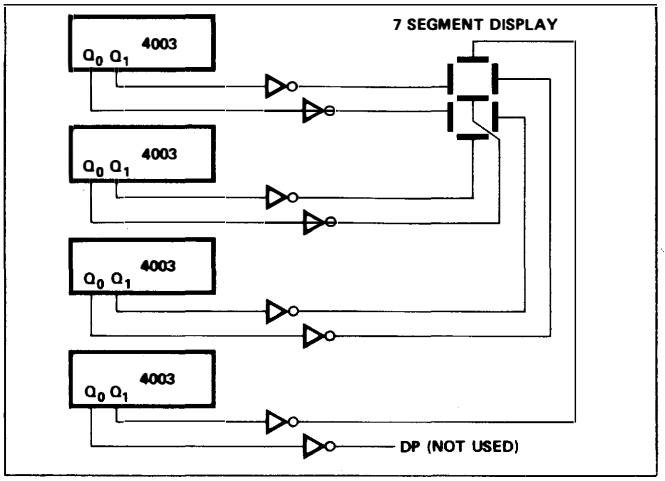


Figure 10. Shift Registers Driving Seven Segment LED Displays

The main loop is then as follows:

```
Loop,  SRC              /Send address to selected RAM
       RDM              /Read selected RAM character into accumulator
       SRC              /Send address to RAM #2
       WMP              /Write contents of accumulator (previously slected RAM
                         character) into Port #2
       SRC              /Send address to RAM #1
       LDM    0         /Set accumulator to "0"
       ADD    15
       WMP
       ADD    15        Generate 1 clock pulse
       WMP
       INC              /Increment by 1 the contents of the register pair holding
                         the selected RAM address
       ISZ    14 Loop   /Increment contents of register 1110.  Jump if result ≠ 0,
                         otherwise skip.
```

The loop above uses 3 pairs of registers for RAM and port selection, and two registers for temporary storage and indexing. The initialization must provide for loading each of these registers.

Example #5

The example above might be extended if for example, the 4003's were driving seven segment LED displays: A 4 line to 7 segment code converter could be used for each display device driven. However, the ROM table lookup capability of the 4004 can be utilized to advantage to save these converters. Suppose the LED displays are wired as shown in Fig.10 with each LED using two adjacent locations in each of the 4003's.

The instruction FIN allows a ROM table to be accessed based on the contents of registers 0 and 1. To save register space, the fetched data may be loaded over the table addresses. The table address may be intialized by an FIM or by the sequence

```
LDM
XCH
```

where the data in the LDM represents the high-order 4-bits of the table address. The low order 4 bits will be derived from the data character itself.

The main loop now becomes as follows:

```
FIM     0       /initial table address
SRC             /fetch data character
RDM             /Read into ACC
XCH     1       /store at register 1
FIN     0       /fetch from ROM table
SRC             /select output port
XCH     0       /fetch lst half of 7 segments
WMP             /transfer to output port
SRC             /select clock port
LDM     0       /Set accumulator to "0"
ADD     15      \
WMP              } /generate one clock pulse
ADD     15       }
WMP             /
SRC             /select output port
XCH     1       /transfer 2nd half of display
WMP             /transfer to output port
SRC             /select clock port
LDM     0       /Set accumulator to "0"
ADD     15      \
WMP              } /generate one clock pulse
ADD     15       }
WMP             /
INC             /set next RAM character
ISZ     14      /test for no. of characters
```

Note that two data characters (8 bits) are transferred for each digit to be displayed.

This loop must be initialized by setting the registers to their initial conditions. The following sequence of 4 instructions is sufficient:

```
FIM             /select RAM register for display
FIM             /initialize clock port selector
FIM             /initialize output port selector
FIM             /initialize no. of digits and set reg. = 8
```

Example #6 - Subroutines

Proceeding with the example outlined above, suppose that the user finds it necessary to display the contents of a number of different RAM registers, at different places in the program. The sequence of instructions could be used whenever this was necessary. However, by making the entire sequence a "subroutine", the user can call out the sequence each time it's needed with only a JMS instruction.

The JMS utilizes the address push down stack. When a JMS is executed, the program counter is pushed up one level and is reloaded with the address to which the jump to take place, and execution will proceed

from this new location. However, before the program counter is reloaded, the old value is saved in the "stack". This stack operates as follows:*

1. Each time a JMS is executed, all addresses saved in the stack are pushed down 1 level. The last value of the program counter is loaded into the top of the stack, the program counter value corresponds to the instruction immediately following the JMS.

2. The BBL instruction raises every entry in the stack one level, with the top value in the stack entering the program counter.

In the example shown, if the RAM register to be transferred to the display is different in different parts of the program, the FIM which selects the RAM register should not be made part of the subroutine. The subroutine would then include the three FIM instructions followed by the main loop and terminated by the BBL.

To display any register from any point in the program, the programmer need use only 4 bytes of ROM:

FIM

JMS

The FIM selects the register and the JMS calls the subroutine.

Example #7: Storing and Fetching a floating point decimal number in the 4002 RAM (How to use the Status and Main Memory Characters in the 4002 RAM)

The 4002 RAM has 4 registers, each with twenty 4-bit characters subdivided into 16 main memory characters and 4 status characters. (320 bits total). Each register is capable of storing a 20 digit, unsigned, fixed point, binary-coded decimal (BCD) number. A more practical usage for the register is the storage of a signed, floating point, BCD number having a 16-digit mantissa (fraction) and a 2-digit exponent.

Consider the number

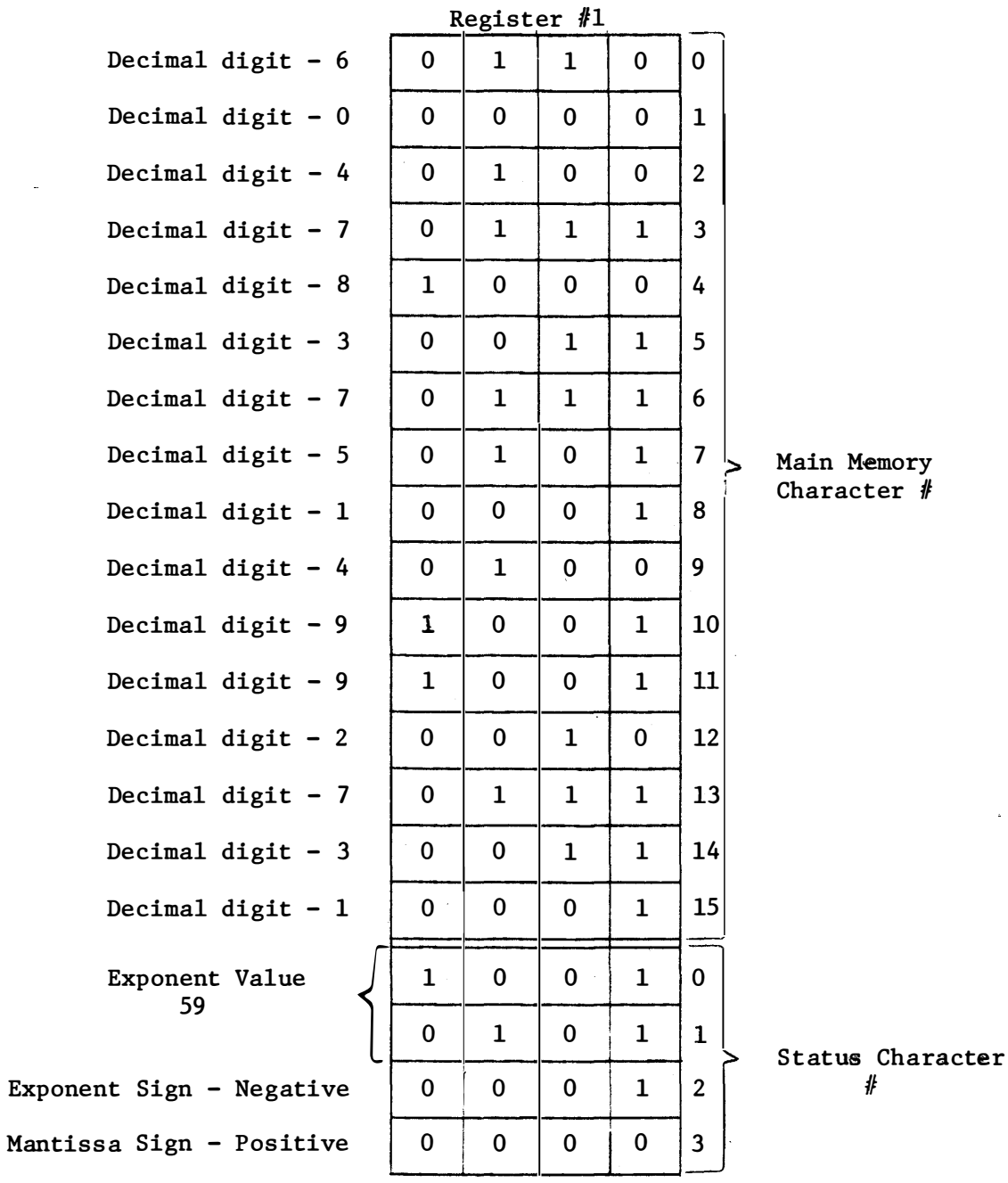
$$+ .1372994157387406 \times 10^{-59}$$

Mantissa (16 digits) Exponent (2 digits)

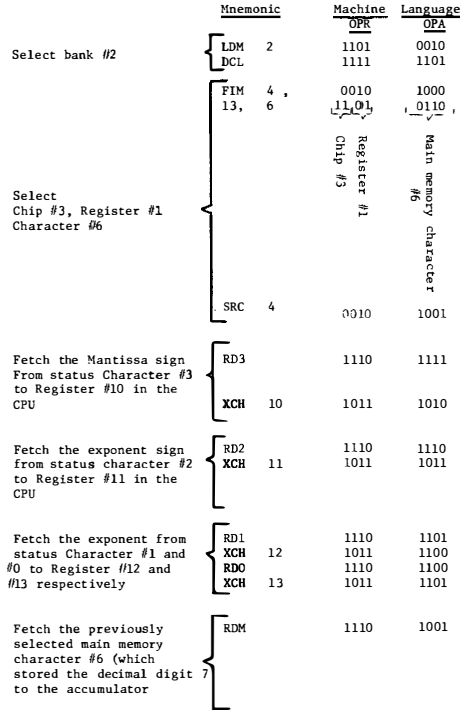
Storage is required for both the sign of the mantissa (in this case positive) and the sign of the exponent (in this case negative), 16 digits of mantissa and 2 digits of exponent. The 4 status characters of the register can be used to hold the signs (in this case a "1" represents minus - this definition is completely arbitrary and is completely up to the user) and the 2 digit exponent. The 16 main memory characters are used to hold the 16 digit mantissa.

* This description of the operation of the address stack is equivalent to the description in Section IIIB (3). It just looks at it from a different viewpoint.

For example let's store the previously shown number in Bank #2, Chip number #3, register #1. It would be stored in the 4002 as follows:



The following instructions would be used to fetch character #6, the signs, and exponent value:



Example 8 - Interpretive Mode

Interpretive mode programming may be used to reduce the amount of ROM required to implement a particular system function. In this mode, data words fetched from ROM or RAM are treated as instructions of a computer which might be quite different than the MCS-4. The MCS-4 program "interprets" the data, using it to call appropriate subroutines which simulate the instructions of the different computer. In effect another computer architecture is simulated.

In the interpretive mode, the instructions of the simulated computer (pseudo instructions) may be derived from RAM or ROM. The instructions are fetched from RAM via the normal RAM operations (SRC, RDM), using a simulated program counter to maintain the address. The JIN instruction is often useful for interpreting the fetched instruction. (Address for the JIN is computed from the fetched pseudo instruction. Each address value is the location of a JMP, or JMS to an appropriate routine, or the routine itself.)

When fetching pseudo instructions from ROM, the FIN is used. As the FIN instruction must be located on the same ROM chip as the fetched data, one cannot use all 256 8-bit bytes of a ROM for pseudo instructions. It is sufficient to allow an FIN followed by a BBL on the ROM chip. Thus up to 254 bytes of each ROM chip can be used for pseudo instructions. The simulated program counter must correspond to this address structure. If the FIN and BBL instructions are located in the first two locations of the ROM chip, the 254 step program address counter can be implemented by initializing the chip address to location 2 rather than location 0. If the interpretive mode program exceeds 254 bytes, the program control routine must determine the proper chip to find the next pseudo instruction. The instruction is then fetched by a JMS to address 0 of the appropriate chip.

IX. PROGRAMMING EXAMPLES

A. MCS-4 Program Routine Format Notes

Routines A, B, and C Assume the Form Shown Below.
Routine D uses Decimal Values for Column 1 and 2.

Example

Column #	1	2	3	4	5	6	7
	0001 0002	0040 0000	ADDITN,	FIM	ϕ<;	ϕ /	IR(ϕ-1)=ϕ

Where:

The first column represents the octal address of this byte.

The second column represents the octal byte value of the instruction word.

The third column is the address label field and can be blank.

The fourth column is the mneumonic field, terminated by a space or a semicolon (;).

The fifth column is the OPA field for the lst byte terminated by a semicolon (;) or space or slash (/) or carriage return.

The sixth column is the second byte specification field (for a 2-word instruction).

The last column is the comment field preceded by a slash (/).

SPECIAL NOTES:

- Each complete line followed by a carriage return is considered a symbolic record.
- All source data following a slash will be considered comment data by the assembler (ignored).
- Any operand followed by a less-than sign (<) will be truncated at three (3) bits and used as an octal numeral.
- The (<) will only work with those instructions which manipulate register pairs in the 4004.
- The semicolon (;) is used to indicate the end of argument for the first byte of a two (2)-byte instruction. Arguments for the second byte must immediately follow the semicolon.

B. 16 - Digit Decimal Addition Routine

This program performs the addition of two 16-digit decimal numbers. These numbers are stored in RAM chip Ø, register Ø and 3. The least significant digit of each of these numbers is located in the character Ø of each RAM register and the most significant digits are in character 15 respectively. The contents in the corresponding characters of the registers are added. If there is a carry it will be added to the next character. The subtotal is stored back into RAM register Ø. Index register 6 is used as a 4-bit binary digit counter. Every time the corresponding digits in the registers are added IR(6) is incremented by one. As the 16th digits are added IR(6) is reset back to zero. Then the program proceeds to check for overflow; i.e., to check whether the carry is "1". If the carry is "1" the program will print out 16 x's, clear RAM register Ø, and jump to location NEXT of the main program that calls for the ADD routine. Otherwise, the program will jump directly to location "NEXT." The following flow chart further clarifies the sequence of the program.

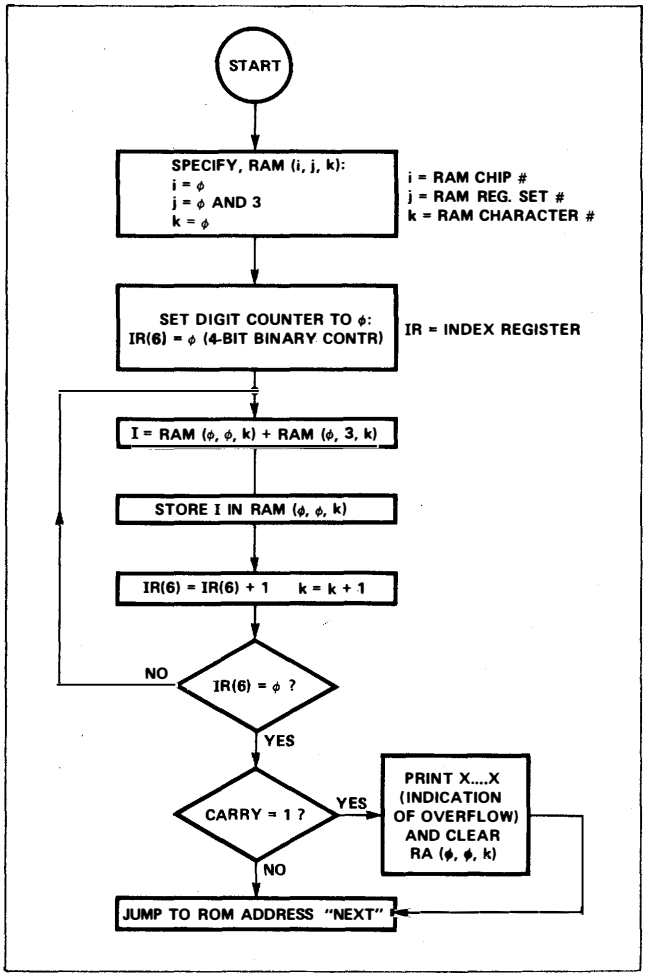


Figure 11. Flow Chart for 16 Digit Decimal Routine

16-DIGIT DECIMAL ADDITION ROUTINE

```
0000 0040
0001 0000 ADDITN, FIM 0<;0          / IR(0-1)=0
0002 0044
0003 0060         FIM 2<;48         / IR(4)=3;IR(5)=0
0004 0320         LDM 0             / LOAD 0 TO AC
0005 0266         XCH 6             / EXCHANGE C(AC) AND IR(6)
0006 0361         CLC               / CLEAR CARRY REG.
0007 0045 AD1,    SRC 2<            / DEFINE RAM ADDRESS $(1
0010 0351         RDM               / READ RAM TO AC
0011 0041         SRC 0<            / DEFINE RAM ADDRESS
0012 0353         ADM               / ADD C(RAM) TO AC, CARRY ENABLED
0013 0373         DAA               / DECIMAL ADDRESS ACC
0014 0340         WRM               / WRITE AC TO RAM
0015 0141         INC 1             / INCREMENT IR(1)
0016 0145         INC 5             / INCREMENT IR(5)
0017 0166
0020 0007         ISZ 6;AD1         / IR(6)=IR(6)+1; SKIP IF C(IR6)=0

0021 0022
0022 0025 OVERFL, JCN CN;XXX        / TEST CARRY; JUMP IF 1
0023 0100
0024 0310         JUN ;NEXT         / SEE NOTE $(2
0025 0320 XXX,    LDM 0             / LOAD AC WITH 0
0026 0272         XCH 10            / EXCHANGE IR(10) AND AC
0027 0042
0030 0330 OVFL1,  FIM 1<;216        / IR(1)=8;IR(2)=13 [X]
0031 0120
0032 0536         JMS ;PRINT
0033 0172
0034 0027         ISZ 10;OVFL1      / IR(19)=IR(10)+1;SKIP IF IR(10)0
0035 0044
0036 0000         FIM 2<;0          / SET IR(4-5)=0
0037 0120
0040 0454         JMS ;CLRRAM       / CLEAR RAM DATA
0041 0100
0042 0310         JUN ;NEXT         / SEE NOTE $(2


/DUMMY ARGUMENTS
CLRRAM=0300
NEXT=0200
PRINT=350
/ $(1 RAM ADDRESSING DEFINE AS TO STANDARDS IN
/ SPEC SHEET.
/ BITS NUMBERED FROM LEFT TO RIGHT MSB TO LSB
/ 0 1 2 3 4 5 6 7
/ BITS 0-1 SELECT RAM CHIP 1 OF 4
/ BITS 2-3 SELECT RAM REGISTER 1 OF 4
/ BITS 4-7 SELECT REGISTER CHARACTER 1 OF 16
/
/ $(2 NEXT, PRINT AND CLRRAM ARE ADDRESS TAGS USED FOR
/ ASSEMBLY
/ NEXT CAN BE THE RETURN POINT OF THIS ROUTINE
/ CLRRAM AND PRINT ARE ROUTINES CALLED BY THIS PROGRAM
/
/
```

C. BCD to Binary Conversion

The following program converts BCD numbers (Ø - 255) to its binary equivalent. In this program it is assumed that a 3 - digit BCD number is previously stored in character Ø, 1, and 2 of register Ø in RAM chip Ø by the main program. Then this program proceeds as follows: First it sets index registers Ø, 1, 2, 3, and 4 to zero (0000), index register 5 to 10 (1010), and index register 6 to 14 (1110). Then the conversion begins by transfering the least significant digit (which is the content of character Ø in the RAM) into index register 3, IR (3). No conversion is made on this digit since it has the same bit pattern as its binary representation. Now recall that each unit value of the second digit of the BCD number (which is the content of character 1 in the RAM) has a value of 10. Hence the program continues as follows: Transfer the second digit to the accumulator (AC) and examine whether the digit is zero. If the digit is not zero the content of the AC is decreased by one (i.e. the value of the second digit is decreased by one), and the result is stored back into the same location in the RAM. Then the content of index register 3 is transfered to AC and the content of index register 5 (which is 10) is added to AC. The result is then stored back into index register 3. Next the content of index register 2, IR (2) is transfered to AC and the content of index register 4 (which is zero) is added to AC; and the result is stored back into index register 2. The process of checking the second digit is repeated until it is down count to zero. Then the program proceeds to set IR (4) to 6 and IR (5) to 4, examines the last BCD digit (which is the content of character 2 in the RAM) and repeats the process in the same manner except, in this case, the content of IR (5) is added to index register 3 and the contents of IR(4) is added to index register 2. This is equivalent to adding 100 (in binary form) to an 8 - bit binary number. The binary number obtained is stored in IR (2) and IR (3). IR (3) contains the lower order 4 bits and IR (2) contains the higher order 4 bits. Index register 6, IR (6), is used as a digit counter to verify that all the 3 BCD digits has been checked.

The following flow chart further explains the details of the program.

BCD TO BINARY CONVERSION ROUTINE

```
0000 0040
0001 0000 BCDBIN, FIM 0<;0        / IR(0-1)=0
0002 0042
0003 0000         FIM 1<;0        / IR(2-3)=0
0004 0044
0005 0012         FIM 2<;10       / IR(4)=0;IR(5)=10
0006 0336         LDM 14          / LOAD AC WITH 14
0007 0266         XCH 6           / EXCHANGE IR(6) AN AC
0010 0041         SRC 0<          / DEFINE RAM ADDRESS
0011 0351         RDM             / READ RAM DATA TO AC
0012 0263         XCH 3           / EXCHANGE AC WITH IR(3)
0013 0141 BDBN,   INC 1           / IR(1)=IR(1)+1
0014 0041         SRC 0<          / DEFINE RAM ADDRESS
0015 0351 BB1,    RDM             / READ RAM DATA TO AC
0016 0024
0017 0033         JCN AZ;BB2      / JUMP IF AC-0
0020 0370         DAC             / AC=AC-1
0021 0340         WRM             / WRITE AC TO RAM
0022 0361         CLC             / CLEAR CARRY REG
0023 0243         LD 3            / LOAD AC WITH C(IR(3))
0024 0205         ADD 5           / ADD IR(5) TO AC
0025 0263         XCH 3           / EXCHANGE IR(3) AND AC
0026 0242         LD 2            / LOAD AC WITH IR(2)
0027 0204         ADD 4           / ADD IR(4) TO AC
0030 0262         XCH 2           / EXCHANGE AC WITH IR(2)
0031 0100
0032 0015         JUN ;BB1        / JUMP UNCONDITIONAL
0033 0044
0034 0144 BB2,    FIM 2<;100      / IR(4)=6;IR(5)=4
0035 0166
0036 0013         ISZ 6;BDBN      / IR(6)=IR(6)+1;SKIP IF IR(6)=0
0037 0300         BBL             / RETURN TO CALLING ROUTINE AC=0
                  /
                  /
```

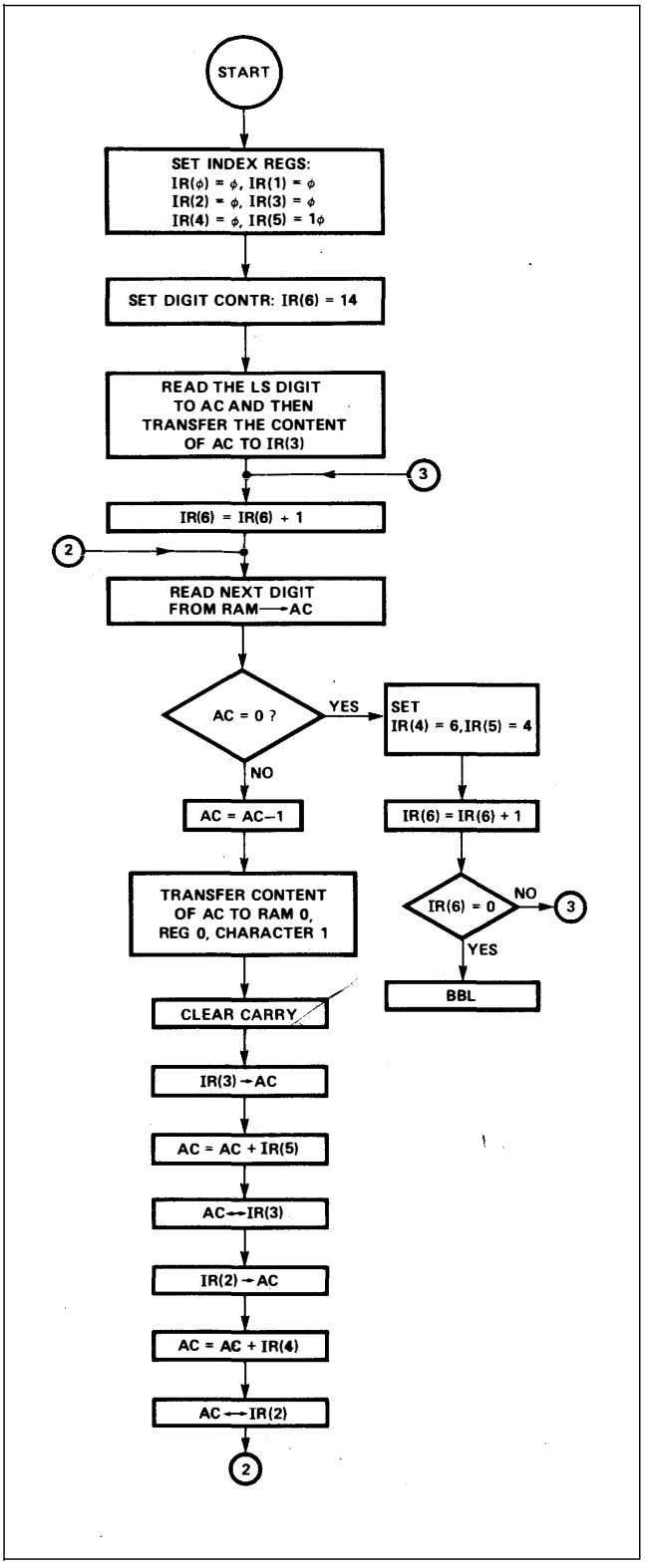


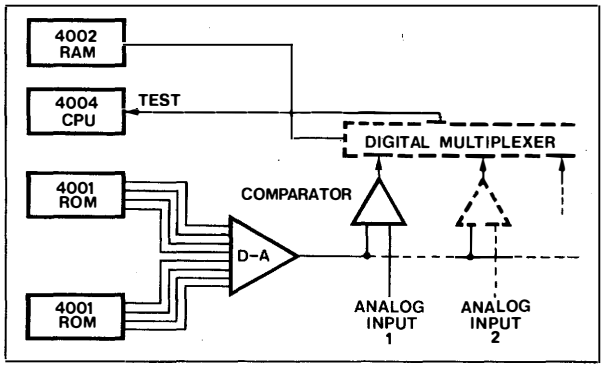
Figure 12. Flow Chart for BCD to Binary Conversion

D. A-D CONVERTER USING DAC With MCS-4

One application using the Intel MCS-4 single-chip computer family is to determine the value of an analog voltage. While it was possible to use the conventional approach of interfacing an analog to digital converter to the microprocessor, a cost saving is achieved by having a microprocessor execute a program which enables a digital to analog converter and a comparator to perform the analog digital converter function. The first figure shows how the conversion is achieved. The MCS-4 uses a "port" for input/output communication. A four-wire port is associated with each read-only memory or read-write memory chip. Two of these output ports have been used to drive the inputs of a digital to analog converter (DAC). The DAC is wired to a comparator which allows the output of the DAC to be compared with the analog input signal. The output of the comparator is in turn wired to the test input of the 4004 central processor. This test input line is interrogated when the central processor executes a certain conditional jump instruction. Whereas the normal instruction execution flow within the MCS-4 system is sequential through program memory, when the conditional jump is executed, the processor jumps to a new location in memory, starting a new instruction sequence.

The second figure lists the program for the analog to digital convertor in MCS-4 assembly language. The program implements a successive approximation conversion technique. Starting with the highest order bit, each bit in turn is turned on and the output of the comparator tested. If turning on the bit results in a signal from the DAC that is larger than the analog input, the bit is turned off and the next bit in turn tested. However, if turning on a bit leaves the output of the digital-to analog converter still smaller than the analog input signal, then that bit will be left turned on. The coding for the program consists of testing each of the lines of one port in turn using in-line coding, then repeating the sequence for the next set of port lines by looping back. Setting a bit is accomplished by loading the accumulator with a load immediate instruction (LDM) and then writing the contents of the accumulator to the output port. The output port is selected at the beginning of the program by the combination of fetch immediate (FIM) and send register control (SRC) instructions. Register #4 (R4) is used to contain the current estimate of the value for the 4-bits being tested. A bit under test is retained or cleared by updating or not updating the contents of register 4. At the end of the basic 4-line test sequence of instructions, the contents of register 4 are saved in an alternate location by a series of exchange (XCH) instructions and the instruction increment and skip on zero (ISZ) is used to perform the function of counting the number of passes through the loop and jumping back to the loop start. The loop selects the next port in turn by the increment (INC) instruction which modified registers RO so that when the next SRC instruction is executed, it will select the next port in sequence. This basic program can be easily modified to handle 12 bit binary or 2 or 3 digit decimal conversions. Execution of the sequence of instructions takes less than one millisecond and as can be seen from the listing, occupies some 29 words of read-only memory.

A multiplexer for multiple analog inputs can be added quite easily by providing a separate comparator for each analog input and performing digital multiplexing at the input to the test terminal of the 4004 central processor. An alternate use of the structure shown in the first figure permits determining which, if any, of the several signals is above or below some predetermined analog threshold value. The analog threshold value is deposited at the output ports driving the DAC and the outputs of the comparators are then read into the MCS-4 system at an input port or at the test terminal of the CPU.



Block Diagram of A-D Converter using DAC and MCS-4

```
/SET UP FOR SELECTION OF ROM OUTPUT PORT (RO, RI=PO),
   USING RI /AS A LOOP COUNTER -- VALUES IN BINARY
0000  00032          FIM PO 00001111B
      00015
/CLEAR REGISTERS R4, R5. (THESE TWO REGISTERS ARE
/DESIGNATED PAIR 2 OR P2 BY THE FIM INSTRUCTION).
R4 AND R5 /WILL BE USED TO RECEIVE THE RESULT OF THE
   CONVERSION
0002  00036          FIM P2 0
      00000
/START OF MAIN LOOP
0004  00033  ADLP,   SRC PO        /SELECT PORT USING CONTENTS OF RO, RI
0005  00240          CLB           /CLEAR ACCUMULATOR AND CARRY FLIP-FLOP
0006  00216          LDM 8         /LOAD ACCUMULATOR WITH 1000
/LDM 8 SETS THE HIGH ORDER BIT OF THE ACCUMULATOR
0007  00226          WRR           /WRITE ACCUMULATOR TO ROM OUTPUT PORT
0008  00025          JCN TI *+3    /JUMP PAST SCH IF RESULT TOO BIG
      00011
0010  00180          XCH R4        /SAVE RESULT IF NOT TOO BIG
/NOW REPEAT FOR 2ND HIGHEST BIT
0011  00212          LDM 4         /LOAD ACCUMULATOR WITH 0100
0012  00132          ADD R4        /ADD RESULT OF PREVIOUS TEST
0013  00226          WRR           /WRITE TO ROM OUTPUT PORT
0014  00025          JCN TI *+3    /JUMP PAST XCH IF RESULT TOO BIG
      00017
0016  00180          XCH R4        /SAVE CURRENT RESULT IF NOT TOO BIG
/REPEAT PROCEDURE FOR LAST TWO BITS OF THIS PORT
0017  11210          LDM 2         /LOAD ACCUMULATOR WITH 0010
0018  00132          ADD R4
0019  00226          WRR
0020  00025          JCN TI *+3
      00023
0022  00180          XCH R4
0023  00209          LDM 1         /LOAD ACCUMULATOR WITH 0001
0024  00132          ADD R4
0025  00226          WRR
0026  00025          JCN TI *+3
      00029
0028  00180          XCH R4
/NOW WRITE FINAL RESULT TO ROM PORT
0029  00164          LD R4         /LOAD FINAL RESULT TO ACCUMULATOR
0030  00226          WRR           /WRITE TO ROM OUTPUT PORT
/NEXT MOVE THESE 4 BITS TO R5 AND CLEAR R4 AND CLEAR R4 FOR NEXT PASS
/NOTE R5 INITIALLY CONTAINED ZERO
0031  00181          XCH R5        /ACCUMULATOR TO R5, R5 TO ACCUMULATOR
0032  00180          XCH R4        /CLEARS R4 IF AT END OF FIRST PASS
0033  00096          INC R0        /PREPARE FOR SELECTION OF NEXT ROM PORT
0034  00113          ISZ RI ADLP   /RETURN FOR SECOND PASS AFTER PASS 1
    · 00004
/AFTER PASS 2, PROGRAM CONTINUES PAST THIS POINT. HIGH ORDER
/BITS OF RESULT WILL BE IN R4, LOW ORDER BITS IN R5.
```

Program for A-D Converter Using DAC and MCS-4

E. MCS-4 SOFTWARE AND FIRMWARE LIBRARY

MCS-4 Assembler and Simulator Software Package

Intel now offers an assembler and simulator software package to help develop programs for micro computer systems built from Intel's MCS-4 set of integrated computer circuits.

The software is written in general Fortran IV for the PDP-10 computer, and may be adapted for most other computers by minor modifications. The package consists of a simulating routine, which enables the computer to simulate the operation of an MCS-4 micro computer, and an assembly routine, used primarily as an aid to programming the simulated micro computer. See Appendix H for complete details.

The routines may be procured from Intel on paper tape or punched cards. Alternatively, designers may contract three nation-wide computer time-sharing services – AL/COM, G.E., and Tymshare – for access to the programs.

SIM4 Hardware Assembler

The SIM4 hardware assembler is a program which translates a symbolic assembly language into bit patterns suitable for MCS-4 control storage programming. It operates on the SIM4-01 or the SIM4-02 micro computer system with an ASR-33 teletype.

The assembler accepts input source text from the teletype keyboard or paper tape reader on each of two required passes. A name table and source listing are created on the first pass. On the second pass, the source text is re-read and a programming paper tape and associated listing are generated. The programming tape is suitable for programming of the 1702 PROM using the MP7-03 programmer system. The same tape may be used for programming the 4001 metal mask ROM. See Appendix F.

SIM4 Hardware Simulator

The SIM4-02 Hardware Simulator is a program written for the MCS-4. This program will provide interactive control over the debugging of other MCS-4 programs. See Appendix G.

The minimum configuration required is a SIM4-02 prototype card with three 4002 RAMs and a Teletype. When fully stuffed with 16 RAMs, test programs up to 512 bytes (locations) in length may be accomodated. The hardware simulation program itself occupies nine full ROMs.

The Hardware Simulation Program has two basic functions:

1. To simulate the execution of a test program, tracing its progress, and apprehending gross errors.
2. To allow the user to dynamically interact with and/or modify his test program, in order to facilitate the debugging process.

These two functions are implemented by means of a set of directives or commands which the user types in at the teletype keyboard. Some of the directives call for typeouts by the simulator program, some of the directives signal the input of data or program modifications, and some of the directives involve both typeouts and input response or data.

4. MCS-4 Program Library

- Text editor, assembler and loader for the MCS-4 that runs on the PDP-8.
- Subroutine for driving a Seiko printer.
- Program which enables a computer to sample liquid levels in bottles.
- RAM test program for the SIM4-01/SIM4-02.
- Program to control the tape motion of an IBM tape drive.
- Hex programmer for the SIM4-01/MP7-03 PROM programmer
- MCS-4 logic subroutines AND, XOR, IOR, LOGIC
- Sixteen Digit Decimal Addition Routine (A0700)[1]
- Exerciser Program (4001-0009)[2]
- Chebychev polynominal approximation subroutines for addition, subtraction, multiplication, division, sine, cosine, arctangent, exponential, and natural logs.
- Teletype Keyboard Input Routine
- PROM Programming Software Package for the SIM4-01/ MP7-03 and SIM4-02/MP7-03 PROM Programming System. (A0540, A0541, A0543)[1]
- A-D Converter using DAC and MCS-4.
- SIM4 Hardware Assembler. Four PROMs (A0740, A0741, A0742, A0743)[1] plug into either SIM4 prototype board enabling assembly of programs on the micro computer itself.
- SIM4 Hardware Simulator. Nine PROMs (A0750-A0758)[1] plug into the SIM4-02 providing capability for program debugging.
- PROM Duplication and Verification Program. (A0544–[1] see Appendix E)
- BCD to Binary Conversion Routine

These program listings are available to all Intel micro computer users. We encourage all users to submit all non-proprietary programs to Intel to add to the program library so that we may make them available to other users.

NOTES:

1. These are the program numbers that should be used when ordering the programs in PROMs.
2. This is the number that should be used when ordering this program. The program is contained in a standard 4001 ROM.

X. INTERFACE DESIGN FOR THE MCS-4 SYSTEM

A. General Discussion

MCS-4 computer systems are often used to replace random logic controllers in a wide variety of systems. In each of these systems a number of peripheral devices, such as keyboards, switches, indicator lamps, numeral displays, printer mechanisms, relays, solenoids, etc., may have to be interrogated or controlled. The engineer who wishes to utilize an MCS-4 system must include, as part of his design, suitable interface circuits and programs.

Devices to be operated or interrogated by an MCS-4 computer are attached to the system via the input and output data ports associated with the 4001 ROM and 4002 ROM. The design of an interface consists of the following steps:

1. Assign peripheral device connections to port connections. If the number of available output ports is insufficient, 4003 output port expanders may be used. When the number of input lines is insufficient, multiplexers must be added. These multiplexers must be controlled by output ports.

2. Develop the necessary level conditioning circuits for each signal. Port inputs and outputs are at MOS levels (logic Ø = 0V with a series output resistance of typically 150Ω, logic 1 = -7v with a series resistance of typically 2k for outputs. Inputs use the same levels, and appear as a capacitive load of approximately 5Pf). These levels must be converted to the levels necessary to drive solenoids, nixies, etc. For TTL compatibility refer to Appendix A.

3. Write the programs necessary to interpret inputs and generate the output levels necessary for proper operation of the peripherals.

Any interface design requires all three of these steps. Each design will typically involve decisions concerning the interaction of the three areas. For example, techniques which reduce the number of output lines may result in more complicated programs.

The following sections describe typical interfaces for a number of common peripheral devices.

B. Keyboards

The MCS-4 can be programmed to scan and debounce a keyboard or can interface to a keyboard which presents precoded (such as ASCII) data. The output lines from a keyboard with precoded data are read at one or more input ports. An input port line or the test line of the 4004 CPU may be interrogated to determine if a key has been pressed.

Scanning and debouncing a keyboard takes a more elaborate program. The keyboard is usually arranged as an n x m (n columns, m rows) matrix of key switches. This type of keyboard is connected as if it had n inputs and m outputs - that is, it requires n output lines from the MCS-4 and m input lines. Under program control, each output is activated in turn. The input ports connected to the keyboard are read and tested to see if a key has been pressed. This testing may utilize the KBP instruction.

After reading (into the ACC) 4 bits corresponding to key status information for one column of the keyboard arrays, execution of the KBP rearranges the data as follows:

1. If no key is pressed (ACC=0000), the ACC remains at 0000.
2. If more than one key is pressed, ACC is set to 1111.
3. If one key is pressed, the ACC indicates the bit position of the key, as shown below.

ACC before		ACC after
0001		0001
0010	KBP	0010
0100	⟶	0011
1000		0100

Scanning of a keyboard is implemented by moving a single "Ø" in a field of "1"s across the lines driving the keyboard inputs. The 4003 shift register is useful for generating the scans. In addition, the 4003 has the characteristic that if two outputs are connected, with one at a logic "1" (-6v) and the other at a logic "Ø", the result will be equivalent to a logic "Ø". By scanning a keyboard with a moving "Ø", multiple key presses in a row can be resolved. Furthermore, if the 4003 is disabled, all outputs go to logic "Ø" and all keys can be sampled simutaneously to determine if a scan is required..

Figure 13 shows the keyboard interface. The ROM inputs are complemented.

Debouncing of the keyboard inputs, etc., is accomplished by testing for the same "press" condition on several successive scans.

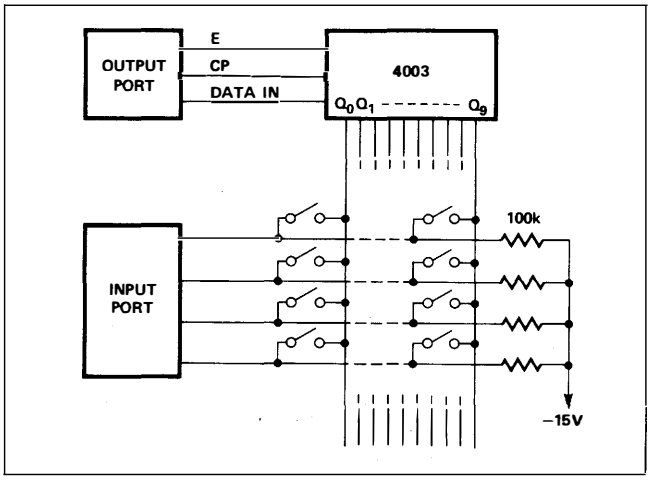


Figure 13. Keyboard Interface - (Scanned Array)

C. Display

Display devices such as NIXIE tubes and LED arrays are easily interfaced to the MCS-4 system. These displays may be DC driven or multiplexed. (In the multiplexed mode, a number of display devices are activated one at a time in rapid sequence. For sufficiently rapid scanning, the eye accepts the data as a continuous display.) To use the multiplexed mode, the display device usually requires some form of coincident selection technique. For example, NIXIE tubes are activated only when the anode supply is present at the same time that the appropriate cathode is grounded (through the proper resistance). In a multiplexed NIXIE array, one set of (10 or 11) cathode drivers is used in combination with one anode driver for each NIXIE tube. Under program control, the array is scanned. One tube is selected; the cathode driver corresponding to the numeral for that position is activated, and then the anode driver for that position is activated for a period. The same steps are executed for the next position in turn.

To avoid flicker, a scan rate of approximately 100 complete scans per second (or higher) should be maintained. This figure allows a scanning program to have up to 60 instruction executions per displayed digit, giving a 16-digit display.

Multiplexed displays typically require high peak driving currents to maintain reasonable average brightness. The drivers used must be capable of supplying the peak currents.

Although the technique described above specifically mentioned NIXIE tubes, the same technique can be applied to 7 segment LED numeral displays.

In systems which combine a numeric display and a keyboard, considerable savings in program memory space and external hardware can be achieved by combining the display scan and keyboard scan. The same loop control and output port logic can be used for keyboard column selection and numeral digit position selection.

D. Teletype Interface

The MCS-4 system is designed to interface with all types of terminal devices. Interface with teletype is a typical example. The interface consists of three simple transistor circuits which is shown in Fig. 15 One transistor is used for receiving serial data from the teletype, one for transmitting data back into the teletype, and the third one for tape reader control.

It requires approximately 100 msec for the teletype to transmit or receive serially 8 data plus 3 control bits. The first and the last bits are idling bits. The second bit is a start bit. The following eight bits are data. Each bit stays on for about 9.09 msec. The MCS-4 system is ideal for this timing control. Following is a simple program which is written for this purpose. This program not only controls the teletype timing but also stores the data temporarily in the index register 2 and 3 in 4004 CPU chip and prints out the character. The flow chart further explains the details of the program.

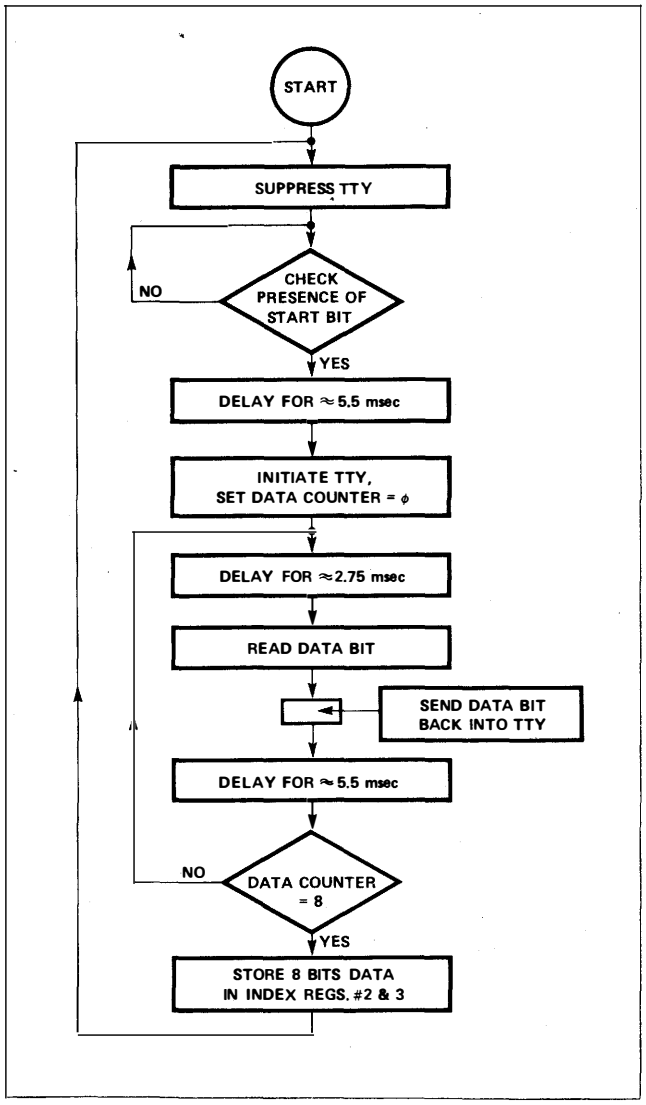


Figure 14. Flow Chart for Teletype Interface

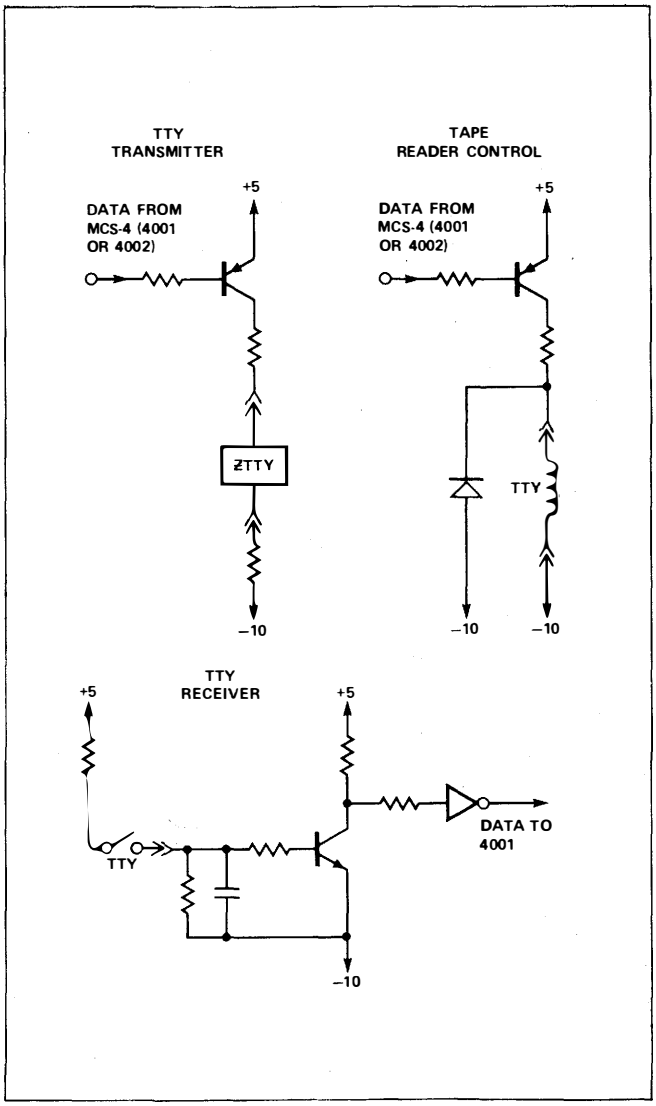


Figure 15. MCS-4 & Teletype Interface Circuits

KEYBOARD INPUT ROUTINE

```
0000 0337 BEGIN,  LDM 15          /SILENCE TTY
0001 0040
0002 0000         FIM 0<;0        / BY SETTINT BIT 3 OF 0-TO 3 TO 1
0003 0041         SRC 0           / DEFINE RAM ADDRESS
0004 0341         WMP             / WRITE DATA TO RAM PORT
0005 0361         CLC
0006 0021
0007 0006 ST,     JCN TZ;ST       / WAIT FOR DATA INPUT SIGNAL
0010 0120
0011 0065         JMS;SBR1        / 5.00MS TIME OUT
0012 0040
0013 0015         FIM 0<;13       / IR(0)=0;IR(1)=13
0014 0161
0015 0014 TEST,   ISZ 1;TEST      / COMPLETE TIMMING FOR BIT SAMPLE
0016 0041         SRC 0<          / DEFINE ROM PORT ADDRESS
0017 0352         RDR             / READ ROM INPUT TO AC
0020 0364         CMA
0021 0341         WMP             / COMPLEMENT DATA AND ECHO
0022 0120
0023 0074         JMS;SBR2        / DO FINAL TIME OUT 300 MS
0024 0040
0025 0000         FIM 0<;0        / IR(0-1)=0
0026 0320         LDM 0
0027 0262         XCH 2           / IR(2)=0
0030 0320         LDM 0
0031 0263         XCH 3           / IR(3)=0
0032 0330         LDM 8
0033 0264         XCH 4           / IR(4)=8
0034 0120
0035 0065 ST1,    JMS ;SBR1
0036 0361         CLC
0037 0041         SRC 0<
0040 0352         RDR             / READ DATA INPUT
0041 0364         CMA
0042 0341         WMP
0043 0366         RAR             / STORE DATA IN CARRY
0044 0242         LD 2            / LOAD AC=IR(2)
0045 0366         RAR             / TRANSFER BIT
0046 0262         XCH 2           / RESTORE NEW DATA WORD
0047 0243         LD 3
0050 0366         RAR
0051 0263         XCH 3           / EXTEND REGISTER TO MAKE 8 BITS
0052 0120
0053 0074         JMS ;SBR2
0054 0164
0055 0034         ISZ 4;ST1
0056 0337         LDM 15
0057 0040
0060 0000         FIM 0<;0
0061 0041         SRC 0<
0062 0341         WMP
0063 0100
0064 0006         JUN ;ST         / RETURN TO INPUT
                  /
                  /
                  /
                  / SUBROUTINES
                  /
0065 0040
0066 0000 SBR1,   FIM 0<;0        / IR(0-1)=0
                                  /5.47MS TIME OUT
0067 0160
0070 0067 L1,     ISZ 0;L1
0071 0161
0072 0067         ISZ 1;L1
0073 0300         BBL
                  /
                  /
0074 0040
0075 0010 SBR2,   FIM 0<;8        / IR(0)=0,IR(1)=8
0076 0160
0077 0076 L2,     ISZ 0;L2        / 2.75 MS TIME OUT
0100 0161
0101 0076         ISZ 1;L2
0102 0300         BBL
                  /
                  /
```

XI. SIM4-01/SIM4-02 PROTOTYPING SYSTEM

A. General System Description

During the development phase of the equipment using the MCS-4 microcomputer set, the designer will often find it helpful to have a means for testing out his program. An interface circuit in which 1701 or 1702 electrically programmable and erasable read only memories simulate the 4001 mask programmable read only memories will help serve this purpose. Using this interface, it is possible for the system designer to program the 1701's or 1702's, plug them into the system, check out the programs and make corrections as necessary. In this way, the development check-out cycle can be typically reduced to one hour or less. With mask programmable ROM's, this cycle is usually four to six weeks. Intel has developed two microcomputer prototyping kits, SIM4-01 and SIM4-02, which use the electrically programmable and erasable ROM.

The maximum directly addressable system configuration is available with the SIM4-02. The 4004 CPU directly controls up to sixteen 1701's or 1702's and up to sixteen RAMs. Eight ROM output ports and eight ROM input ports are provided. These ports are associated with the first eight ROM in the system. Of course, the user must be aware that individual lines of a ROM port can be used for input or output but not both. The input-output option is of course fixed at the time that the 4001 is mask-programmed. Sixteen RAM output ports are also provided. In addition, all data, timing, and memory controls signals are brought to the connector to permit future memory expansion.

The SIM4-01 is designed for small systems. This board contains provision for up to four 1701's or 1702's and four 4002's. It provides up to four RAM output ports (each port contains 4-bits), four ROM output ports and four ROM input ports.

Both systems come complete with the 4004 CPU and four 4002 RAMs. Additional RAMs and ROMs may be added as required. Sockets are provided on the boards for all MCS-4 components and for all ROMs. Note that all programs written for the SIM4-01 may be used without alterations on the SIM4-02.

IMPORTANT

It should be noted that the 1701 and 1702's are described in the data sheet with respect to "positive logic" (high level = p-logic 1). On the other hand the MCS-4 system is defined in terms of "negative logic" (low level = n-logic 1). As a result, when 1701 or 1702 ROM's are being programmed to simulate the 4001, characters should be defined as P = high level = n-logic 0 or an N = low level = n-logic 1. For instance, consider the instruction code for ADM (one of the 45 instructions for the MCS-4).

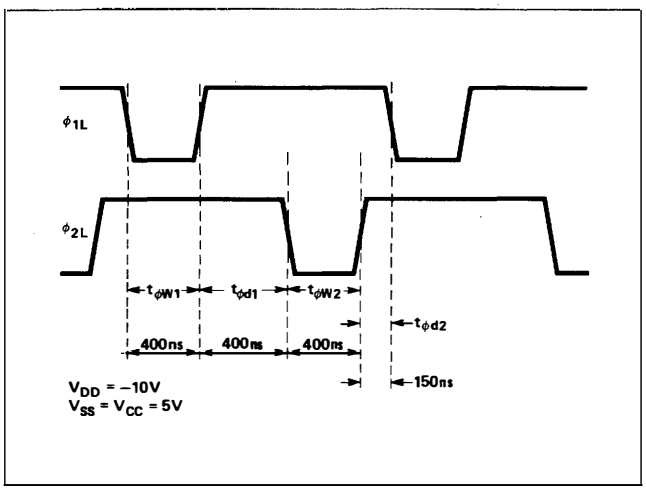
11101011

When preparing the program tape it should be typed,

BNNNPNPNNF

This is the code that will be put into the 4001 when the final system is defined. It will correctly simulate the 4001 operation when the 1701 or 1702 is used with the SIM4-01 or SIM4-02 system.

The schematics and block diagrams for both prototyping systems are shown on the following pages. The 4004 and the 4002's are used as they would be in a conventional system. Additional circuitry is used to simulate the 4001 ROM's. The two phase clocks are generated by the 9602 single shot multivibrator using discrete clock drivers.



Prototype System Clock Drivers

The 9316 counter together with the 3205 - one of eight decoder - serve to decode the cycle timing for the system, thus simulating one of the functions implemented on the 4001 chips. The output of the 3205 decoder indicates which cycle the unit is executing; i.e., the A_1, the A_2, the A_3, the M_1, etc. The discrete transistors serve to convert data bus levels to TTL levels and vice versa. Two 3404 hex latches are wired as the equivalent of three quad latch units. These latches act as address registers for the 1701 or 1702 memory array. The quad latch units are loaded on the A_1, A_2, and A_3 cycles respectively. Those address bits loaded during A_1 and A_2 drive the 1701 or 1702 address line directly, while a 3205 decoder is used to generate the chip select signals for the 1701 or 1702 memory array from the four bits loaded during A_3. Two such decoders would be used if a full array of sixteen 1701's or 1702's were to be utilized. The output of the 1701 or 1702 array is one byte or 8 bits wide. A multiplexer is used to gate four bits at a time onto the four bit wide data bus. The first four bits are selected on the M_1 cycle, the second four bits on the M_2 cycle. The signals at the output of the multiplexer are at TTL levels. These levels are converted to the MOS levels on the 4004 data bus by means of a set of four discrete level shifter circuits. The pull down resistors for these circuits are connected via diode disconnects to a pull down resistor activator circuit. This circuit is activated during the M_1 and M_2 cycles via the two input NAND gate driver. This driver receives two of its three inputs from the M_1 and M_2 decoder.

The balance of the circuitry shown in the schematic is used to implement the input/output port functions associated with the 4001 read only memories. The execution of an SRC instruction (which is used to activate a port) is indicated to the port control circuitry by the presence of the command signal at X_2 time. This condition is decoded and used to load a two latch port selection register. The contents of this register

are in turn decoded by means of four two-input NAND gates. Execution of a port control instruction is indicated by the presence of the command signal ($\overline{CM}$) during M_2 time combined with the appropriate code on the data bus. For instance the READ ROM INPUT condition is detected by a seven-input NAND gate. When this instruction is detected a flip-flop consisting of two-input NAND gates is set. (The presence of a "1" in the port read-control flip-flop is used to enable the inputs from one of two multiplexers onto the data bus during X_2 time.) Data is then transferred into the 4004 from an input port at X_2 time. The port read-control flip-flop is reset at X_3 time so that it will not influence operations on the instruction.

In general, the number of output ports provided by the array of 4002s is adequate. However, to fully duplicate the effects of the 4001s, it may be necessary to implement ROM output ports as well as input ports. Although the two latch port selection register and decoder need not be duplicated, another seven-input NAND gate together with a flip-flop is provided to detect the condition for a WRITE ROM OUTPUT port. A four bit latch is provided for each output port to be implemented. During the subsequent X_2 cycle, the data on the data bus is loaded into the selected port latches. These latches then retain the data. The flip-flop controlling this operation should also be reset at X_3 time. The ROM outputs invert the output data and are TTL compatible. RAM outputs are MOS compatible. Refer to the schematics and pin configurations for both the SIM4-01 and SIM4-02.

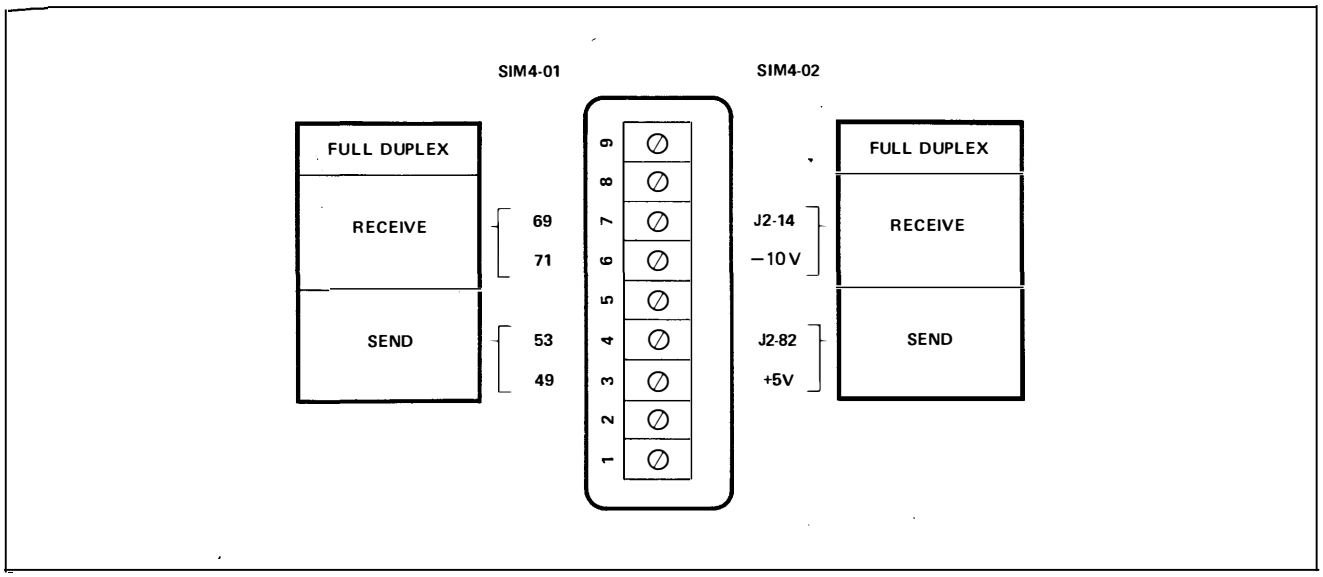
Discrete interface circuits are provided on the cards to communicate with a teletype. Data can be entered through the simulated ROM input ports either from the keyboard or the paper tape reader of the teletype. The receiving and transmitting of data are in serial form. Other terminal devices such as typical commercial keyboards, printers, LED's, CRTs and cassettes can readily communicate with the system with proper single interface.

These systems may be reset to zero by using a RESET switch as indicated on the board pin connector list. Debouncing for the switch is provided on the board.

The TEST signal may be transmitted directly to the TEST pin of the 4004 or through a debouncer and one-shot multivibrator. When the TEST signal comes from the one-shot, the program executed by the CPU should be looping through a JCN instruction waiting for TEST signal.

Teletype Interface

The MCS-4 is designed to operate with all types of terminal devices. A typical example of peripheral interface is the teletype (ASR-33). The SIM4 contains three simple transistor TTY interface circuits. Refer



Teletype Terminal Strip
(See Appendix D for details)

to the appropriate SIM4 schematics for the actual circuit diagrams. One transistor is used for receiving serial data from the teletype, one for transmitting data back to the teletype, and the third for tape reader control.

The teletype must be operating in the full duplex mode. Refer to your teletype operating manual for making connections within the TTY itself. Since all teletypes are not identical, it is impossible to present a general interconnection scheme with either of the SIM4 boards. Many models include a nine terminal barrier strip in the rear of the machine. It is at this point where the connections are made for full duplex operation. The interconnections to the SIM4 for transmit and receive are made at this same point.

To use the teletype reader with the SIM4, the machine must contain a reader power pack. The contacts of a 10V dc relay must be connected in series with the TTY automatic reader (refer to TTY manual) and the coil is connected to the SIM4 tape reader control as shown. This relay must be supplied by the user.

Note that the SIM4 clock generator must remain set at 750 kHz.

In order to sense the start character, data in is also sensed at the TEST input. It requires approximately 110ms for the teletype to transmit or receive eight serial data bits plus three control bits. The first and last bits are idling bits, the second is the start bit, and the following eight bits are data. Each bit stays 9.09ms. While waiting for data to be transmitted, the 4004 is executing a JCN based on the TEST input. When the start character is received, the processor jumps to the TTY processing routine. Under software control, the processor can determine the duration of each bit and strobe the character at the proper time.

A listing of a teletype control program is shown in Section X, Part D.

CAUTION:

In one mode of operation, these prototype systems do not truly simulate the activity of the 4001. After the system is reset and the program counter in the CPU is returned to address zero, a two word instruction in the first two steps of the program may be improperly executed. (This is characteristic of the prototype boards, not of the MCS-4 components.) This is the result of an asynchronous reset pulse applied to the "simulated" 4001 ROM memory.

To insure proper operation of the prototype systems one of the following techniques must be implemented:

1. *Use a NOP in the first location of program memory (ADDRESS 0). Any other single word instruction may also be used.*
2. *Use the SYNC pulse to synchronize the reset signal to the system. Then the prototype system will truly simulate the program memory in the 4001.*

B. SIM4-01 / SIM4-02 Specifications

FEATURES:

• Complete Micro Computer System for Prototyping and/or Production • Reprogrammable ROMs Simulate 4001s • TTY Interface on Clock • Two Phase Clock Generator on Card • Test and Reset Signal Generator on Card

SIM4-01 SPECIFICATIONS

Card Dimensions:
- 8.4 inches high
- 5.7 inches deep

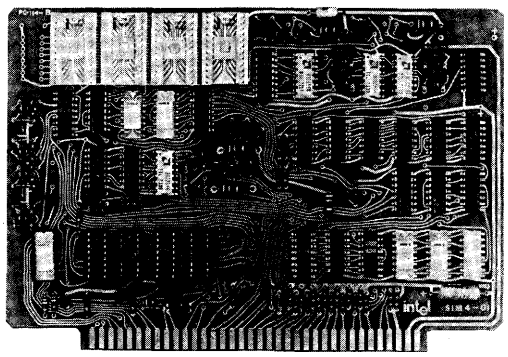
MCS-4 Components included on Board:
(sockets included for memory expansion)
- one 4004
- four 4002s

Maximum Memory Configuration:
- four 4002 RAMs – 320 x 4
- four 1702 ROMs – 1024 x 8

Operating Speed:
- 1.35 μs clock period
- 10.8 μs instruction cycle

DC Power Requirement:
- Voltage–
  - $V_{CC} = V_{SS} = 5V \pm 5\%$
  - TTL GND = 0V
  - $V_{DD} = -10V \pm 5\%$
- Current–
  - No load operation
    - I_{CC} = 1.5 amp
    - I_{DD} = 0.6 amp
  - Worst case loading (16 TTL inputs and outputs)
    - I_{CC} = 1.6 amp
    - I_{DD} = 1.5 amp

Connector:
a. Solder lug type/Amphenol
   72 pin connector
   P/N 225-23621-101
b. Wire Wrap type/Amphenol
   72 pin connector
   P/N 261-15636-2
c. Wire Wrap type/CDC
   72 pin connector
   P/N VPBOIE36300A1



SIM4-01 Prototyping Board

SIM4-02 SPECIFICATIONS

Card Dimensions:
- 11.5 inches high
- 9.5 inches deep

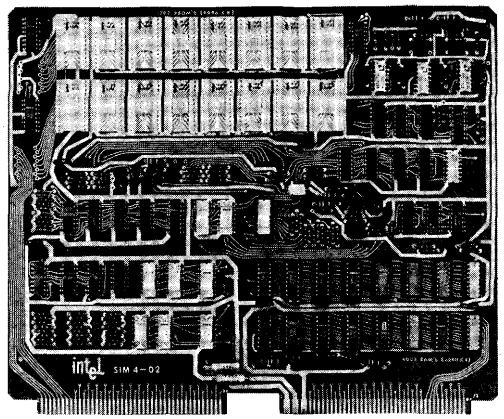
MCS-4 Components included on Board:
(sockets included for memory expansion)
- one 4004
- four 4002s

Maximum Memory Configuration:
- sixteen 4002 RAMs – 1280 x 4
- sixteen 1702 ROMs – 4096 x 8

Operating Speed:
- 1.35 μs clock period
- 10.8 μs instruction cycle

DC Power Requirement:
- Voltage–
  - $V_{CC} = V_{SS} = 5V \pm 5\%$
  - TTL GND = 0V
  - $V_{DD} = -10V \pm 5\%$
- Current–
  - No load operation
    - I_{CC} = 1.8 amp
    - I_{DD} = 0.95 amp
  - Worst case loading (32 TTL inputs and outputs)
    - I_{CC} = 2.75 amps
    - I_{DD} = 1.85 amp

Connector
- Wire Wrap type/Amphenol
- 86 pin connector
- P/N 261-10043-2



SIM4-02 Prototyping Board

C. MCS-4 STANDARD MEMORY AND INTERFACE SET (4008/4009)

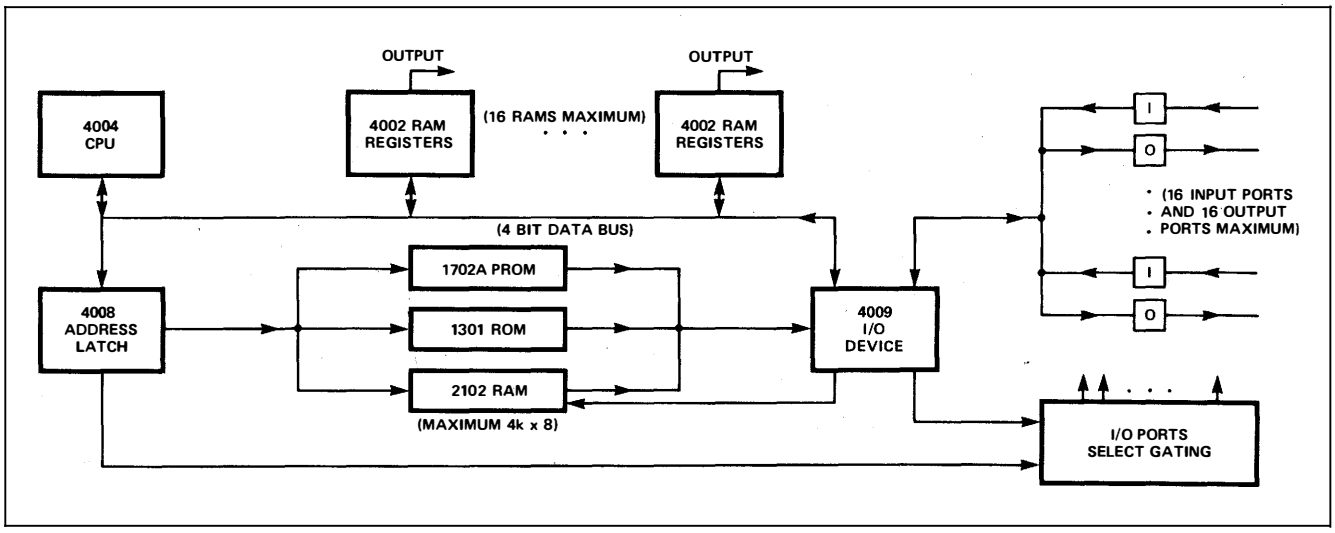
Both prototype systems, the SIM4-01 and SIM4-02 are designed to permit the use of 1702A PROMs instead of metal masked 4001 ROMs. The TTL used in the prototype systems to simulate the control logic of the 4001 is now embodied in two special interface devices. These new devices, the 4008 and 4009, provide direct interface to standard program memory, either ROM or RAM, and to TTL I/O ports.

The 4008 is used as the address latching unit, accepting twelve bits of address in each of three time periods A1, A2, A3. The address is available to the program memory during M1 and M2 when the CPU accepts instructions and data. The program memory may contain up to sixteen 256 byte pages. The 4008 also stores the I/O port selection code so the appropriate input or output port can be selected during the execution times X2 and X3. Demultiplexing of the eight-bit instruction word from program memory and transmission to the data bus is carried out by the 4009 at M1 and M2 time. By way of a four-bit I/O bus which can communicate with up to sixteen input and output ports, data is transmitted to and from the accumulator of the CPU via the 4009.

These silicon gate p-channel MOS devices packaged in 24-pin dual-in-line packages can replace more than twelve packages of standard TTL logic in systems similar to either the SIM4-01 or SIM4-02. Appendix B provides the complete specification for the 4008 and 4009 along with examples of various system configurations.

FEATURES

- **Directly Compatible With 4004 CPU**
- **Interface 1702A PROMs Directly to 4004 CPU — Completely Eliminates TTL Interface**
- **Permits Program Storage in Alterable Memory**
- **Easily Combine PROMs (1702A), Metal Mask ROMs (1301), and RAMs (1101, 2102) for Program Storage**
- **Expanded I/O Port Capability**
- **Each Port May be Both Input and Output — Up to 16 4-bit Input Ports and 16 4-bit Output Ports**
- **Number of I/O Ports is Independent of the Size of the Program Memory**
- **I/O Ports and Control Lines are TTL Compatible**
- **Execute MCS-4 Programs from any Mix of Standard Intel ROMs and RAMs**
- **New Instruction WPM (Write Program Memory) is Used for Loading Alterable Program Storage (RAM)**



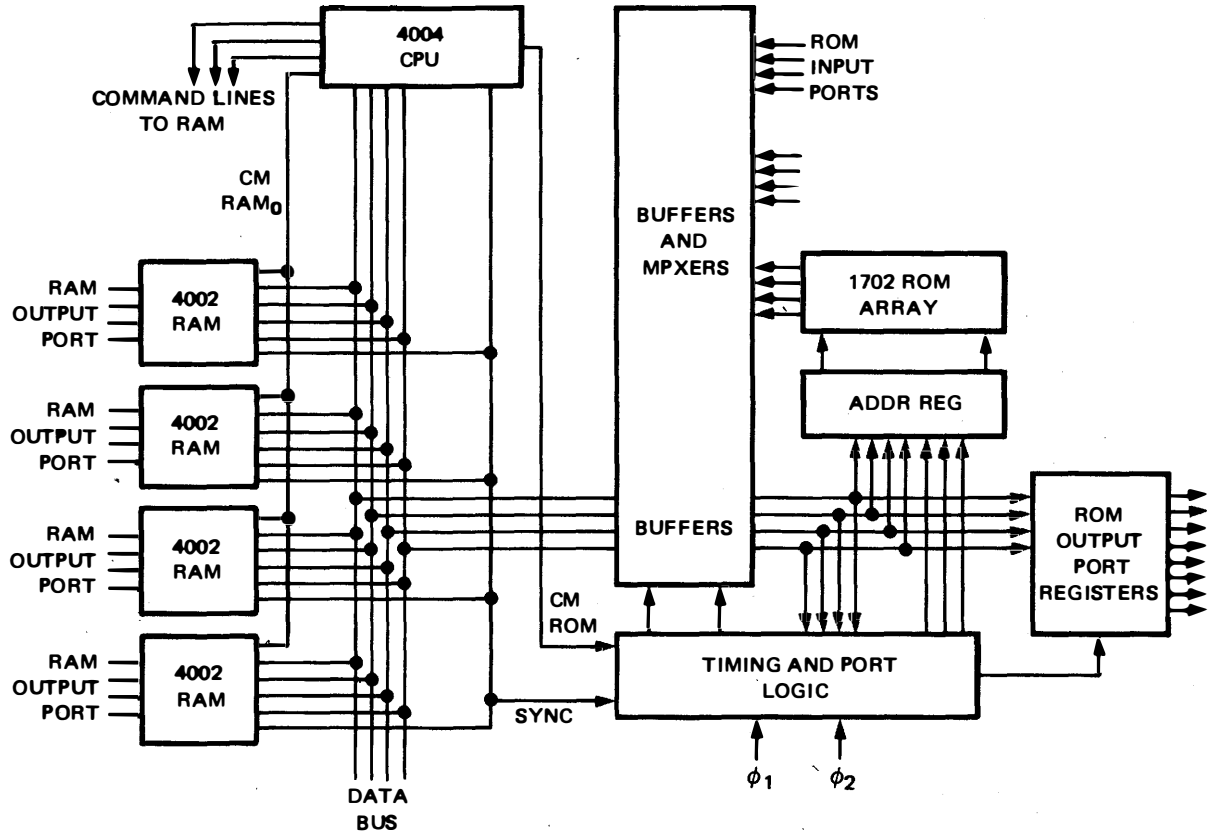
Basic MCS-4 System Using 4008 and 4009

NOTES: UNLESS OTHERWISE SPECIFIED

1. ALL DIODES ARE IN914
2. ALL TRANSISTOR ARE EN2907
3. RESISTOR VALUES ARE IN OHMS 1/4W, 5%.

Figure 16. Detail Drawing — 4001 ROM Simulator Using 1701/1702 as Used on Both the SIM4-01 and SIM4-02 Boards.

D. SIM4-01 Prototype System



MCS-4 System Using 1701 PROM

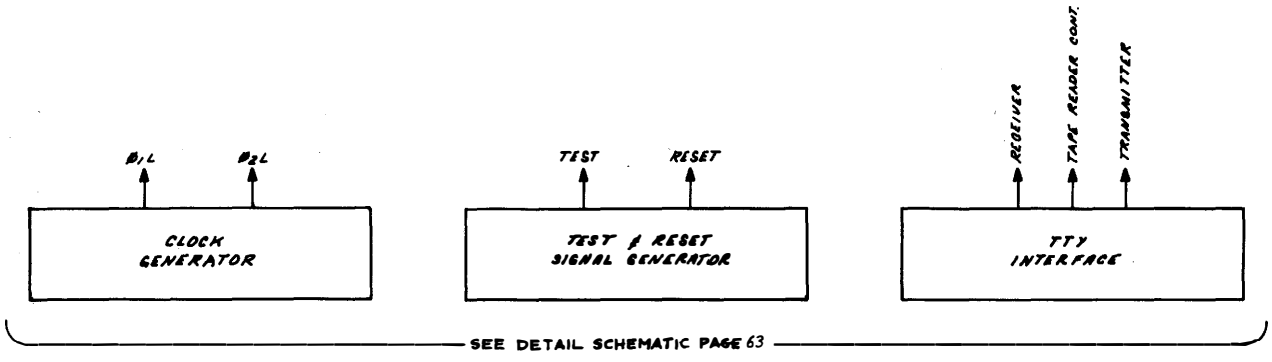


Figure 17. SIM4-01 System Block Diagram

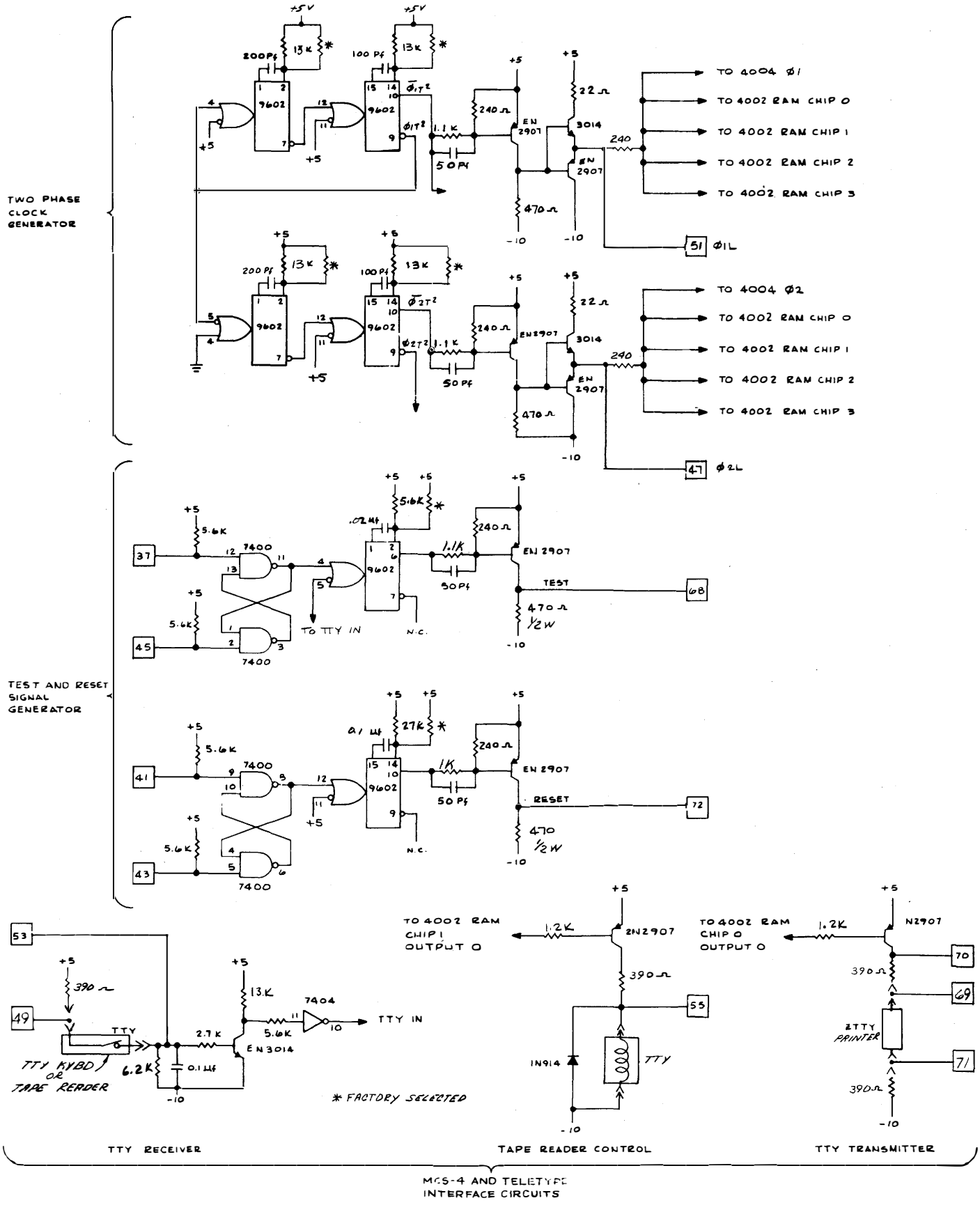
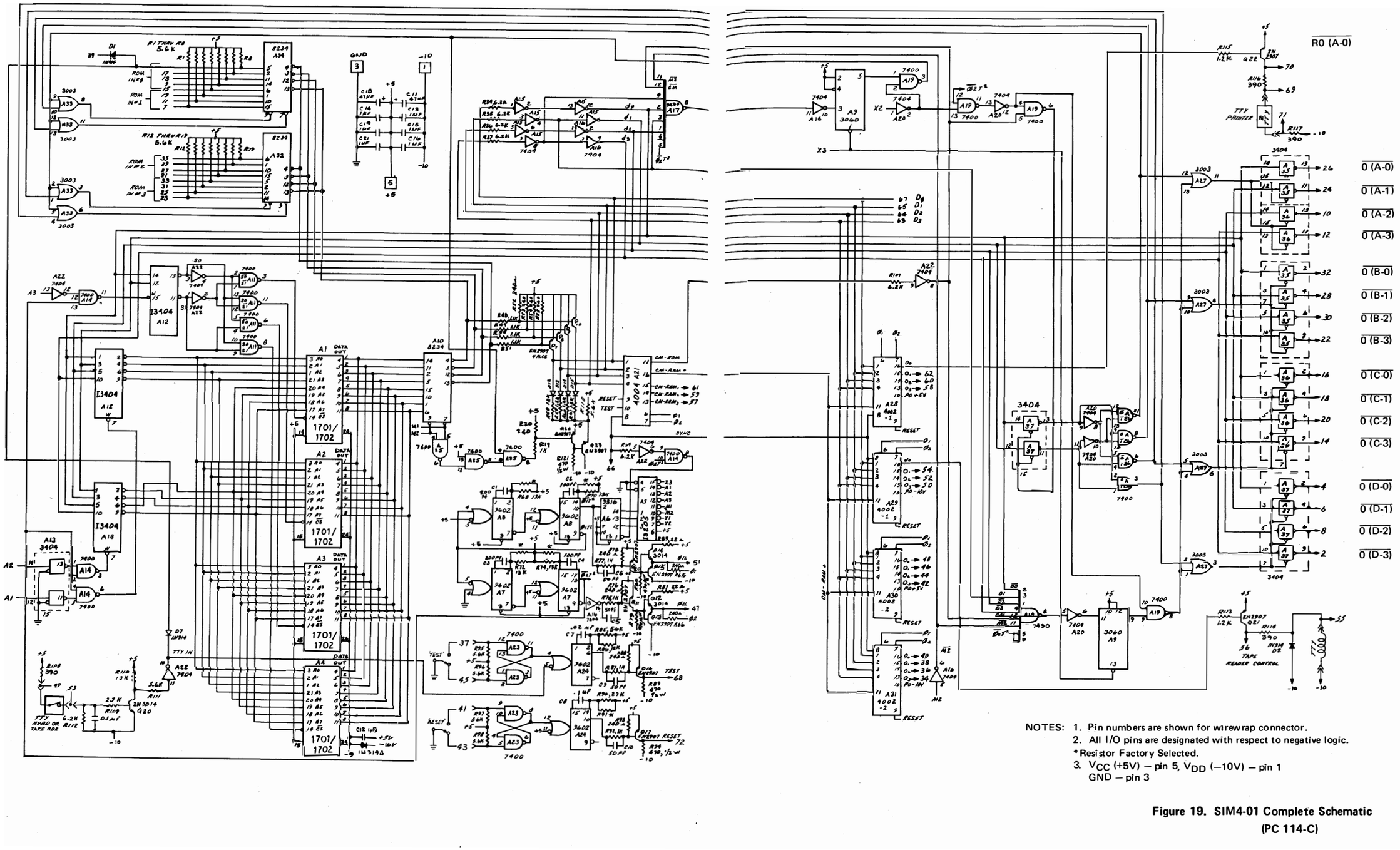
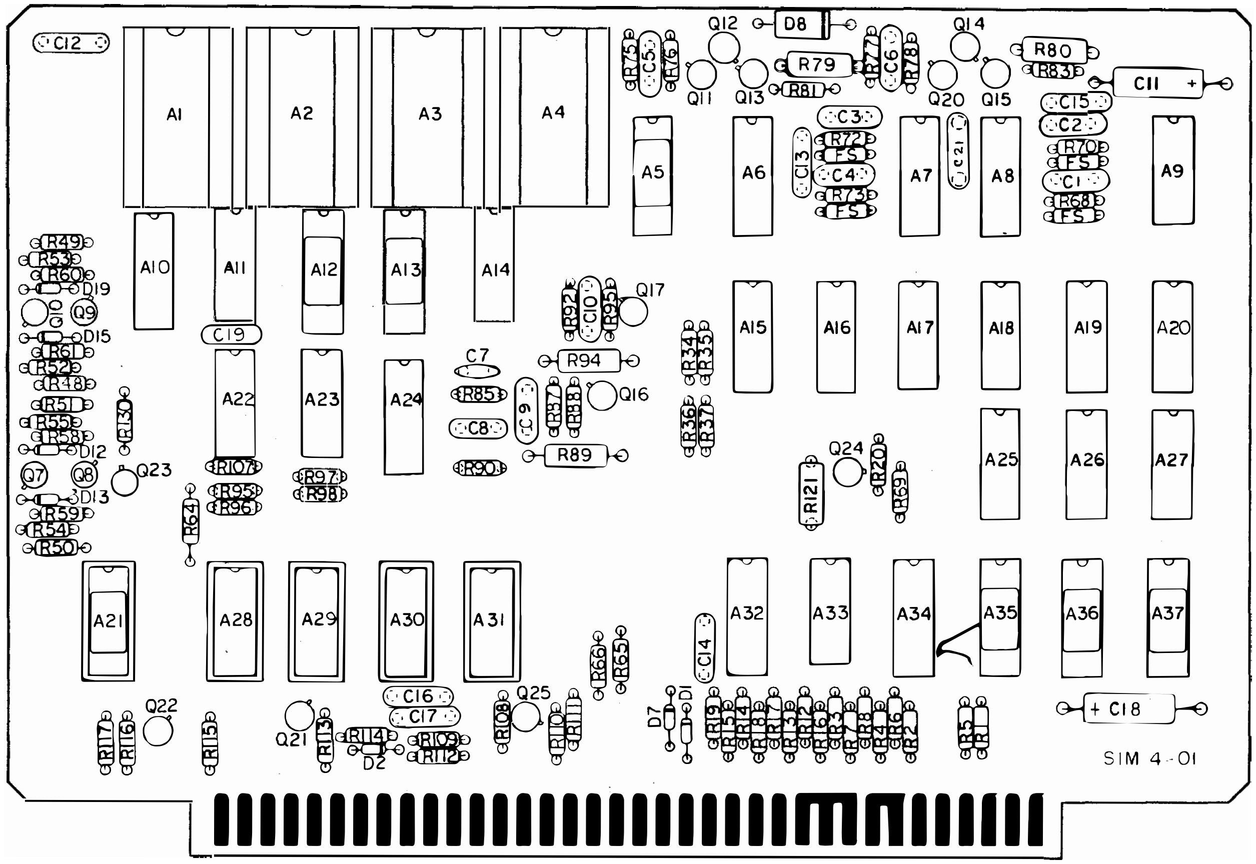


Figure 18. SIM4-01 Clock Generator, Test Signal, Reset Generator and Teletype Interface Detail Drawings



NOTES: 1. Pin numbers are shown for wirewrap connector.
2. All I/O pins are designated with respect to negative logic.
*Resistor Factory Selected.
3. V_{CC} (+5V) — pin 5, V_{DD} (−10V) — pin 1
GND — pin 3

Figure 19. SIM4-01 Complete Schematic (PC 114-C)



Solder Connector P/N 225-23621-101	R̄	P̄	N̄	M̄	L̄	K̄	J̄	H̄	F̄	Ē	D̄	C̄	B̄	Ā	Z	Y	X	W	V	U	T	S	R	P	N	M	L	K	J	H	F	E	D	C	B	A	Amphenol
WirewrapConnector P/N 261-15636-2	71	69	67	65	63	61	59	57	55	53	51	49	47	45	43	41	39	37	35	33	31	29	27	25	23	21	19	17	15	13	11	9	7	5	3	1	
WirewrapConnector P/N VPB01E36E00A1	72	70	68	66	64	62	60	58	58	54	52	50	48	46	44	42	40	38	36	34	32	30	28	26	24	22	20	18	16	14	12	10	8	6	4	2	CDC

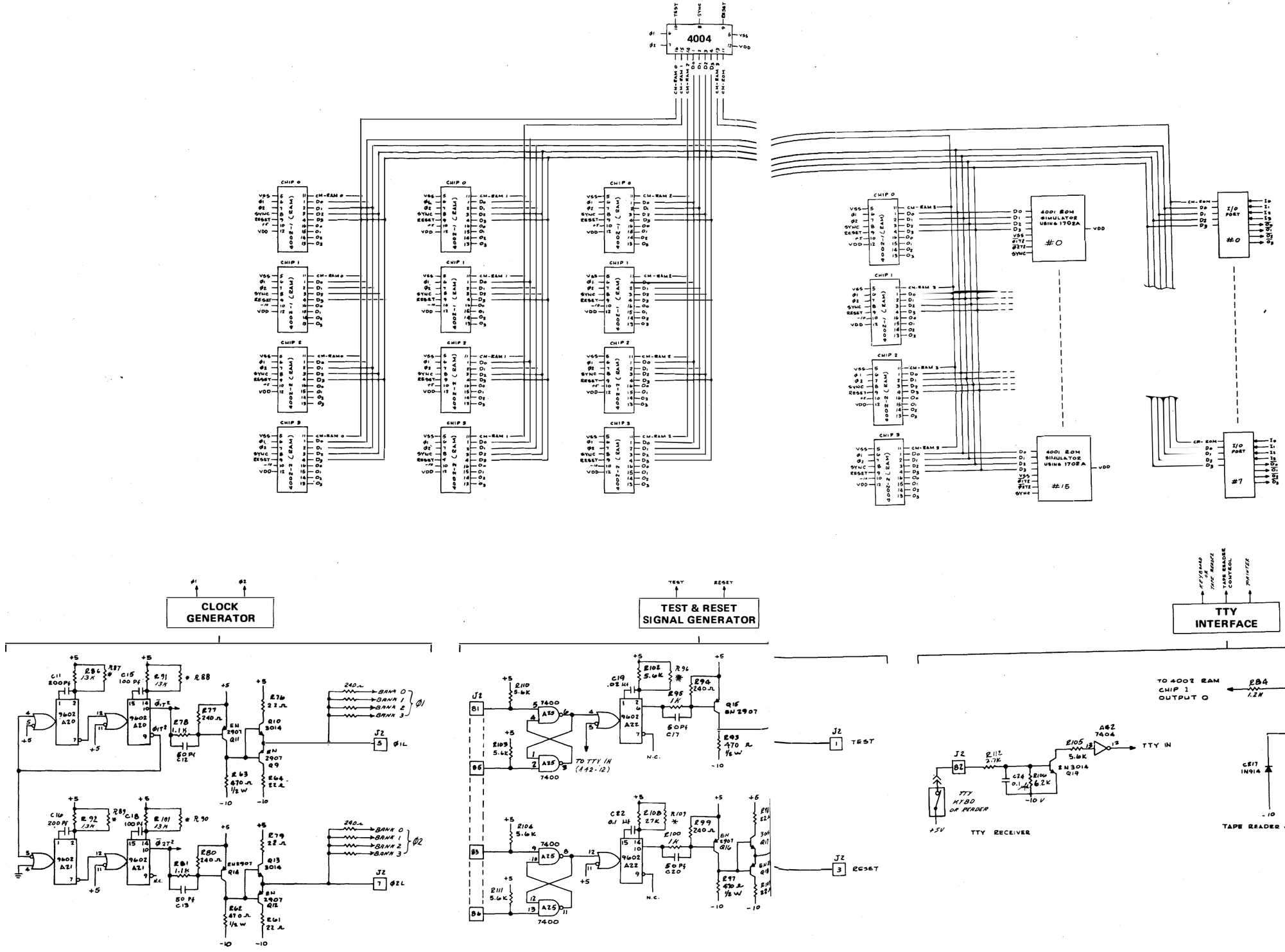
Figure 20. Component Side of SIM4-01 Board

Solder Connector P/N 225-23621-101	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	Amphenol
Wirewrap Connector P/N 261-15636-2	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34	36	38	40	42	44	46	48	50	52	54	56	58	60	62	64	66	68	70	72	
Wirewrap Connector P/N VPB01E36E00A1	1	3	5	7	9	11	13	15	17	19	21	23	25	27	29	31	33	35	37	39	41	43	45	47	49	51	53	55	57	59	61	63	65	67	69	71	CDC

Figure 21. Pin Definition – Reverse Side of SIM4-01 Board

(Pin numbers are shown for wirewrap connector —
All inputs and outputs are designated with respect to negative logic)

PIN NO.	SYMBOL	DESCRIPTION	
1	-10V	-10VDC POWER SUPPLY - V_{DD}	
3	GND	0V - TTL GROUND	
5	+5V	+5VDC POWER SUPPLY - V_{CC} AND V_{SS}	
37	TS	TEST SWITCH CONTROL (NORMALLY OPEN)	
45	TR	TEST SWITCH CONTROL (NORMALLY CLOSED)	
41	RS_1	RESET SWITCH CONTROL (NORMALLY OPEN)	
43	RS_2	RESET SWITCH CONTROL (NORMALLY CLOSED)	
49	TTY(R1)	TELETYPE KEYBOARD or TAPE READER CONNECTION 1	
53	TTY(R2)	TELETYPE KEYBOARD or TAPE READER CONNECTION 2	
55	TTY(T1)	TELETYPE TAPE READER CONTROL	
69	TTY(X1)	TELETYPE PRINTER CONNECTION 1	
71	TTY(X2)	TELETYPE PRINTER CONNECTION 2	
51	$\emptyset_{1L}$	PHASE 1 CLOCK	
47	$\emptyset_{2L}$	PHASE 2 CLOCK	
67	D_0	DATA BUS 0	MOS COMPATIBLE OUT
65	D_1	DATA BUS 1	
64	D_2	DATA BUS 2	
63	D_3	DATA BUS 3	
66	SYNC	MACHINE CYCLE SYCHRONIZATION SIGNAL	
61	$CM\text{-}RAM_1$	RAM BANK 1 COMMAND LINE	
59	$CM\text{-}RAM_2$	RAM BANK 2 COMMAND LINE	
57	$CM\text{-}RAM_3$	RAM BANK 3 COMMAND LINE	
68	TEST	TEST SIGNAL USED IN CONJUNCTION WITH JCN INSTR.	
72	RESET	RESET SIGNAL USED TO CLEAR THE SYSTEM	
39	I (A-0)	ROM INPUT PORT, ROM 0	TTL COMPATIBLE IN
17	I (A-1)	ROM INPUT PORT, ROM 0	
13	I (A-2)	ROM INPUT PORT, ROM 0	
9	I (A-3)	ROM INPUT PORT, ROM 0	
15	I (B-0)	ROM INPUT PORT, ROM 1	
19	I (B-1)	ROM INPUT PORT, ROM 1	
11	I (B-2)	ROM INPUT PORT, ROM 1	
7	I (B-3)	ROM INPUT PORT, ROM 1	
35	I (C-0)	ROM INPUT PORT, ROM 2	
29	I (C-1)	ROM INPUT PORT, ROM 2	
27	I (C-2)	ROM INPUT PORT, ROM 2	
21	I (C-3)	ROM INPUT PORT, ROM 2	
33	I (D-0)	ROM INPUT PORT, ROM 3	
31	I (D-1)	ROM INPUT PORT, ROM 3	
25	I (D-2)	ROM INPUT PORT, ROM 3	
23	I (D-3)	ROM INPUT PORT, ROM 3	
26	$\overline{O\ (A\text{-}0)}$	ROM OUTPUT PORT, ROM 0	TTL COMPATIBLE OUT
24	$\overline{O\ (A\text{-}1)}$	ROM OUTPUT PORT, ROM 0	
10	$\overline{O\ (A\text{-}2)}$	ROM OUTPUT PORT, ROM 0	
12	$\overline{O\ (A\text{-}3)}$	ROM OUTPUT PORT, ROM 0	
32	$\overline{O\ (B\text{-}0)}$	ROM OUTPUT PORT, ROM 1	
28	$\overline{O\ (B\text{-}1)}$	ROM OUTPUT PORT, ROM 1	
30	$\overline{O\ (B\text{-}2)}$	ROM OUTPUT PORT, ROM 1	
22	$\overline{O\ (B\text{-}3)}$	ROM OUTPUT PORT, ROM 1	
16	$\overline{O\ (C\text{-}0)}$	ROM OUTPUT PORT, ROM 2	
18	$\overline{O\ (C\text{-}1)}$	ROM OUTPUT PORT, ROM 2	
20	$\overline{O\ (C\text{-}2)}$	ROM OUTPUT PORT, ROM 2	
14	$\overline{O\ (C\text{-}3)}$	ROM OUTPUT PORT, ROM 2	
4	$\overline{O\ (D\text{-}0)}$	ROM OUTPUT PORT, ROM 3	
6	$\overline{O\ (D\text{-}1)}$	ROM OUTPUT PORT, ROM 3	
8	$\overline{O\ (D\text{-}2)}$	ROM OUTPUT PORT, ROM 3	
2	$\overline{O\ (D\text{-}3)}$	ROM OUTPUT PORT, ROM 3	
70	$\overline{RO(A\text{-}0)}$	RAM OUTPUT PORT, RAM 0	TRANSISTOR BUFFER OUT
62	RO(A-1)	RAM OUTPUT PORT, RAM 0	MOS COMPATIBLE OUT
60	RO(A-2)	RAM OUTPUT PORT, RAM 0	
58	RO(A-3)	RAM OUTPUT PORT. RAM 0	
56	$\overline{RO(B\text{-}0)}$	RAM OUTPUT PORT, RAM 1	TRANSISTOR BUFFER OUT
54	RO(B-1)	RAM OUTPUT PORT, RAM 1	MOS COMPATIBLE OUT
52	RO(B-2)	RAM OUTPUT PORT, RAM 1	
50	RO(B-3)	RAM OUTPUT PORT, RAM 1	
48	RO(C-0)	RAM OUTPUT PORT, RAM 2	
46	RO(C-1)	RAM OUTPUT PORT, RAM 2	
44	RO(C-2)	RAM OUTPUT PORT, RAM 2	
42	RO(C-3)	RAM OUTPUT PORT, RAM 2	
40	RO(D-0)	RAM OUTPUT PORT, RAM 3	
38	RO(D-1)	RAM OUTPUT PORT, RAM 3	
36	RO(D-2)	RAM OUTPUT PORT, RAM 3	
34	RO(D-3)	RAM OUTPUT PORT, RAM 3	


4004
TEST
SYNC
RESET
CM-RAM 0
CM-RAM 1
CM-RAM 2
CM-RAM 3
CM-ROM
CHIP 0
CHIP 1
CHIP 2
CHIP 3
4002-1 (RAM)
4002-2 (RAM)
4001: ROM SIMULATOR USING 1702A
#0
#15
I/O PORT
#0
#7
CLOCK GENERATOR
TEST & RESET SIGNAL GENERATOR
TTY INTERFACE
BANK 0
BANK 1
BANK 2
BANK 3
TEST
RESET
TO TTY IN (A12-12)
TTY IN
TTY KYBD OR READER
TTY RECEIVER
TAPE READER CONTROL
TTY TRANSMITTER
TTY
PRINTER
TO 4002 RAM CHIP 1 OUTPUT 0
TO 4002 RAM CHIP 0 OUTPUT 0
FACTORY SELECTED
N.C.

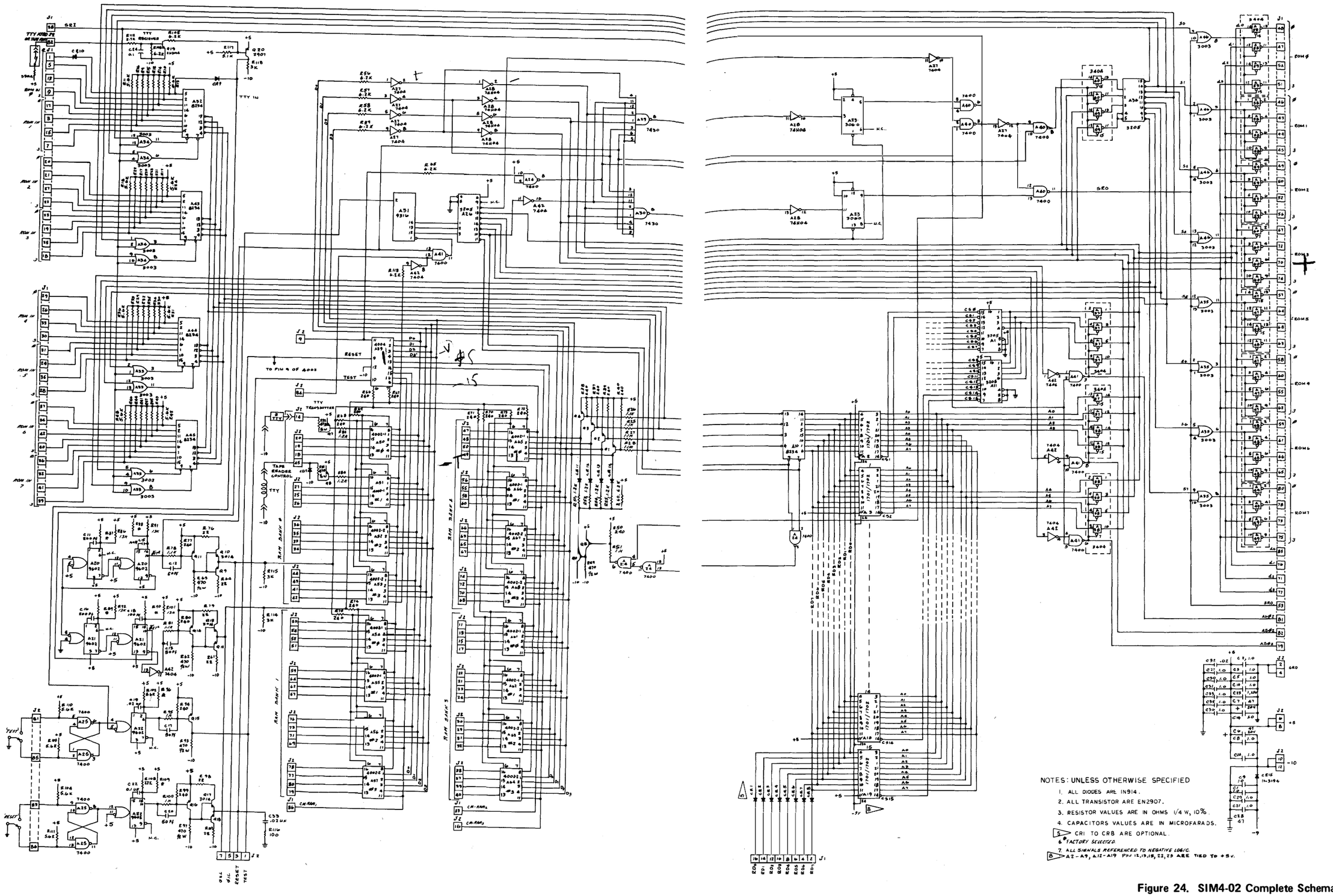
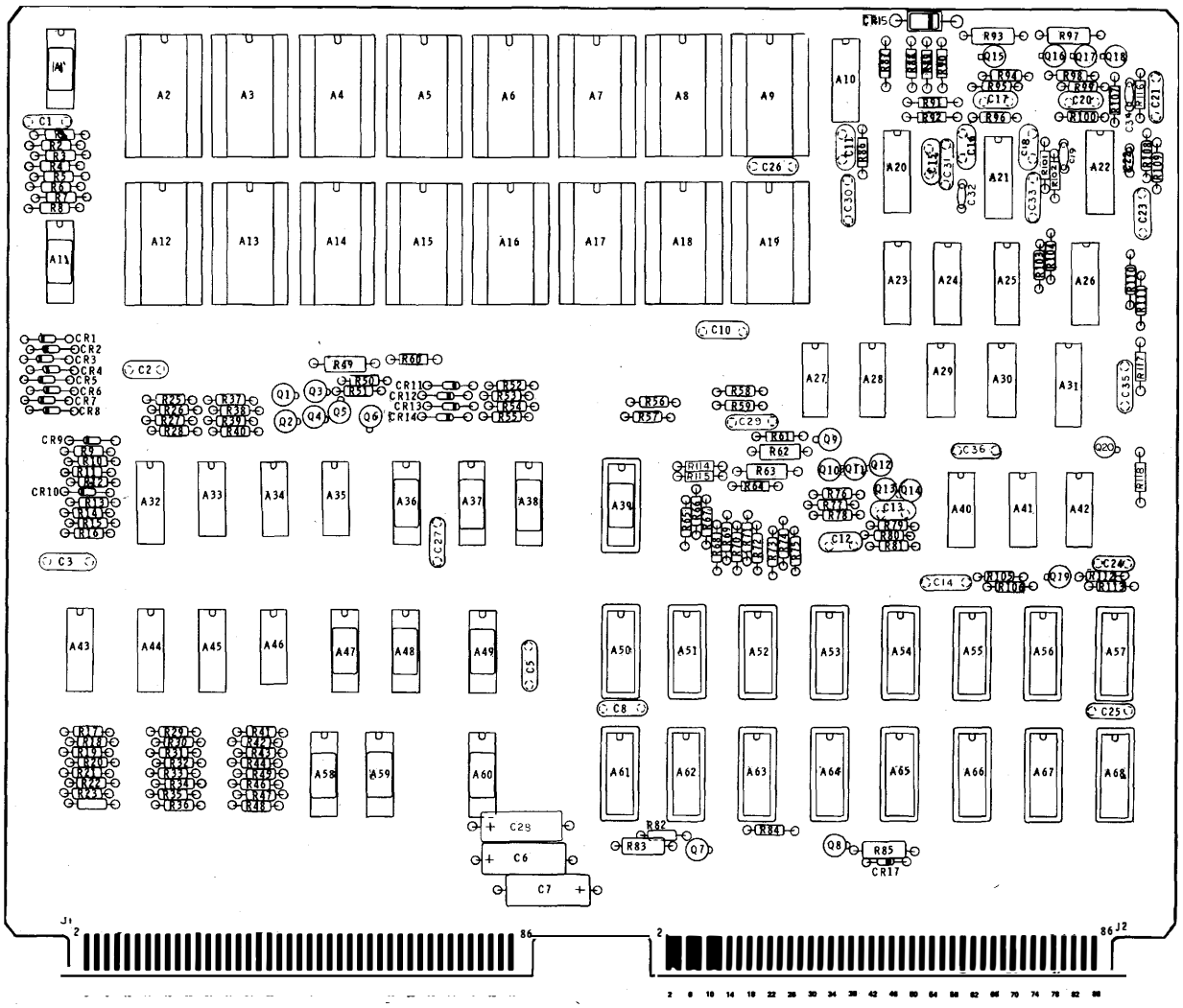


Figure 24. SIM4-02 Complete Schematic



NOTES: Unless otherwise specified.

1. All diodes are 1N914.
2. All transistors are EN2907
3. Resistor values are in ohms 1/4 W, 10%
4. Capacitor values are in microfarads
5. A29 and A30 was 3015
6. R1-R8 option for additional memory only

*Factory Selected

Amphenol Connector
PN 261-10043-2

Figure 25. Component Side of SIM4-02 Board

Figure 26. Pin Definition – Reverse Side of SIM4-02 Board

SIM4-02 Board Pin Description

(Inputs and outputs designated with respect to negative logic)

PIN NO.	CONNECTOR	SYMBOL	DESCRIPTION
6 & 8	J2	+5V	+5VDC Power Supply
10 & 12	J2	-10V	-10VDC Power Supply
2 & 4	J2	GND	Ground
2 & 4	J1	SPARE	Not Used
1	J1	I (A-0)	ROM INPUT PORT, ROM Ø
5	J1	I (A-1)	ROM INPUT PORT, ROM Ø
13	J1	I (A-2)	ROM INPUT PORT, ROM Ø
9	J1	I (A-3)	ROM INPUT PORT, ROM Ø
17	J1	I (B-0)	ROM INPUT PORT, ROM 1
3	J1	I (B-1)	ROM INPUT PORT, ROM 1
15	J1	I (B-2)	ROM INPUT PORT, ROM 1
7	J1	I (B-3)	ROM INPUT PORT, ROM 1
20	J1	I (C-0)	ROM INPUT PORT, ROM 2
21	J1	I (C-1)	ROM INPUT PORT, ROM 2
27	J1	I (C-2)	ROM INPUT PORT, ROM 2
22	J1	I (C-3)	ROM INPUT PORT, ROM 2
23	J1	I (D-0)	ROM INPUT PORT, ROM 3
19	J1	I (D-1)	ROM INPUT PORT, ROM 3
25	J1	I (D-2)	ROM INPUT PORT, ROM 3
18	J1	I (D-3)	ROM INPUT PORT, ROM 3
29	J1	I (E-0)	ROM INPUT PORT, ROM 4
26	J1	I (E-1)	ROM INPUT PORT, ROM 4
33	J1	I (E-2)	ROM INPUT PORT, ROM 4
30	J1	I (E-3)	ROM INPUT PORT, ROM 4
31	J1	I (F-0)	ROM INPUT PORT, ROM 5
24	J1	I (F-1)	ROM INPUT PORT, ROM 5
35	J1	I (F-2)	ROM INPUT PORT, ROM 5
28	J1	I (F-3)	ROM INPUT PORT, ROM 5
37	J1	I (G-0)	ROM INPUT PORT, ROM 6
34	J1	I (G-1)	ROM INPUT PORT, ROM 6
42	J1	I (G-2)	ROM INPUT PORT, ROM 6
40	J1	I (G-3)	ROM INPUT PORT, ROM 6
36	J1	I (H-0)	ROM INPUT PORT, ROM 7
32	J1	I (H-1)	ROM INPUT PORT, ROM 7
41	J1	I (H-2)	ROM INPUT PORT, ROM 7
39	J1	I (H-3)	ROM INPUT PORT, ROM 7
82	J2	TTY (R)	TELETYPE RECEIVER CONNECTION
14	J2	TTY (X)	TELETYPE TRANSMITTER CONNECTION
45	J2	TTY (T)	TAPE READER CONTROL
48	J1	$\overline{O\ (A-0)}$	ROM OUTPUT PORT, ROM Ø
47	J1	$\overline{O\ (A-1)}$	ROM OUTPUT PORT, ROM Ø
54	J1	$\overline{O\ (A-2)}$	ROM OUTPUT PORT, ROM Ø
51	J1	$\overline{O\ (A-3)}$	ROM OUTPUT PORT, ROM Ø
46	J1	$\overline{O\ (B-0)}$	ROM OUTPUT PORT, ROM 1
43	J1	$\overline{O\ (B-1)}$	ROM OUTPUT PORT, ROM 1
44	J1	$\overline{O\ (B-2)}$	ROM OUTPUT PORT, ROM 1
45	J1	$\overline{O\ (B-3)}$	ROM OUTPUT PORT, ROM 1
49	J1	$\overline{O\ (C-0)}$	ROM OUTPUT PORT, ROM 2
50	J1	$\overline{O\ (C-1)}$	ROM OUTPUT PORT, ROM 2
52	J1	$\overline{O\ (C-2)}$	ROM OUTPUT PORT, ROM 2
56	J1	$\overline{O\ (C-3)}$	ROM OUTPUT PORT, ROM 2

TTL COMPATIBLE INPUTS

TTL COMPATIBLE

PIN NO.	CONNECTOR	SYMBOL	DESCRIPTION
67	J1	$\overline{O\ (D-0)}$	ROM OUTPUT PORT, ROM 3
72	J1	$\overline{O\ (D-1)}$	ROM OUTPUT PORT, ROM 3
70	J1	$\overline{O\ (D-2)}$	ROM OUTPUT PORT, ROM 3
74	J1	$\overline{O\ (D-3)}$	ROM OUTPUT PORT, ROM 3
57	J1	$\overline{O\ (E-0)}$	ROM OUTPUT PORT, ROM 5
64	J1	$\overline{O\ (E-1)}$	ROM OUTPUT PORT, ROM 5
68	J1	$\overline{O\ (E-2)}$	ROM OUTPUT PORT, ROM 5
63	J1	$\overline{O\ (E-3)}$	ROM OUTPUT PORT, ROM 5
58	J1	$\overline{O\ (F-0)}$	ROM OUTPUT PORT, ROM 4
60	J1	$\overline{O\ (F-1)}$	ROM OUTPUT PORT, ROM 4
55	J1	$\overline{O\ (F-2)}$	ROM OUTPUT PORT, ROM 4
62	J1	$\overline{O\ (F-3)}$	ROM OUTPUT PORT, ROM 4
59	J1	$\overline{O\ (G-0)}$	ROM OUTPUT PORT, ROM 6
61	J1	$\overline{O\ (G-1)}$	ROM OUTPUT PORT, ROM 6
66	J1	$\overline{O\ (G-2)}$	ROM OUTPUT PORT, ROM 6
65	J1	$\overline{O\ (G-3)}$	ROM OUTPUT PORT, ROM 6
80	J1	$\overline{O\ (H-0)}$	ROM OUTPUT PORT, ROM 7
78	J1	$\overline{O\ (H-1)}$	ROM OUTPUT PORT, ROM 7
73	J1	$\overline{O\ (H-2)}$	ROM OUTPUT PORT, ROM 7
75	J1	$\overline{O\ (H-3)}$	ROM OUTPUT PORT, ROM 7
14	J2	$\overline{RO\ (A-0)}$	RAM OUTPUT PORT BANK 0, RAM 0 AND TTY TRANSMITTER
20	J2	RO (A-1)	RAM OUTPUT PORT BANK 0, RAM 0
19	J2	RO (A-2)	RAM OUTPUT PORT, BANK 0, RAM 0
18	J2	RO (A-3)	RAM OUTPUT PORT, BANK 0, RAM 0
45	J2	$\overline{RO\ (A-4)}$	RAM OUTPUT PORT, BANK 0, RAM 1 AND TAPE READER
27	J2	RO (A-5)	RAM OUTPUT PORT, BANK 0, RAM 1
25	J2	RO (A-6)	RAM OUTPUT PORT, BANK 0, RAM 1
26	J2	RO (A-7)	RAM OUTPUT PORT, BANK 0, RAM 1
36	J2	RO (A-8)	RAM OUTPUT PORT, BANK 0, RAM 2
35	J2	RO (A-9)	RAM OUTPUT PORT, BANK 0, RAM 2
33	J2	RO (A-10)	RAM OUTPUT PORT, BANK 0, RAM 2
34	J2	RO (A-11)	RAM OUTPUT PORT, BANK 0, RAM 2
44	J2	RO (A-12)	RAM OUTPUT PORT, BANK 0, RAM 3
43	J2	RO (A-13)	RAM OUTPUT PORT, BANK 0, RAM 3
41	J2	RO (A-14)	RAM OUTPUT PORT, BANK 0, RAM 3
42	J2	RO (A-15)	RAM OUTPUT PORT, BANK 0, RAM 3
53	J2	R1 (A-0)	RAM OUTPUT PORT, BANK 1, RAM 0
54	J2	R1 (A-1)	RAM OUTPUT PORT, BANK 1, RAM 0
52	J2	R1 (A-2)	RAM OUTPUT PORT, BANK 1, RAM 0
51	J2	R1 (A-3)	RAM OUTPUT PORT, BANK 1, RAM 0
59	J2	R1 (A-4)	RAM OUTPUT PORT, BANK 1, RAM 1
64	J2	R1 (A-5)	RAM OUTPUT PORT, BANK 1, RAM 1
62	J2	R1 (A-6)	RAM OUTPUT PORT, BANK 1, RAM 1
57	J2	R1 (A-7)	RAM OUTPUT PORT, BANK 1, RAM 1
76	J2	R1 (A-8)	RAM OUTPUT PORT, BANK 1, RAM 2
73	J2	R1 (A-9)	RAM OUTPUT PORT, BANK 1, RAM 2
71	J2	R1 (A-10)	RAM OUTPUT PORT, BANK 1, RAM 2
69	J2	R1 (A-11)	RAM OUTPUT PORT, BANK 1, RAM 2
78	J2	R1 (A-12)	RAM OUTPUT PORT, BANK 1, RAM 3
77	J2	R1 (A-13)	RAM OUTPUT PORT, BANK 1, RAM 3
80	J2	R1 (A-14)	RAM OUTPUT PORT, BANK 1, RAM 3

TTL COMPATIBLE

BUFFER OUT MOS BUFFER OUT

MOS COMPATIBLE

PIN NO.	CONNECTOR	SYMBOL	DESCRIPTION	
79	J2	R1 (A-15)	RAM OUTPUT PORT, BANK 1, RAM 3	MOS COMPATIBLE
47	J2	R2 (A-0)	RAM OUTPUT PORT, BANK 2, RAM 0	
48	J2	R2 (A-1)	RAM OUTPUT PORT, BANK 2, RAM 0	
50	J2	R2 (A-2)	RAM OUTPUT PORT, BANK 2, RAM 0	
49	J2	R2 (A-3)	RAM OUTPUT PORT, BANK 2, RAM 0	
56	J2	R2 (A-4)	RAM OUTPUT PORT, BANK 2, RAM 1	
55	J2	R2 (A-5)	RAM OUTPUT PORT, BANK 2, RAM 1	
58	J2	R2 (A-6)	RAM OUTPUT PORT, BANK 2, RAM 1	
60	J2	R2 (A-7)	RAM OUTPUT PORT, BANK 2, RAM 1	
66	J2	R2 (A-8)	RAM OUTPUT PORT, BANK 2, RAM 2	
63	J2	R2 (A-9)	RAM OUTPUT PORT, BANK 2, RAM 2	
65	J2	R2 (A-10)	RAM OUTPUT PORT, BANK 2, RAM 2	
67	J2	R2 (A-11)	RAM OUTPUT PORT, BANK 2, RAM 2	
74	J2	R2 (A-12)	RAM OUTPUT PORT, BANK 2, RAM 3	
72	J2	R2 (A-13)	RAM OUTPUT PORT, BANK 2, RAM 3	
70	J2	R2 (A-14)	RAM OUTPUT PORT, BANK 2, RAM 3	
68	J2	R2 (A-15)	RAM OUTPUT PORT, BANK 2, RAM 3	
11	J2	R3 (A-0)	RAM OUTPUT PORT, BANK 3, RAM 0	
13	J2	R3 (A-1)	RAM OUTPUT PORT, BANK 3, RAM 0	
15	J2	R3 (A-2)	RAM OUTPUT PORT, BANK 3, RAM 0	
17	J2	R3 (A-3)	RAM OUTPUT PORT, BANK 3, RAM 0	
22	J2	R3 (A-4)	RAM OUTPUT PORT, BANK 3, RAM 1	
21	J2	R3 (A-5)	RAM OUTPUT PORT, BANK 3, RAM 1	
23	J2	R3 (A-6)	RAM OUTPUT PORT, BANK 3, RAM 1	
24	J2	R3 (A-7)	RAM OUTPUT PORT, BANK 3, RAM 1	
30	J2	R3 (A-8)	RAM OUTPUT PORT, BANK 3, RAM 2	
29	J2	R3 (A-9)	RAM OUTPUT PORT, BANK 3, RAM 2	
31	J2	R3 (A-10)	RAM OUTPUT PORT, BANK 3, RAM 2	

PIN NO.	CONNECTOR	SYMBOL	DESCRIPTION	
32	J2	R3 (A-11)	RAM OUTPUT PORT, BANK 3, RAM 2	MOS COMPATIBLE
38	J2	R3 (A-12)	RAM OUTPUT PORT, BANK 3, RAM 3	
37	J2	R3 (A-13)	RAM OUTPUT PORT, BANK 3, RAM 3	
39	J2	R3 (A-14)	RAM OUTPUT PORT, BANK 3, RAM 3	
40	J2	R3 (A-15)	RAM OUTPUT PORT, BANK 3, RAM 3	
86	J1	CM-RAM$_1$	RAM BANK 1 COMMAND LINE	
83	J1	CM-RAM$_2$	RAM BANK 2 COMMAND LINE	
16	J2	CM-RAM$_3$	RAM BANK 3 COMMAND LINE	
84	J2	SYNC(TTL)	MACHINE CYCLE SYNCHRONIZATION SIG. (TTL LEVELS)	
9	J2	CM-ROM	MOS (EXPANSION)	
81	J2	TS$_1$	TEST SWITCH CONTROL (NORMALLY OPEN)	
85	J2	TS$_2$	TEST SWITCH CONTROL (NORMALLY CLOSED)	
83	J2	RS$_1$	RESET SWITCH CONTROL (NORMALLY OPEN)	
86	J2	RS$_2$	RESET SWITCH CONTROL (NORMALLY CLOSED)	
5	J2	Ø1L MOS	PHASE 1 CLOCK	
7	J2	Ø2L MOS	PHASE 2 CLOCK	
3	J2	RESET	RESET SIGNAL FOR 4004 (EXPANSION)	
1	J2	TEST	TEST SIGNAL FOR 4004 (EXPANSION)	
38	J1	SRI	ROM INPUT PORT STROBE (EXPANSION)	
53	J1	SRO	ROM OUTPUT PORT STROBE (EXPANSION)	
16	J1	RD$_0$	ROM DATA OUTPUT 0 (EXPANSION)	
14	J1	RD$_1$	ROM DATA OUTPUT 1 (EXPANSION)	
12	J1	RD$_2$	ROM DATA OUTPUT 2 (EXPANSION)	
10	J1	RD$_3$	ROM DATA OUTPUT 3 (EXPANSION)	
8	J1	RD$_4$	ROM DATA OUTPUT 4 (EXPANSION)	
6	J1	RD$_5$	ROM DATA OUTPUT 5 (EXPANSION)	
4	J1	RD$_6$	ROM DATA OUTPUT 6 (EXPANSION)	
2	J1	RD$_7$	ROM DATA OUTPUT 7 (EXPANSION)	
79	J1	A$_2$ Ø$_2$	ROM MIDDLE ADDRESS SELECT EXPANSION	
82	J1	A$_1$ Ø$_2$	ROM LOWER ADDRESS SELECT EXPANSION	
81	J1	A$_3$ Ø$_2$	ROM UPPER ADDRESS SELECT EXPANSION	
85	J1	d$_0$	DATA BUS 0 (TTL COMPATIBLE)	
76	J1	d$_1$	DATA BUS 1 (TTL COMPATIBLE)	
71	J1	d$_2$	DATA BUS 2 (TTL COMPATIBLE)	
77	J1	d$_3$	DATA BUS 3 (TTL COMPATIBLE)	

XII. SAMPLE SIXTEEN DIGIT DECIMAL ADDITION PROGRAM WITH TTY KEYBOARD and PRINTER INTERFACE (Intel ROM Program Number A0700)

An MCS-4 program has been developed to demonstrate both the control and arithmetic features of this microcomputer system. This program adds a sixteen digit integer to the content of the accumulator and prints the new content of the accumulator. The programmed ROM may be used with either the SIM4-01 or SIM4-02 prototyping system. Input/output capability is provided by an ASR 33 teletype. The prototyping system and the teletype should be connected as shown in Section XIII. The TTY, keyboard interrogation, arithmetic operation, and TTY printer output are all controlled by the CPU (4004) using this special decimal addition program.

To use this program:

1) Insert the ROM in ROM position zero in the prototype system and reset the system, including the accumulator, to zero.

2) Enter from one to sixteen integers from the TTY keyboard.

3) If fewer than sixteen digits are entered, depress the (+) key.

4) The number entered is added to the content of the accumulator.

5) This procedure may be repeated until the content of the sixteen digit accumulator overflows (X's will be printed). This overflow will automatically reset the system.

6) To reset the system at any other time, use the reset switch.

This program uses both the keyboard process routine and decimal addition routine explained earlier in this manual. In addition, the TTY printing subroutine is used.

Example of Addition Program

Printout On Teletype

```
                                  Reset System
1234   (+)  ----------------------Enter number and add command
1234        ----------------------Printed result
22     (+)  ----------------------Enter number and add command
1256        ----------------------Printed result
1111111111111111 ----------------Enter sixteen digits
1111111111112367 ----------------Printed result
9999999999999999 ----------------Enter sixteen digits
XXXXXXXXXXXXXXXX ----------------Resulting overflow
                                  System automatically reset
```

The listing of tape A0700 follows.

SIXTEEN DIGIT INTEGER ADDITION MICROPROGRAM WITH TTY KEYBOARD AND PRINTER INTERFACE

```
                / PROPERTY OF INTEL CORP SANTA CLARA CALIFORNIA
                /
                / PROGRAM APE-C020-4 I4004 16-DIGIT ADDING-
                / MACHINE W/TTY-KBD DRIVER
                /
                /
                / PROGRAMMER PHIL TAI ;APPLICATIONS ENGINEERING
                / DATE OCTOBER 27,1971 @ 0830
                / PALL-IV ASSEMBLER
                /
                *0000
                DECIMAL
0000 0337 BEGIN,  LDM 15          / SET RAM (0) PORT TO 1111;8
0001 0040
0002 0000         FIM 0<;0
0003 0041         SRC 0<
0004 0341         WMP
0005 0320         LDM 0
0006 0044
0007 0000         FIM 2<;0        /IR(4-5)=0
0010 0046
0011 0012         FIM 3<;10       /IR(6)=0;IR(7)=10
0012 0045 REP,    SRC 2<
0013 0342         WRR
0014 0144         INC 4
0015 0167
0016 0012         ISZ 7;REP
0017 0320 NEXT,   LDM 0           / SET DIGIT CNTR
0020 0275         XCH 13          / IR(13)=0
0021 0044
0022 0060         FIM 2<;48       / IR(4)=3;IR(5)=0
0023 0120
0024 0260         JMS ;CLRRAM     / CLEAR RAM
0025 0120
0026 0367         JMS ;CRLF       / POSITION CARRIAGE
0027 0361         CLC
                /
                / TEST TTY/KBD INPUTS
                /
0030 0021
0031 0030 ST,     JCN TZ;ST
0032 0120
0033 0271         JMS;SBR1
0034 0040
0035 0015         FIM 0<;13       / IR(0)=0;IR(1)=13
0036 0161
0037 0036 TEST,   ISZ 1;TEST
0040 0041         SRC 0<
0041 0352         RDR
0042 0364         CMA
0043 0341         WMP
0044 0120
0045 0300         JMS;SBR2
0046 0040
0047 0000         FIM 0<;0        / IR(0-1)=0
0050 0320         LDM 0
0051 0262         XCH 2           / IR(2)=0
0052 0320         LDM 0
0053 0263         XCH 3           / IR(3)=0
0054 0330         LDM 8
0055 0264         XCH 4           / IR(4)-8
0056 0120
0057 0271 ST1,    JMS ;SBR1
0060 0361         CLC
0061 0041         SRC 0<
0062 0352         RDR             / READ DATA INPUT
0063 0364         CMA
0064 0341         WMP
0065 0366         RAR             / STORE DATA IN CARRY
0066 0242         LD 2            / LOAD AC=IR(2)
0067 0366         RAR             / TRANSFERE BIT
0070 0262         XCH 2           / RESTORE NEW DATA WORD
0071 0243         LD 3
0072 0366         RAR
0073 0263         XCH 3           / EXTEND REGISTER TO MAKE 8 BITS
0074 0120
0075 0300         JMS ; SBR2
0076 0164
0077 0056         ISZ 4;ST1
0100 0337         LDM 15
```

```
0101 0040
0102 0000          FIM 0<;0
0103 0041          SRC 0<
0104 0341          WMP
0105 0333 COMADD,  LDM 11
0106 0223          SUB 3
0107 0361          CLC
0110 0024
0111 0154          JCN AZ;ADDITN    / IF AC=0; JUMP
                   /
0112 0050
0113 0100 WRITE,   FIM 4<;64        / IR(8)=4;IR(9)=0
0114 0243          LD 3
0115 0051          SRC 4<
0116 0342          WRR
0117 0150          INC 8
0120 0242          LD 2
0121 0051          SRC 4<
0122 0342          WRR
                   /
0123 0040
0124 0001 STORE,   FIM 0<;1         / IR(0)=0;IR(1)=1
0125 0044
0126 0077          FIM 2<;63        / IR(4)=3;IR(5)=15
0127 0046
0130 0076          FIM 3<;62        / IR(6)=3;IR(7)=14
0131 0047 REP1,    SRC 3<
0132 0351          RDM
0133 0045          SRC 2<
0134 0340          WRM
0135 0245          LD 5
0136 0370          DAC
0137 0265          XCH 5
0140 0361          CLC
0141 0247          LD 7
0142 0370          DAC
0143 0267          XCH 7
0144 0361          CLC
0145 0161
0146 0131          ISZ 1;REP1
0147 0243          LD 3
0150 0045          SRC 2<
0151 0340          WRM
0152 0175
0153 0030          ISZ 13;ST
                   /
0154 0120
0155 0367 ADDITN,  JMS ;CRLF        / POSITION CARRIAGE
0156 0040
0157 0000          FIM 0<;0         / IR(0-1)=0
0160 0044
0161 0060          FIM 2<;48        / IR(4)=3;IR(5)=0
0162 0320          LDM 0
0163 0266          XCH 6
0164 0361          CLC
0165 0045 AD1,     SRC 2<
0166 0351          RDM
0167 0041          SRC 0<
0170 0353          ADM
0171 0373          DAA
0172 0340          WRM
0173 0141          INC 1
0174 0145          INC 5
0175 0166
0176 0165          ISZ 6;AD1
                   /
0177 0022
0200 0242 OVERFL,  JCN CN;XXX       / TEST FOR CARRY
0201 0054
0202 0017          FIM 6<;15        / IR(12)-0;IR(13)=15
0203 0320          LDM 0
0204 0272          XCH 10
0205 0044
0206 0000          FIM 2<;0
0207 0321          LDM 1
0210 0045          SRC 2<
0211 0344          WR0
0212 0045 AD2,     SRC 2<
0213 0354          RD0
0214 0366          RAR
0215 0055          SRC 6<
0216 0351          RDM
```

```
0217 0034
0220 0223          JCN AN;SKP1      / TEST FOR AC.NE.0
0221 0022
0222 0233          JCN CN; SKIP     / TEST FOR CA=1
0223 0263 SKP1,    XCH 3
0224 0333          LDM 11
0225 0262          XCH 2
0226 0320          LDM 0
0227 0045          SRC 2<
0230 0344          WR0
0231 0120
0232 0307          JMS ;PRINT
0233 0255 SKIP,    LD 13
0234 0370          DAC
0235 0275          XCH 13
0236 0172
0237 0212          ISZ 10;AD2
0240 0100
0241 0017          JUN ;NEXT
0242 0320 XXX,     LDM 0
0243 0272          XCH 10
0244 0042
0245 0330 OVFL1,   FIM 1<;216       / IR(1)=8;IR(2)=13 [X]
0246 0120
0247 0307          JMS ;PRINT
0250 0172
0251 0244          ISZ 10;OVFL1
0252 0044
0253 0000          FIM 2<;0
0254 0120
0255 0260          JMS ;CLRRAM
0256 0100
0257 0017          JUN ;NEXT
0260 0320 CLRRAM,  LDM 0
0261 0261          XCH 1
0262 0320 CLEAR,   LDM 0
0263 0045          SRC 2<
0264 0340          WRM
0265 0145          INC 5
0266 0161
0267 0262          ISZ 1;CLEAR
0270 0300          BBL
                   /
                   /
                   / SUBROUTINE
                   /
0271 0040
0272 0000 SBR1,    FIM 0<;0         / IR(0-1)=0
                                    /547
0273 0160
0274 0273 L1,      ISZ 0;L1
0275 0161
0276 0273          ISZ 1;L1
0277 0300          BBL
                   /
                   /
0300 0040
0301 0010 SBR2,    FIM 0<;8         / IR(0)=0,IR(1)=8
0302 0160
0303 0302 L2,      ISZ 0;L2         / 275
0304 0161
0305 0302          ISZ 1;L2
0306 0300          BBL
                   /
                   /
                   /
                   /
                   / PRINT ROUTINE
                   /
0307 0040
0310 0020 PRINT,   FIM 0<;16        / IR(0)=1;IR(1)=0
0311 0327          LDM 7
0312 0041          SRC 0<
0313 0340          WRM
0314 0141          INC 1
0315 0263          XCH 3
0316 0041          SRC 0<
0317 0340          WRM
0320 0141          INC 1
0321 0262          XCH 2
0322 0041          SRC 0<
0323 0340          WRM
0324 0050
0325 0020          FIM 4<;16        / IR(8)=1;IR(9)=0
0326 0042
0327 0320          FIM 1<;208       / IR(2)=13,IR(3)=0
```

```
0330 0044
0331 0014 ST7,    FIM 2<;12        / IR(4)=0,IR(5)=12
0332 0051         SRC 4<
0333 0351         RDM
0334 0341 ST8,    WMP
0335 0264         XCH 4            / SAVE C(AC) IN INDEX REG 4
0336 0120
0337 0271         JMS;SBR1         / DELAY ROUTINE #1
0340 0120
0341 0300         JMS;SBR2         / DELAY ROUTINE #2
0342 0264         XCH 4            / RESTORE SAVED C(AC)
0343 0165
0344 0360         ISZ 5;ST9        / NUMBER OF ROTATIONS
0345 0151         INC 9            / NUMBER OF DIGITS
0346 0000         NOP
0347 0000         NOP
0350 0000         NOP
0351 0000         NOP
0352 0000         NOP
0353 0162
0354 0330         ISZ 2;ST7        / NUMBER OF 4-BIT WORDS
0355 0337         LDM 15
0356 0341         WMP
0357 0300         BBL
0360 0046
0361 0014 ST9,    FIM 3<;12        / IR(6)=0;IR(7)=12
0362 0167
0363 0362 ST12,   ISZ 7;ST12
0364 0366         RAR
0365 0100
0366 0334         JUN 0;ST8
                  /
                  / CR/LF ROUTINES
                  /
0367 0042
0370 0215 CRLF,   FIM 1<; 141      / IR(2)=8;IR(3)=13 (CR)
0371 0120
0372 0307         JMS; PRINT
0373 0042
0374 0212 LF,     FIM 1<;138       / IR(2)=8;IR(3)=10 (LF)
0375 0120
0376 0307         JMS ;PRINT
0377 0300         BBL
                  /
                  /
                  OCTAL
                  /
```

ADDRESS LABELS USED IN PROGRAM

ADDITN	0154	AD1	0165	AD2	0212	BEGIN	0000
CLEAR	0262	CLRRAM	0260	COMADD	0105	CRLF	0367
LF	0373	L1	0273	L2	0302	NEXT	0017
OVERFL	0177	OVFL1	0244	PRINT	0307	REP	0012
REP1	0131	SBR1	0271	SBR2	0300	SKIP	0233
SKP1	0223	ST	0030	STORE	0123	ST1	0056
ST12	0362	ST7	0330	ST8	0334	ST9	0360
TEST	0036	WRITE	0112	XXX	0242		

XIII. MCS-4 PROM PROGRAMMING SYSTEM

A. General System Description and Operating Instructions

Intel has developed a low-cost micro computer programming system for its electrically programmable ROMs. Using Intel's eight bit micro computer system and a standard ASR 33 teletype (TTY), a complete low cost and easy to use ROM programming system may be assembled. The system features the following functions:

1) Memory loading
2) Format checking
3) ROM programming
4) Error checking
5) Program listing

For specifications of the Intel PROMs, refer to the Intel Data Catalog.

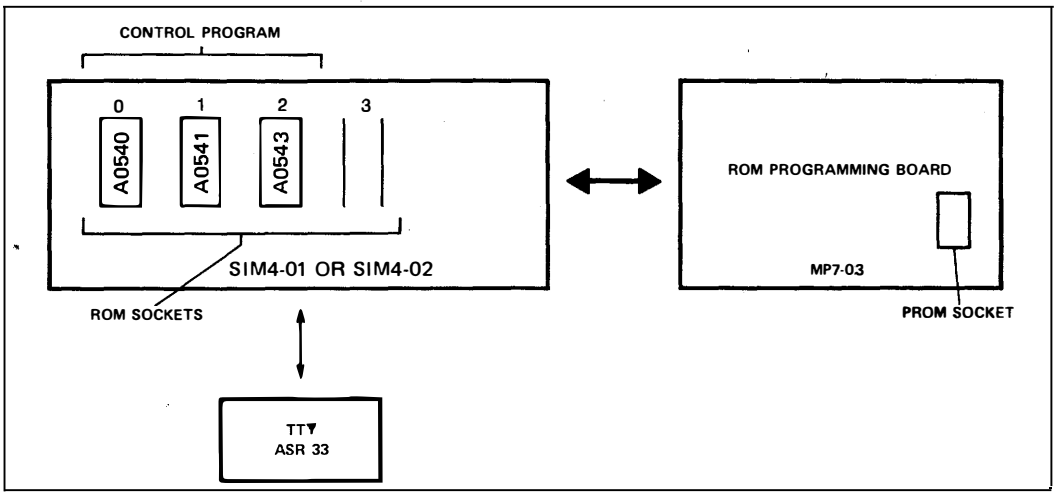


Figure 27. MCS-4 PROM Programming System

This programming system has four basic parts:

1) The micro computer (SIM4-01 or SIM4-02)
   This is the **MCS-4** prototype board, a complete micro computer which uses 1702A PROMs for the microprogram control. The total system is controlled by the 4004 CPU.

2) The control program (A0540, A0541, A0543)
   These control ROMs contain the microprograms which control the bootstrap loading, programming, format and error checking, and listing functions. For high speed programming of Intel's new 1702A PROM (three minutes) use control PROM A0543 in place of A0542.

3) The programmer (MP7-03)
   This is the programmer board which contains all of the timing and level shifting required to program the Intel ROMs. This is the successor of the MP7-02.

4) ASR 33 (Automatic Send Receive) Teletype
   This provides both the keyboard and paper tape I/O devices for the programming system.

In addition, a short-wave ultraviolet light is required if the erasable and reprogrammable 1702s are used.

This system has two modes of operation:

1) Automatic – A paper tape is used in conjunction with the tape reader on the teletype. The tape contains the program for the ROM.

2) Manual – The keyboard of the TTY is used to enter the data content of the word to be programmed.

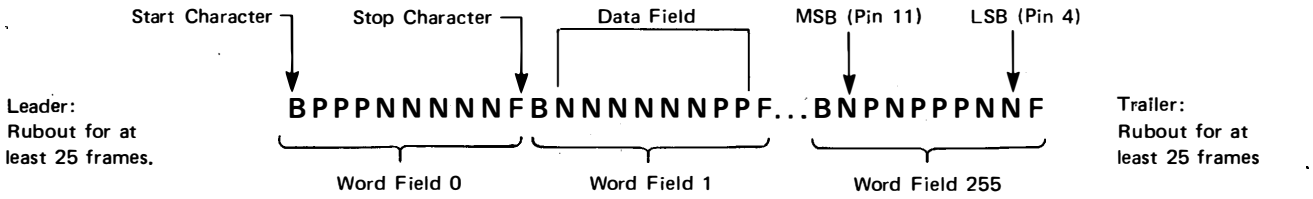
PROGRAMMING THE 1602A/1702A

Information is introduced by selectively programming "1"s (output high) and "0"s (output low) into the proper bit locations. Note that these ROMs are defined in terms of positive logic.

Word address selection is done by the same decoding circuitry used in the READ mode. The eight output terminals are used as data inputs to determine the information pattern in the eight bits of each word. A low data input level (ground – P on tape) will leave a "1" and a high data input level (+48V – N on tape) will allow programming of "0". All eight bits of one word are programmed simultaneously by setting the desired bit information patterns on the data input terminals.

TAPE FORMAT

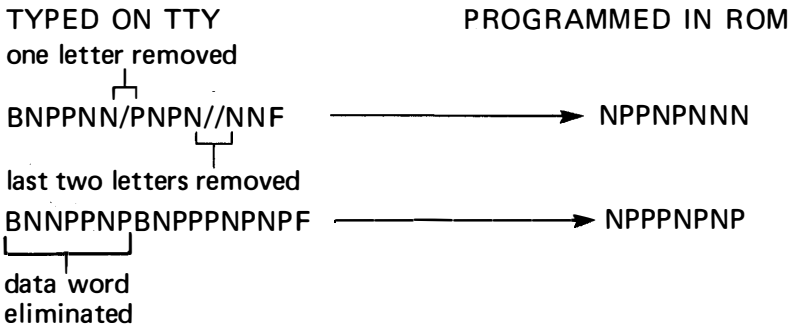
The tape reader used with a model 33 ASR teletype accepts 1" wide paper tape using 7 or 8 bit ASCII code. For a tape to correctly program a 1602A/1702A, it must follow exactly the format rules below:



The format requirements are as follows:

1) There must be exactly 256 word fields in consecutive sequence, starting with word field 0 (all address lines low) to program an entire ROM. If a short tape is needed to program only a portion of the ROM, the same format requirements apply.

2) Each word field must consist of ten consecutive characters, the first of which must be the start character B. Following that start character, there must be exactly eight data characters (P's or N's) and ending with the stop character F. NO OTHER CHARACTERS ARE ALLOWED ANYWHERE IN A WORD FIELD. If an error is made while preparing a tape and the stop character "F" has not been typed, a typed "B" will eliminate the previous characters entered. **This is a feature not available on Intel's 7600 programmer; the format shown in the Intel Data Catalog must be used when preparing tapes for other programming systems.** An example of this error correcting feature is shown below:

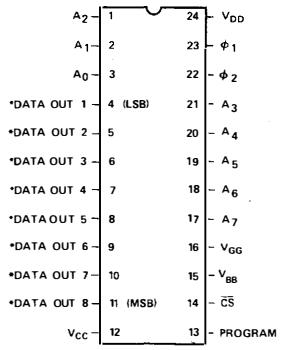


If any character other than P or N is entered, a format error is indicated. If the stop character is entered before the error is noticed, the entire word field, including the B and F, must be rubbed out. **Within the word field, a P results in a high level output, and N results in a low level output.** The first data character corresponds to the desired output for data bit 8 (pin 11), the second for data bit 7 (pin 10), etc.

3) Preceding the first word field and following the last word field, there must be a leader/ trailer length of at least 25 characters. This should consist of rubout punches.

4) Between word fields, comments not containing B's or F's may be inserted. It is important that a carriage return and line feed characters be inserted (as a "comment") just before each word field or at least between every four word fields. When these carriage returns are inserted, the tape may be easily listed on the teletype for purposes of error checking. It may also be helpful to insert the word number (as a "comment") at least every four word fields.

PROM PIN CONFIGURATION



*THIS PIN IS THE DATA INPUT LEAD FOR THE 1602/1702 DURING PROGRAMMING

IMPORTANT

It should be noted that the PROM's are described in the data sheet with respect to "positive logic" (high level = p-logic 1). On the other hand the MCS-4 system is defined in terms of "negative logic" (low level = n-logic 1). As a result, when 1602/1702 ROM's are being programmed to simulate the 4001, characters should be defined as P=high level = n-logic 0 or an N = low level = n-logic 1. For instance, consider the instruction code for ADM (one of the 45 instructions for the MCS-4),

11101011

When entering this code to the programmer it should be typed,

BNNNPNPNNF

This is the code that will be put into the 4001 when the final system is defined.

OPERATING THE PROGRAMMER

The SIM4 is used as the micro computer controller for the programming. It presents data and addresses to the PROM to be programmed and controls the programming pulse. The following steps must be followed when programming a PROM:

1. Place control ROMs (A0540, A0541, A0543) in SIM4 board.
2. Turn on system power.
3. Turn on TTY to "line" position.
4. Reset system.
5. Insert PROM into MP7-03.
6. Load data from TTY and program PROM.
7. Remove PROM from MP7-03. To prevent programming of unwanted bits, never turn power on or off while the PROM is in the MP7-03.

OPERATING THE PROGRAMMER

PROGRAMMING

Two different modes of operation are available*

1) A complete program tape consisting of 256 data words in sequential order may be used.

   a) To program the complete ROM place the ROM to be programmed into the socket on the MP7-02 board, and then type,

S	(start command)
ØØØ	(initial address -address Ø of the ROM)
255	(final address -address 255 of the ROM)

   Start the tape (data words may also be entered in sequence manually). The ROM will be programmed in all 256 locations.

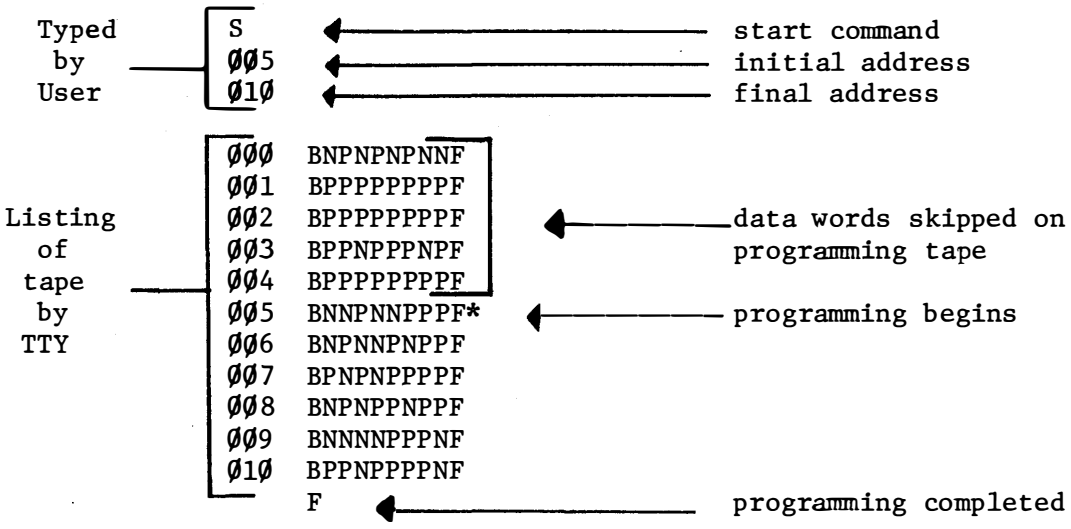
   b) To skip a section of the ROM (skipping the same number of data words on the tape) and then program a section of the ROM while still keeping the addresses in sequential form on the complete tape, type,

S	(start command)
XXX	(initial address)
YYY	(final address)

   Start the tape. The ROM will only be programmed in the specified locations.

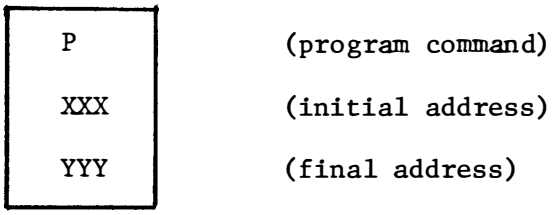
*Operate the TTY in the "line" position. Depress the RETURN key (RET) after each manual data word entry. When using the tape reader to enter the program data, switch the reader to "start" after the final address is entered.

EXAMPLE 1:



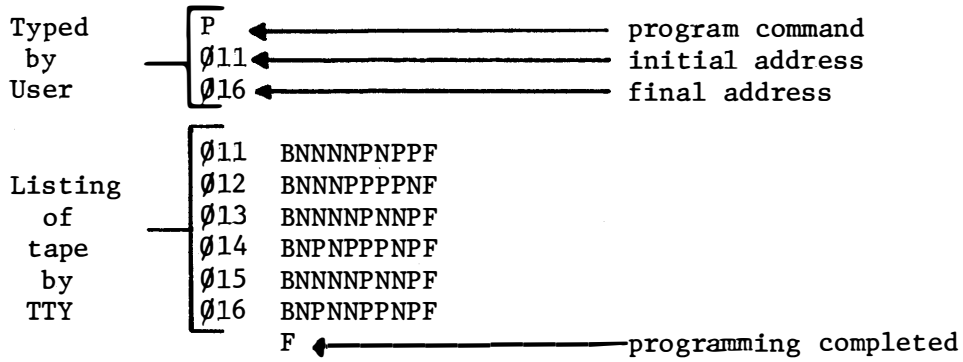
In this example the first five data words on the tape are skipped and the next six data words are programmed in the ROM locations ØØ5 to Ø1Ø. The "*" indicates the first instruction programmed, and the "F" at end of the listing indicates the completion of the programming.

2) To program any portion of the ROM without skipping a section of the tape, use a short tape (data words in sequence) and type



Start the tape (data words may also be entered manually).

EXAMPLE 2:

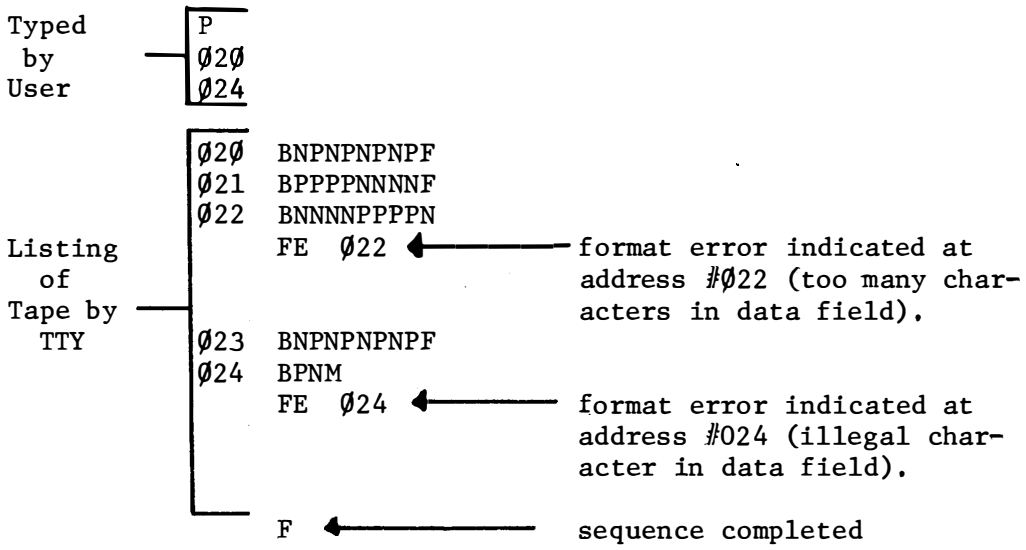


This example shows that the first six instruction words in sequence on a tape are read and directly programmed into ROM locations Ø11 through Ø16 without skipping.

FORMAT CHECKING

When the programmer detects the first format error (data words enter either on tape or manually), it will stop programming the ROM, and it will print out the address where the format error occurred. If a tape is being used, the programming system will continue to list the content of the tape and will print out the address of each subsequent format error.

EXAMPLE 3:

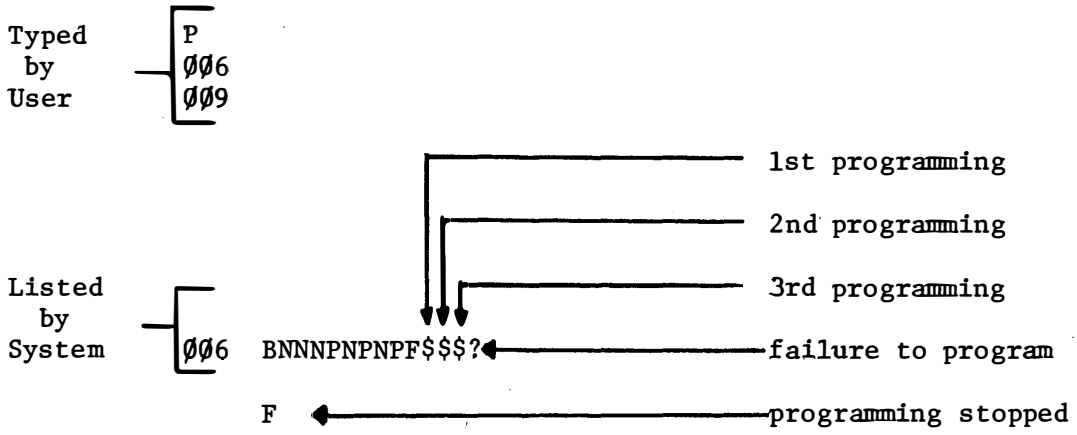


If data words are being entered manually and a format error is encountered, programming may be continued by entering the PROGRAM command, "P", the address of the error, and the final address. The error may be corrected and programming completed.

ERROR CHECKING

After each location in ROM is programmed, the content of the location is read and compared against the programming data. In the event that the programming is not correct, the ROM location will be programmed again. The MCS-4 programming system allows each location of the ROM to be reprogrammed up to four times. A "$" will be printed for each reprogramming. If a location in ROM will not accept a data word after the fourth time, the system will stop programming and a "?" will be printed. This feature of the system guarantees that the programmed ROM will be correct, and incompletely erased or defective ROM's will be identified.

EXAMPLE 4:



If a location in the ROM will not program, a new ROM must be inserted in the programmer. The system must be reset before continuing. (If erasable ROMs are being used, the "faulty" ROM should be erased and reprogrammed).

PROGRAM LISTING

After the programming is complete, the complete content of the ROM, or any portion may be listed on the teletype. A duplicated programming tape may also be made using the teletype tape punch. To list the ROM, type

L	(list command)
XXX	(initial address)
YYY	(final address)

EXAMPLE 5:

Typed by User
L
ØØØ
Ø1Ø

Listed by System
ØØØ BPPPPPPPPF; ØØ1 BPPPPPPPPF; ØØ2 BPPPPPPPPF; ØØ3 BPPPPPPPPF;
ØØ4 BPPPPPPPPF; ØØ5 BNNPNNPPPF; ØØ6 BNPNNPNPPF; ØØ7 BPNPNPPPPF;
ØØ8 BNPNPPNPPF; ØØ9 BNNNNPPPNF; Ø1Ø BPPNPPPPNF;
F ← listing complete

Note that this is a listing of the ROM programmed in Example 1. The listing feature may also be used to verify that a 1701 or 1702 is completely erased. If the PROM is completely erased, "P's" will be listed in every location.

1701, 1702 ERASING PROCEDURE

The 1701 and 1702 may be erased by exposure to high intensity short-wave ultraviolet light at a wavelength of 2537 Å. The recommended integrated dose (i.e., UV intensity x exposure time) is 6W-sec/cm^2. Example of ultraviolet sources which can erase the 1702A in 10 to 20 minutes is the Model S-52 and Model UVS-54 short-wave ultraviolet lamps manufactured by Ultra-Violet Products, Inc. (San Gabriel, California). **The lamps should be used without short-wave filters, and the 1702A to be erased should be placed about one inch away from the lamp tubes.**

B. MP7-03 Programming System

The MP7-03 is the PROM programming board which easily interfaces with the MCS-4. All address and data lines are completely TTL compatible. The MP7-03 requires +5VDC @ 0.8 amps, −10VDC @ 0.1 amps, and 50 Vrms @ 1 amp. Two Stancor P8180 (or equivalent) filament transformers (25.2 Vrms @ 1 amp) with their secondaries connected in series provide the 50 Vrms.

This programmer board is the successor of the MP7-02. The MP7-03 enables programming of Intel's new 1702A, a pin-for-pin replacement for the 1702.

When the MP7-03 is used under SIM4-01 or SIM4-02 control with control ROM A0542 replaced by A0543, the 1702A may be programmed five times faster than the 1702 in less than five minutes.

IMPORTANT:
Only use the A0543 control PROM when programming the new 1702A. Never use it when programming the 1702. The programming duty cycle is too high for the 1702 and it may be permanently damaged.

The MP7-03 features three data control options:

1) Data-in switch (Normal-Complement). If this switch is in the complement position, data into the PROM is complemented.

2) Data-out switch (Normal-Complement). If this switch is in the complement position, data read from the PROM is complemented.

3) Data-out switch (Enable-Disable). If this switch is in the enable position, data may be read from the PROM. In the disable position, the output line may float up to a high level (logic "1"). As a result, the input ports on the prototype system may be used for other functions without removing the MP7-03 card.

MP7-03 Programmer Board Specifications

Features:
- High speed programming of Intel's new 1702A (three minutes)
- Inputs and outputs TTL compatible
- Board sold complete with transformers, capacitor and connector
- Directly interfaces with SIM4-01 or SIM4-02 Boards

Dimensions:
8.4 inches high
9.5 inches deep

Power Requirement:
V_{CC} = +5 @ 0.8 amps
TTL GRD = 0V
*V_{DD} = −10V @ 0.1 amps
V_P = 50Vrms @ 1 amp

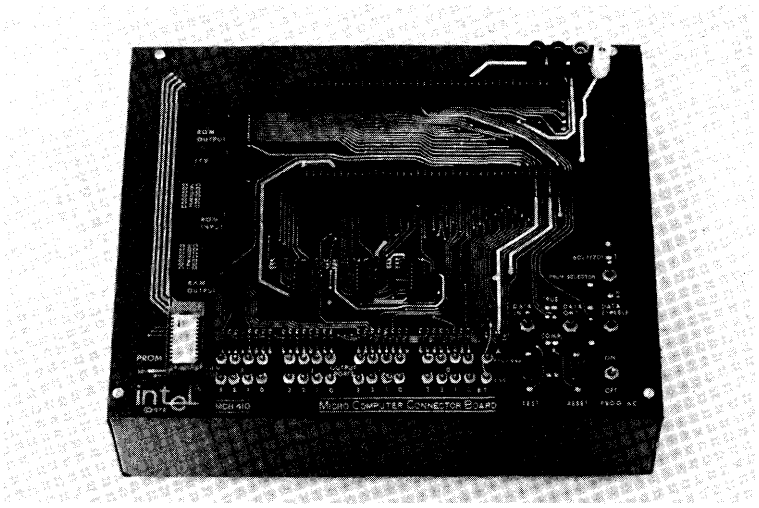
Connector:
a. Solder lug type/Amphenol
72 pin connector
P/N 225-23621-101
b. Wire wrap type - Amphenol
72 pin connector
P/N 261-15636
c. Wire wrap type - CDC
72 pin connector
P/N VPB01E36E00A1

*This board may be used with a −10V supply because a pair of diodes (i.e. 1N914 or equivalent) are located on the board in series with the supply. Select the appropriate pin for either −9V or −10V operation.

A micro computer bulletin which describes the modification of the MP7-02 for programming the 1602A/1702A is available on request. These modifications include complete failsafe circuitry (now on MP7-03) to protect the PROMs and the 50V power supply.

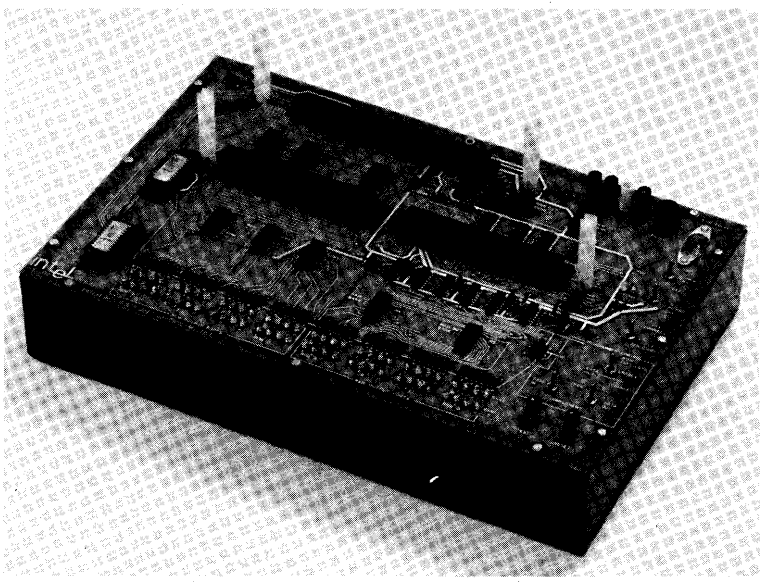
MCB4-10 SYSTEM INTERCONNECT and CONTROL MODULE

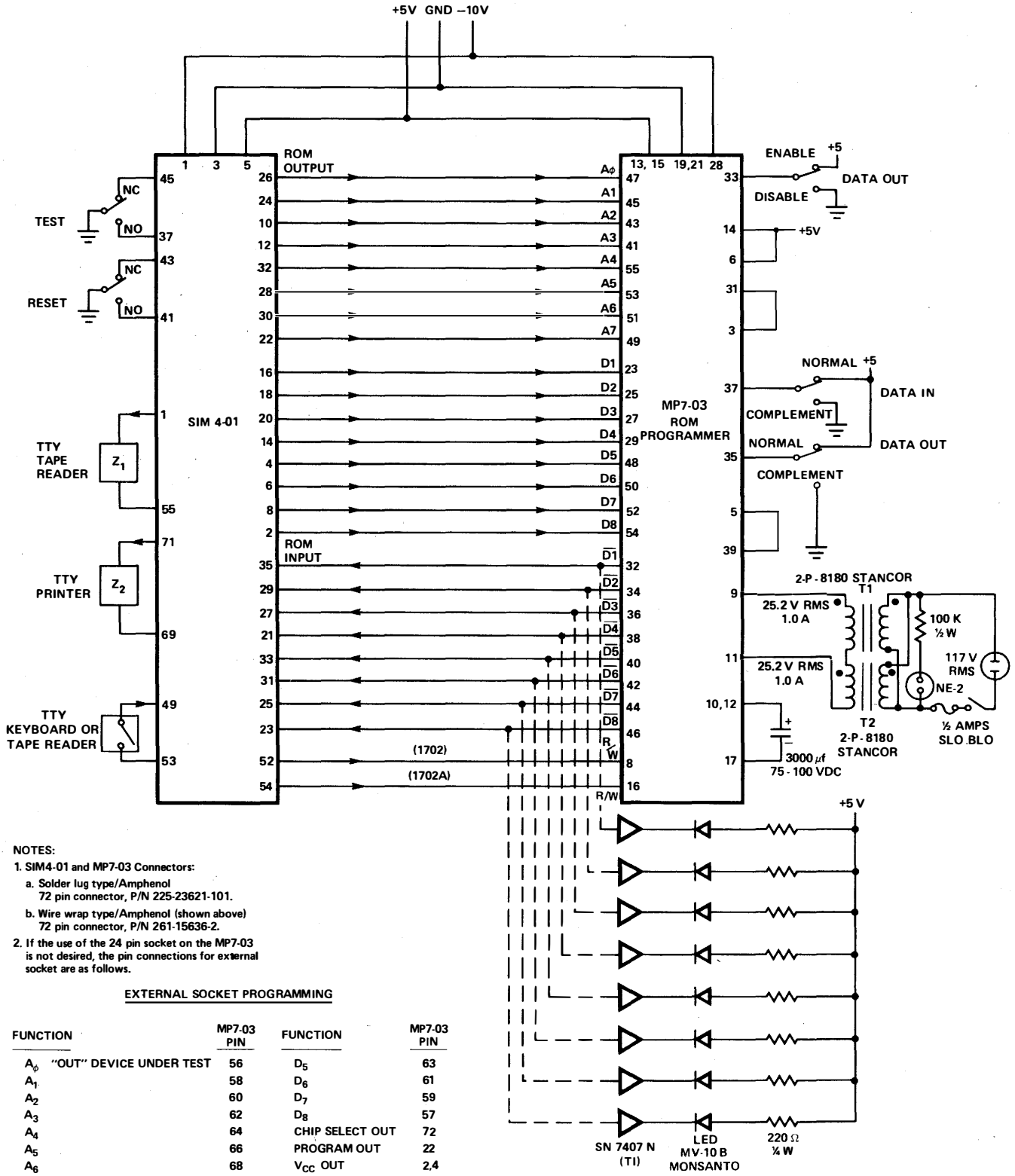
This module provides the complete interconnection between the SIM4-01 and the MP7-03. In addition to the connectors for both boards, there are LED data displays of each microcomputer output port, control switches, the 50 Vrms transformers, and a socket for the PROM being programmed. Plug-in connectors for each input and output port are provided. The SIM4-01 when used alone with the MCB4-10 is a complete microcomputer (except for power supplies). See Appendix E for a complete description of the MCB4-10.



MCB4-20 SYSTEM INTERCONNECT and CONTROL MODULE

This module provides the complete interconnection between the SIM4-02 and the MP7-03. This total package may be used for both a PROM programmer and a microcomputer developmental system. The MCB4-20 has the same basic features as the MCB4-10, and in addition, it provides a socket which can be used for duplication. The MCB4-20 is also fully described in Appendix E.





NOTES:

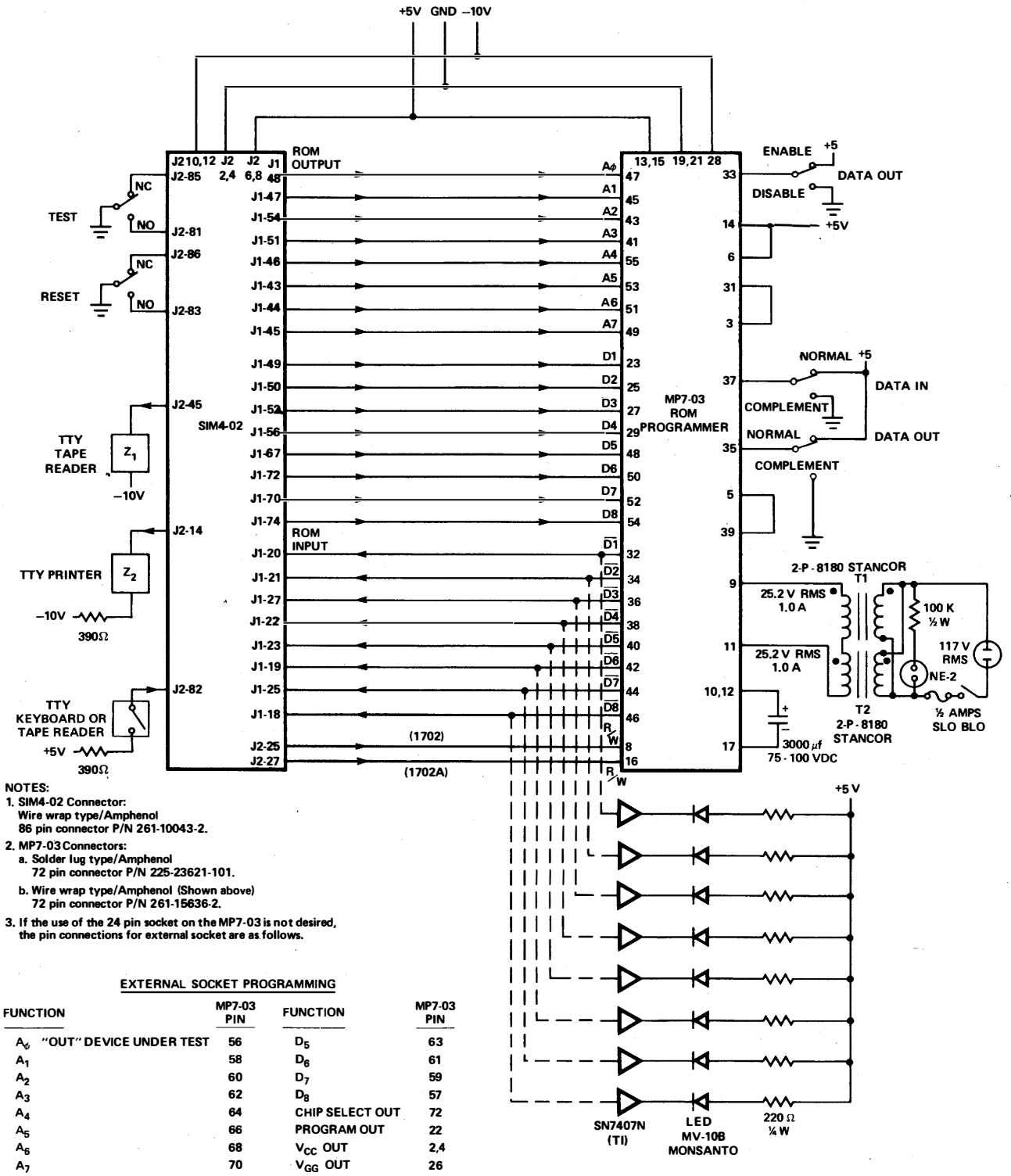
1. SIM4-01 and MP7-03 Connectors:
   a. Solder lug type/Amphenol 72 pin connector, P/N 225-23621-101.
   b. Wire wrap type/Amphenol (shown above) 72 pin connector, P/N 261-15636-2.
2. If the use of the 24 pin socket on the MP7-03 is not desired, the pin connections for external socket are as follows.

EXTERNAL SOCKET PROGRAMMING

FUNCTION	MP7-03 PIN	FUNCTION	MP7-03 PIN
A_ϕ "OUT" DEVICE UNDER TEST	56	D_5	63
A_1	58	D_6	61
A_2	60	D_7	59
A_3	62	D_8	57
A_4	64	CHIP SELECT OUT	72
A_5	66	PROGRAM OUT	22
A_6	68	V_{CC} OUT	2,4
A_7	70	V_{GG} OUT	26
D_1 "OUT" DEVICE UNDER TEST	71	V_{BB} OUT	24
D_2	69	V_{DD} OUT	30
D_3	67	ϕ_1, ϕ_2 OUT	2
D_4	65		

This is a simple module that can be built in the laboratory to interconnect the MP7-03 and the SIM4-01. The complete interconnection between the SIM4-01 and the MP7-03 is provided by the MCB4-10 system interface and control module.

Figure 28. MP7-03/SIM4-01 PROM Programming System



NOTES:

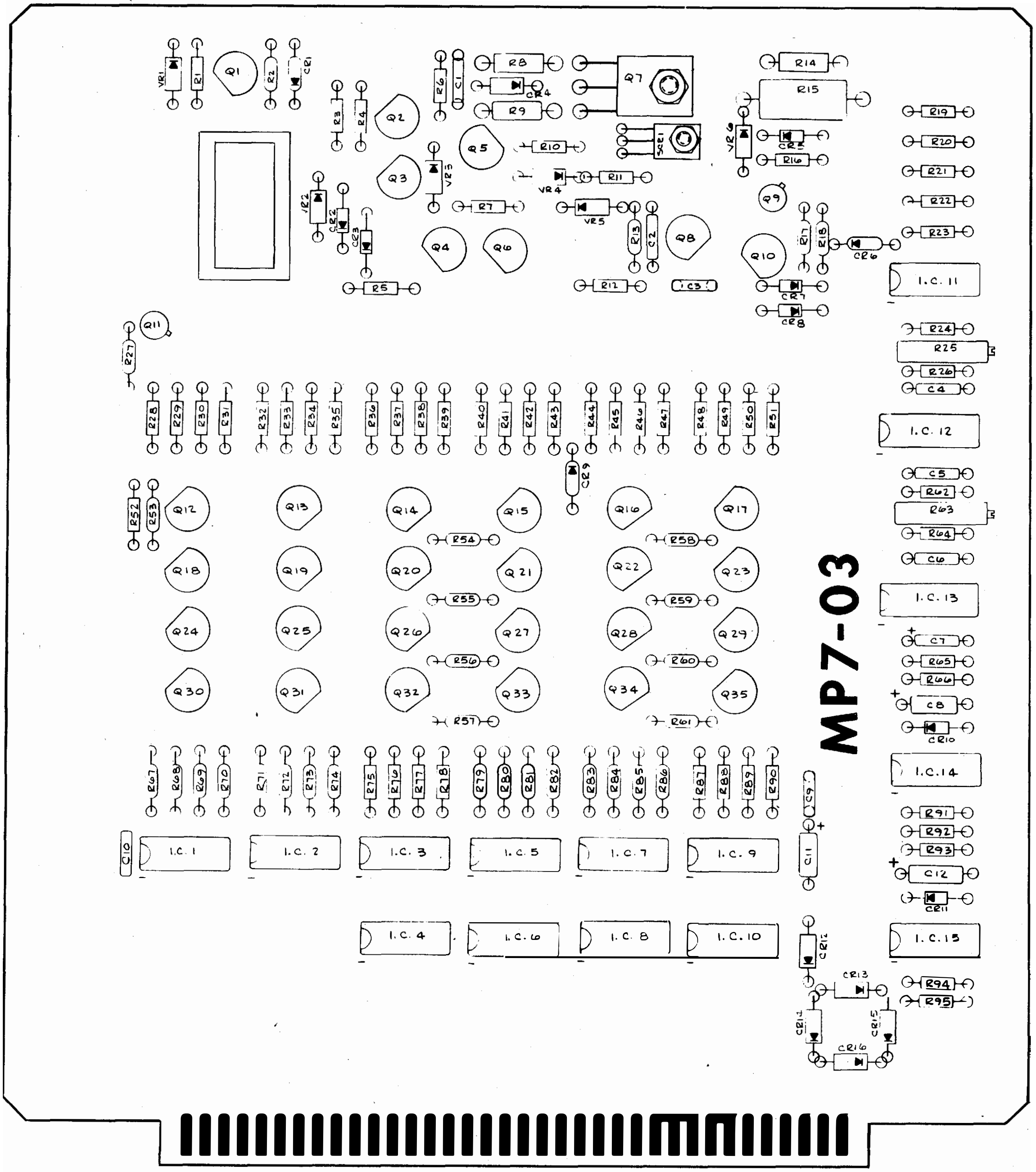
1. SIM4-02 Connector:
   Wire wrap type/Amphenol
   86 pin connector P/N 261-10043-2.
2. MP7-03 Connectors:
   a. Solder lug type/Amphenol
      72 pin connector P/N 225-23621-101.
   b. Wire wrap type/Amphenol (Shown above)
      72 pin connector P/N 261-15636-2.
3. If the use of the 24 pin socket on the MP7-03 is not desired, the pin connections for external socket are as follows.

EXTERNAL SOCKET PROGRAMMING

FUNCTION	MP7-03 PIN	FUNCTION	MP7-03 PIN
A_ϕ "OUT" DEVICE UNDER TEST	56	D_5	63
A_1	58	D_6	61
A_2	60	D_7	59
A_3	62	D_8	57
A_4	64	CHIP SELECT OUT	72
A_5	66	PROGRAM OUT	22
A_6	68	V_{CC} OUT	2,4
A_7	70	V_{GG} OUT	26
D_1 "OUT" DEVICE UNDER TEST	71	V_{BB} OUT	24
D_2	69	V_{DD} OUT	30
D_3	67	ϕ_1, ϕ_2 OUT	2
D_4	65		

This is a simple module that can be built in the laboratory to interconnect the MP7-03 and the SIM4-02. The complete interconnection between the SIM4-02 and the MP7-03 is provided by the MCB4-20 system interface and control module.

Figure 29. MP7-03/SIM4-02 PROM Programming System

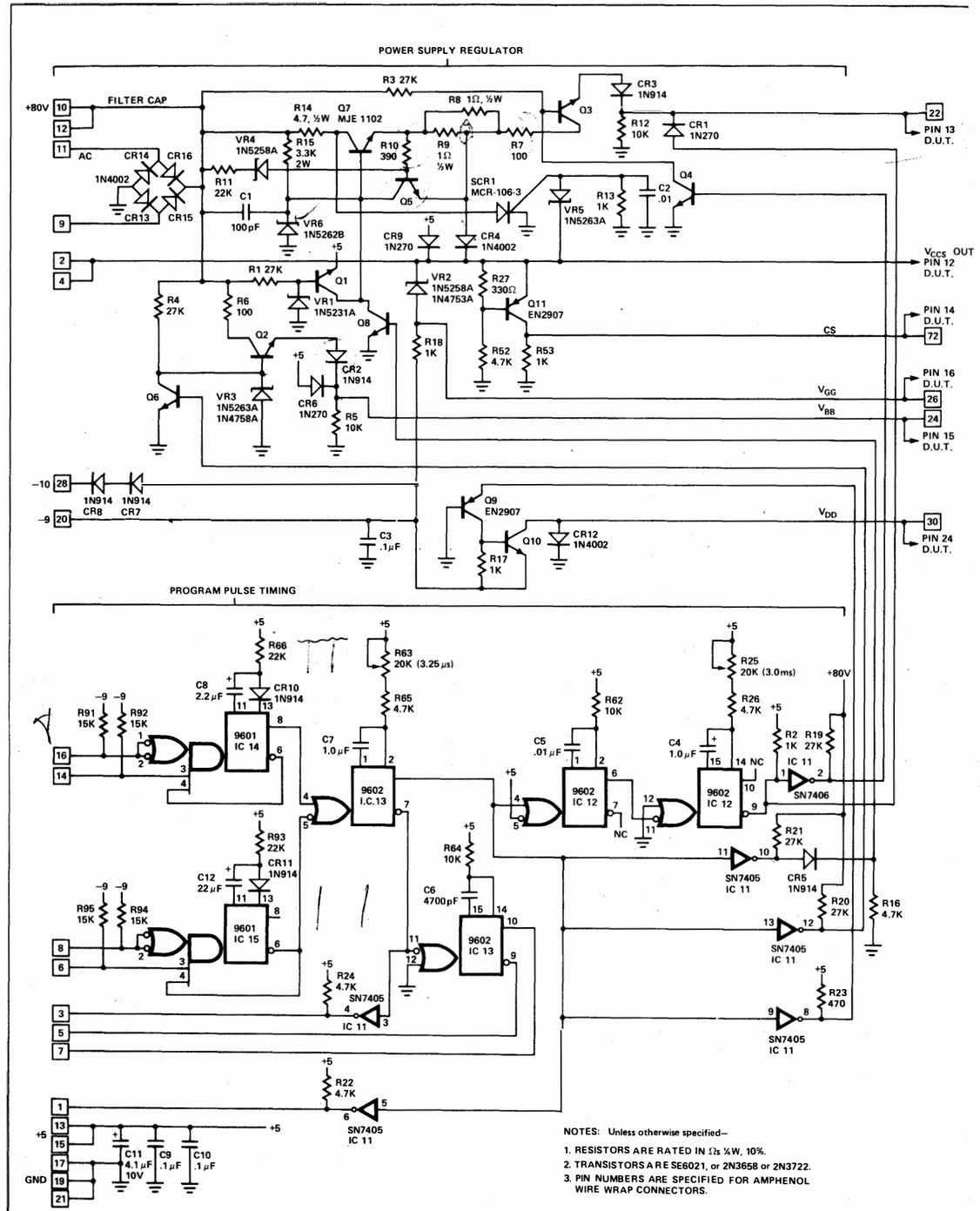


Solder Connector P/N 225-23621-101	$\overline{R}$	$\overline{P}$	$\overline{N}$	$\overline{M}$	$\overline{L}$	$\overline{K}$	$\overline{J}$	$\overline{H}$	$\overline{F}$	$\overline{E}$	$\overline{D}$	$\overline{C}$	$\overline{B}$	$\overline{A}$	Z	Y	X	W	V	U	T	S	R	P	N	M	L	K	J	H	F	E	D	C	B	A	Amphenol
Wirewrap Connector P/N 261-15636-2	71	69	67	65	63	61	59	57	55	53	51	49	47	45	43	41	39	37	35	33	31	29	27	25	23	21	19	17	15	13	11	9	7	5	3	1	
Wirewrap Connector P/N VPB01E36E00A1	72	70	68	66	64	62	60	58	56	54	52	50	48	46	44	42	40	38	36	34	32	30	28	26	24	22	20	18	16	14	12	10	8	6	4	2	CDC

Component Side of MP7-03 Card

Solder Connector P/N 225-23621-101	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	Amphenol
Wirewrap Connector P/N 261-15636-2	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34	36	38	40	42	44	46	48	50	52	54	56	58	60	62	64	66	68	70	72	
Wirewrap Connector P/N VPB01E36E00A1	1	3	5	7	9	11	13	15	17	19	21	23	25	27	29	31	33	35	37	39	41	43	45	47	49	51	53	55	57	59	61	63	65	67	69	71	CDC

Pin Definition — Reverse Side of MP7-03 Card



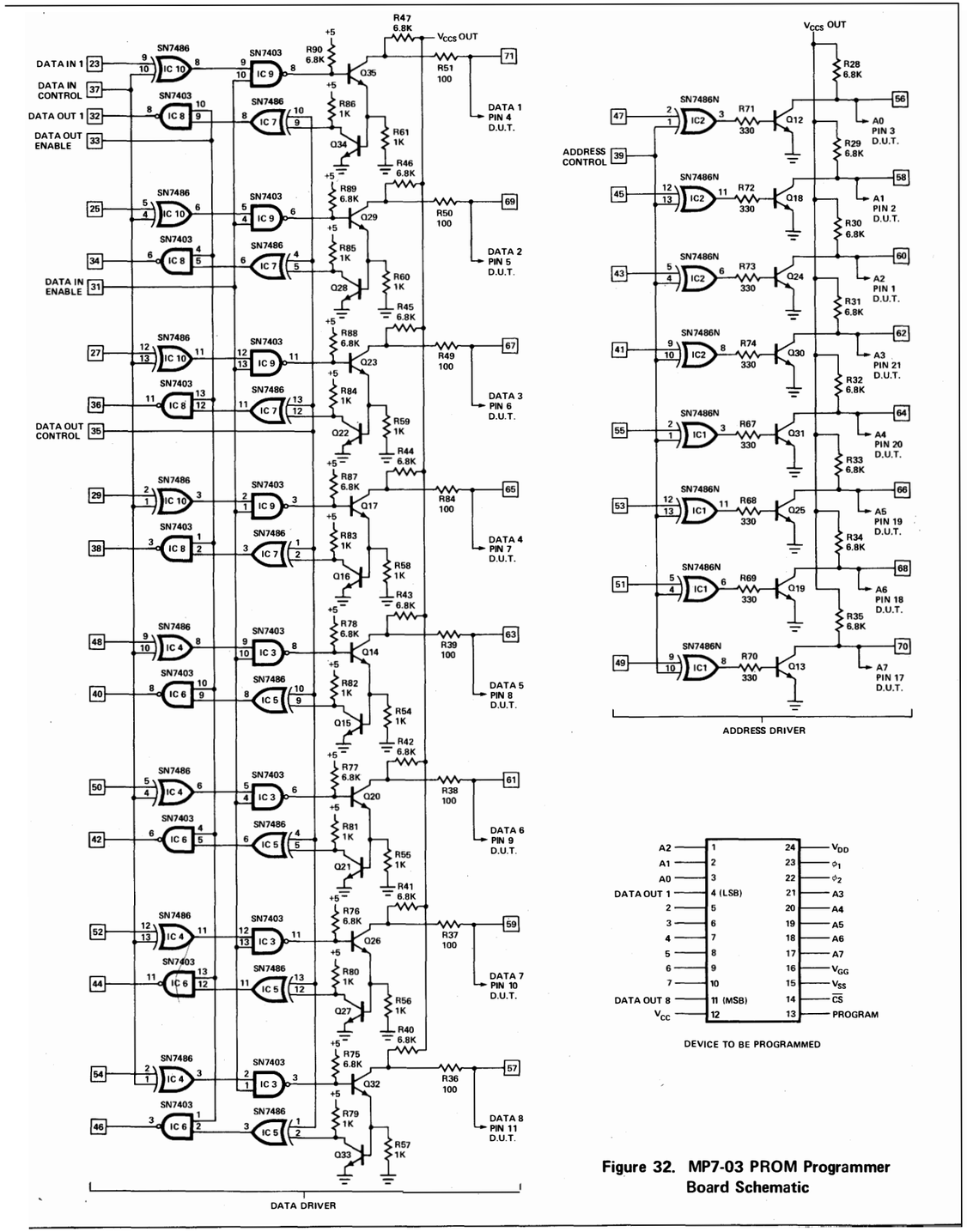


Figure 32. MP7-03 PROM Programmer Board Schematic

XIV. MCS-4 EVALUATION KIT USING THE 4001-0009

This kit provides both a convenient way of evaluating the MCS-4 parts and an educational vehicle to better understand the MCS-4 operation. The 4001-009 stores a microprogram that exercises the 4004 and 4002's and executes all of the 45 instructions in the MCS-4 instruction set.

Fig. 33 shows the hardware that should be used. The circuit for single pass/ continuous can be omitted if only continuous operation is sought. In this case 0_0 (RAM #0) should be connected directly to TEST.

The RESET signal can be provided by either a one-shot circuit or by a pulse generator in the "single pulse" mode. The width of the RESET signal must be at least 32 x 8 clock periods ($\approx$ 350 μsec) to fully clear the RAM storage. If the system is operated in the continuous mode, RESET needs to be applied only at power on. If the system works in the single pass mode, when END of SEQUENCE (Pin 0_3) is "1", the 4004 will "hang" on a loop where the address to Jump to on a jump on TEST = 1 condition is the address of the same jump on condition. To get out of the loop RESET must be applied.

To monitor the program operation a scope should be used in the "B delayed by A" mode. By using the delay time multiplier the program execution can be easily seen. The synchronization signals for the B and A traces are pin 13 of 4002-1 #0 and SYNC, pin 8 of the 4004, respectively.

The 4001-0009 has been coded with the internal chip select circuit always activated, therefore any address at A_3 time will cause the 4001 to be selected. This is different from the normal operation of the 4001 where only one code (out of 16) at A_3 time selects the 4001. The reason for doing so is that we can show the execution of JMS and JUN instructions to any chip number (the A_3 time code) and still use only 1 ROM chip.

The I/O pins of the 4001-0009 are all connected as inverting inputs with no resistors connected.

The two phase clocks, (ϕ_1 and ϕ_2) must be supplied externally according to the MCS-4 data sheet specs.

The program execution is 110 msec, using a clock period of 1.3 usec.

Although the CM-RAM$_i$ lines are not used in this configuration, they are being pulsed. If a scope is hooked up to these lines the waveforms may be observed.

Both 4002-1's must be used in order to fully execute the program stored in the ROM.

Attached is the program flow (with comments) and the truth table.

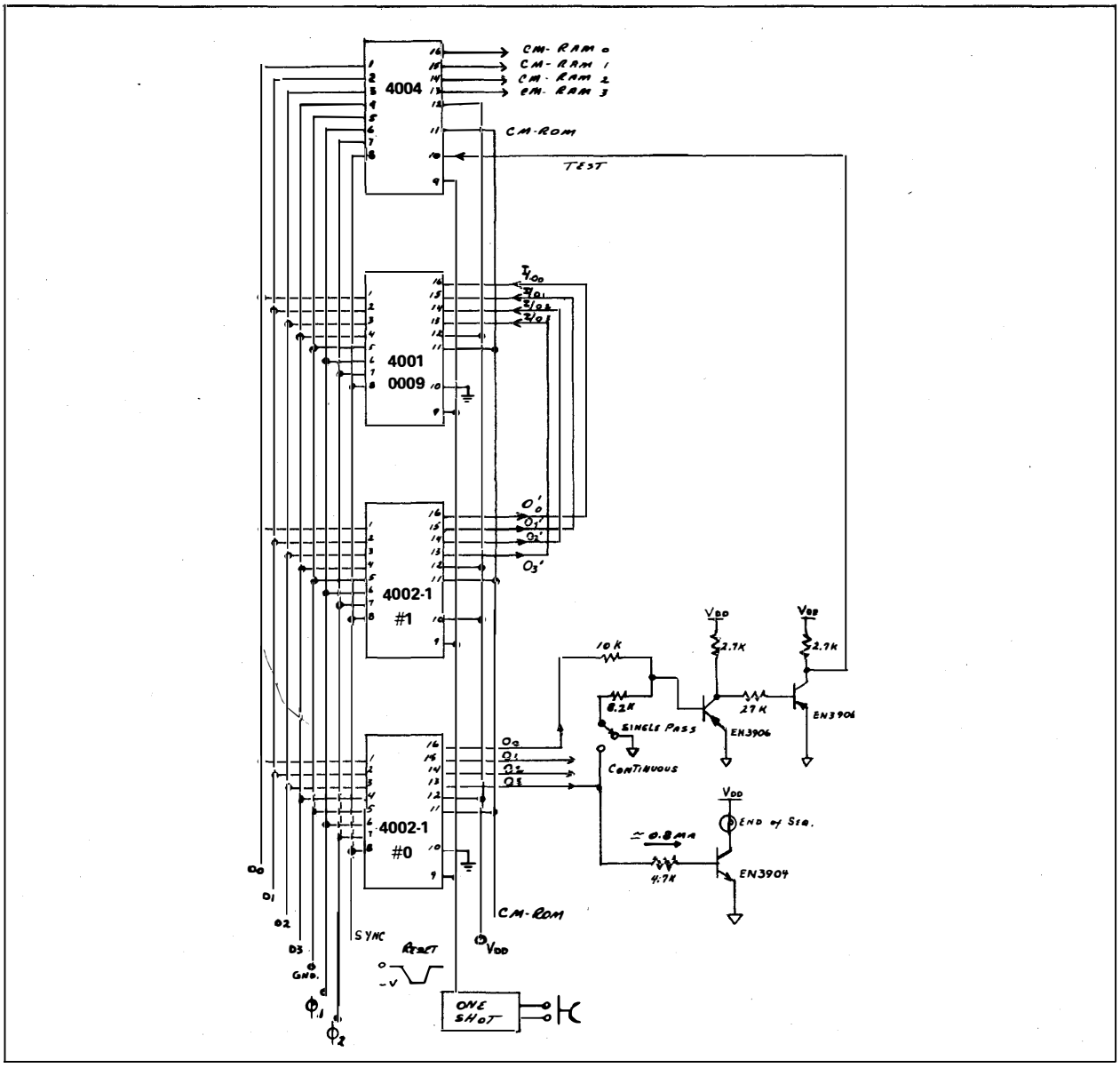


Figure 33. MCS-4 Evaluation Kit Using the 4001-0009

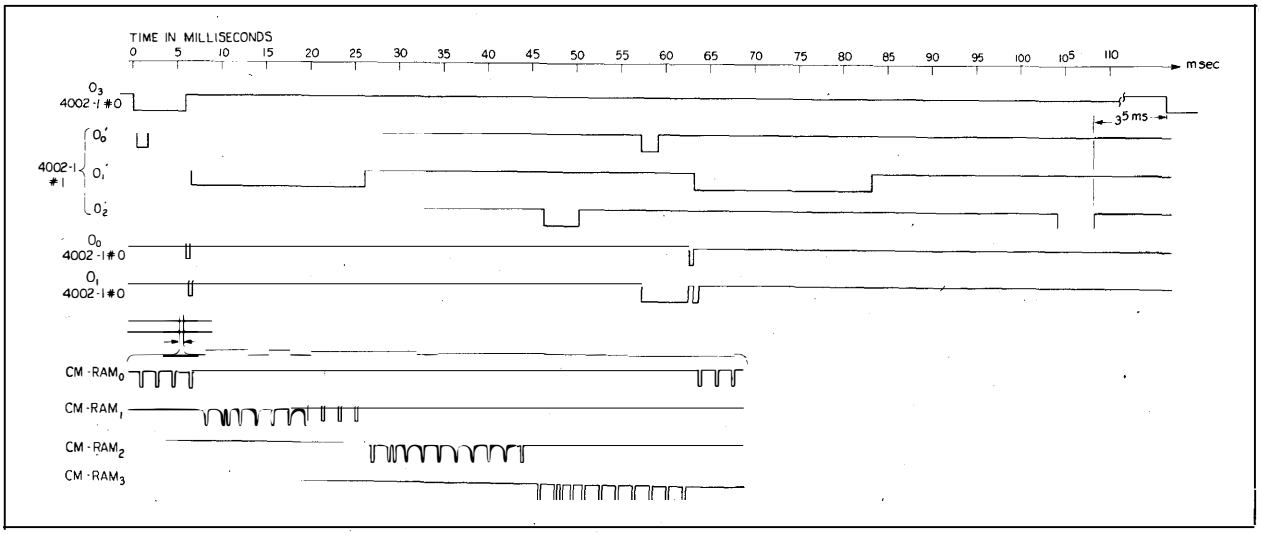
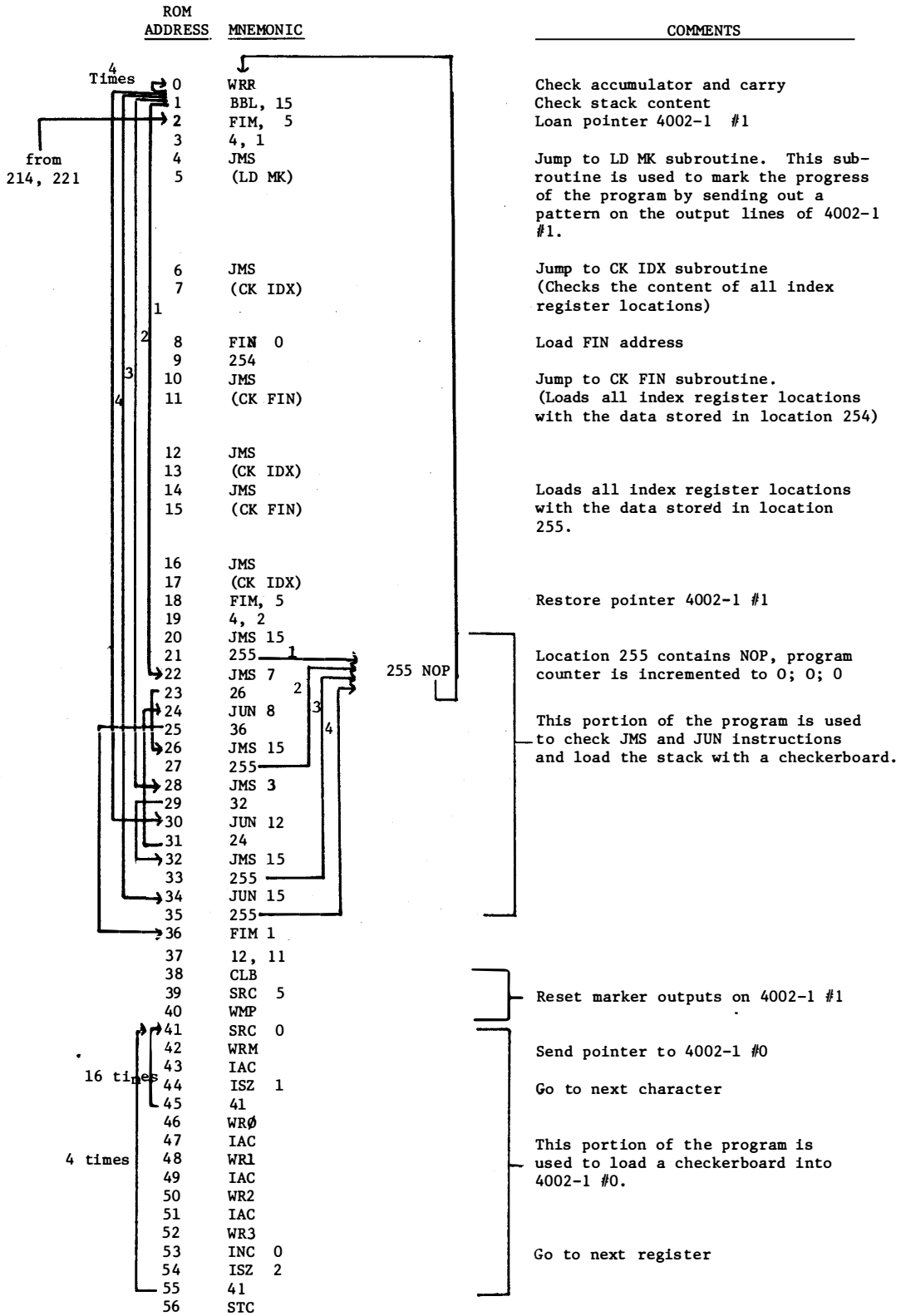
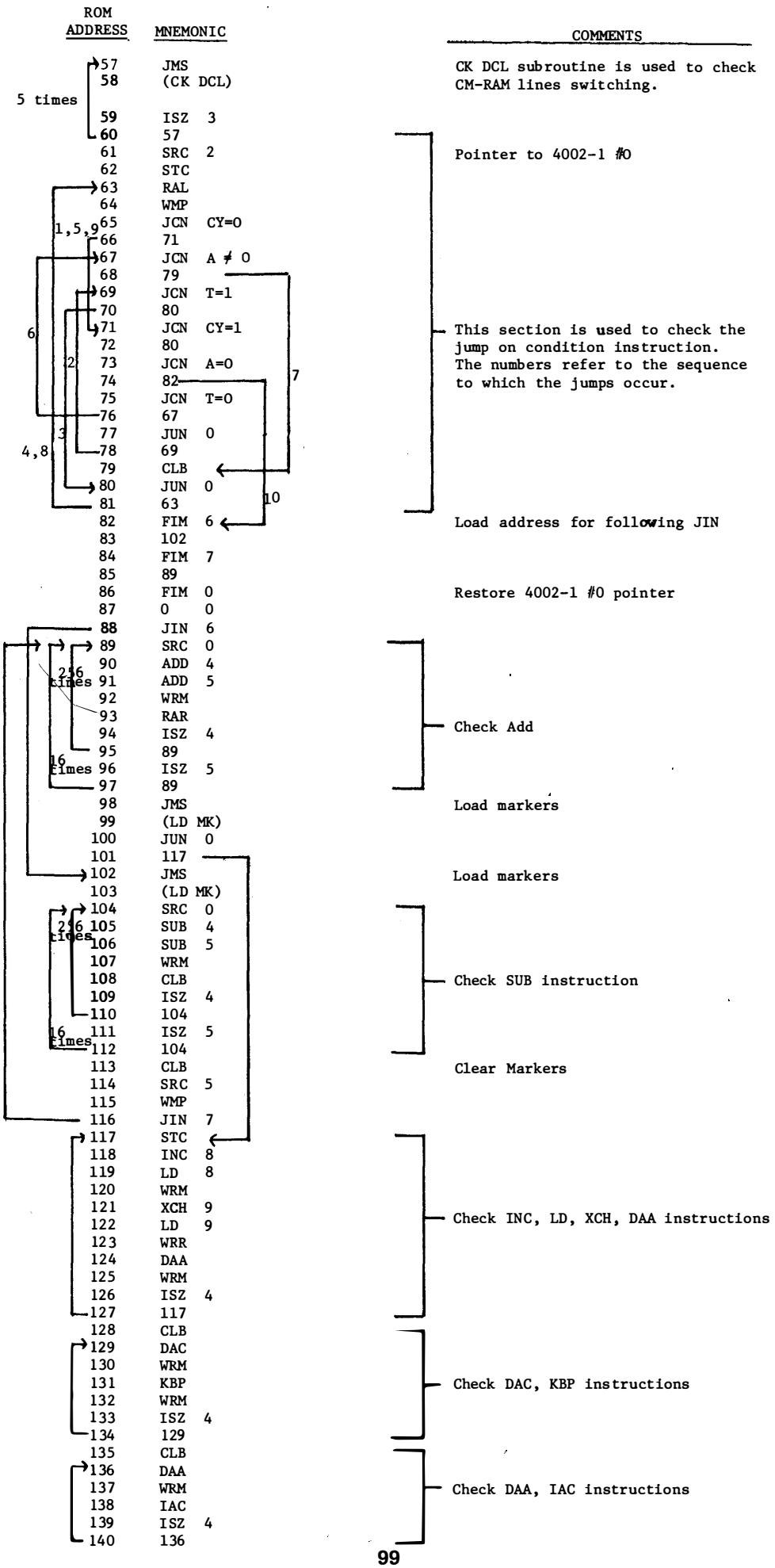


Figure 34. Timing Diagram for the MCS-4 Evaluation Kit Using the 4001-0009

4001-0009 MCS-4 EXERCISER PROGRAM




ROM ADDRESS
MNEMONIC
COMMENTS
57 JMS
58 (CK DCL)
CK DCL subroutine is used to check CM-RAM lines switching.
5 times
59 ISZ 3
60 57
61 SRC 2
Pointer to 4002-1 #0
62 STC
63 RAL
64 WMP
65 JCN CY=0
1,5,9
66 71
67 JCN A ≠ 0
68 79
69 JCN T=1
70 80
71 JCN CY=1
6
72 80
2
73 JCN A=0
74 82
7
75 JCN T=0
76 67
3
77 JUN 0
4,8
78 69
79 CLB
80 JUN 0
10
81 63
This section is used to check the jump on condition instruction.
The numbers refer to the sequence to which the jumps occur.
82 FIM 6
Load address for following JIN
83 102
84 FIM 7
85 89
86 FIM 0
Restore 4002-1 #0 pointer
87 0 0
88 JIN 6
89 SRC 0
90 ADD 4
256 times
91 ADD 5
92 WRM
93 RAR
94 ISZ 4
Check Add
95 89
16 times
96 ISZ 5
97 89
98 JMS
Load markers
99 (LD MK)
100 JUN 0
101 117
102 JMS
Load markers
103 (LD MK)
104 SRC 0
256 times
105 SUB 4
106 SUB 5
107 WRM
108 CLB
109 ISZ 4
Check SUB instruction
110 104
16 times
111 ISZ 5
112 104
113 CLB
Clear Markers
114 SRC 5
115 WMP
116 JIN 7
117 STC
118 INC 8
119 LD 8
120 WRM
121 XCH 9
122 LD 9
Check INC, LD, XCH, DAA instructions
123 WRR
124 DAA
125 WRM
126 ISZ 4
127 117
128 CLB
129 DAC
130 WRM
131 KBP
Check DAC, KBP instructions
132 WRM
133 ISZ 4
134 129
135 CLB
136 DAA
137 WRM
Check DAA, IAC instructions
138 IAC
139 ISZ 4
140 136

ROM ADDRESS	MNEMONIC	COMMENTS
141	LDM 15	Check TCC instruction
142	WRM	
143	TCC	
144	WRM	
145	JCN A ≠ 0	
146	141	
147	CLB	Clear markers
148	SRC 5	
149	WMP	
150	LDM 15	Check TCS instruction
151	TCS	
152	WRM	
153	STC	Check TCC, CMC, RAR instructions
154	TCS	
155	WRM	
156	CMC	
157	RAR	
158	WRM	
159	ISZ 4	
160	156	
161	FIM 2	Read content of all memory locations
162	12 0	
163	SRC 0	
164	RDM	
165	ISZ 1	
166	163	
167	RDØ	
168	RD1	
169	RD2	
170	RD3	
171	INC 0	
172	ISZ 4	
173	163	
174	FIM 0	Check SBM instruction
175	2 0	
176	FIM 1	
177	3 0	
178	SRC 0	
179	SBM	
180	INC 1	
181	SRC 1	
182	SBM	
183	WRM	
184	ISZ 3	
185	178	
186	FIM 0	Check ADM instruction
187	0 0	
188	FIM 0	
189	1 0	
190	CLB	
191	SRC 5	
192	WMP	
193	SRC 0	
194	ADM	
195	INC 1	
196	SRC 1	
197	ADM	
198	WRM	
199	ISZ 3	
200	193	
201	SRC 5	This portion controls the cycle. Status character Ø stores the cycle number. At the end of the 2nd cycle, if pin 13 of the 4002-1 #0 is connected to test of the 4004, the program will stop. To start again RESET signal must be applied to the system (single pass operation). If pin 13 is not connected to TEST the program will be in continuous mode.
202	RDØ	
203	JCN A=0	
204	215	
205	LDM 8	
206	SRC 0	
207	WMP	
208	CLB	
209	SRC 5	
210	WRØ	
211	JCN T=1	
212	211	
213	JUN 0	
214	2	
215	IAC	
216	WRØ	
217	LDM 2	
218	SRC 0	
219	WMP	
220	JUN 0	
221	2	

SUBROUTINES

```
                        222     SRC  5
                        223     LD  11
                        224     CLC
                        225     WMP
LD MK                   226     RAL
                        227     XCH 11
                        228     BBL, 0

                        229     SRC  0
                        230     SRC  1
                        231     SRC  2
                        232     SRC  3
CK IDX                  233     SRC  4
                        234     SRC  5
                        235     SRC  6
                        236     SRC  7
                        237     BBL, 0

                        238     FIN  1
                        239     FIN  2
                        240     FIN  3
                        241     FIN  4
                        242     FIN  5
CK FIN                  243     FIN  6
                        244     FIN  7
                        245     FIN  0
                        246     BBL, 0

                        247     LD   4
                        248     RAL
                        249     DCL
CK DCL                  250     XCH  4
                        251     RDR
                        252     BBL, 0

                        253     NOP
Data                    254     1111 1111
                        255     0000 0000 (NOP)
```

XV. APPENDICES

APPENDIX A. Electrical Characteristics of the MCS-4

The following pages provide the electrical characteristics for the MCS-4 system. For TTL compatibility, a resistor should be added between the output and V_{DD}. All outputs are push-pull MOS outputs.

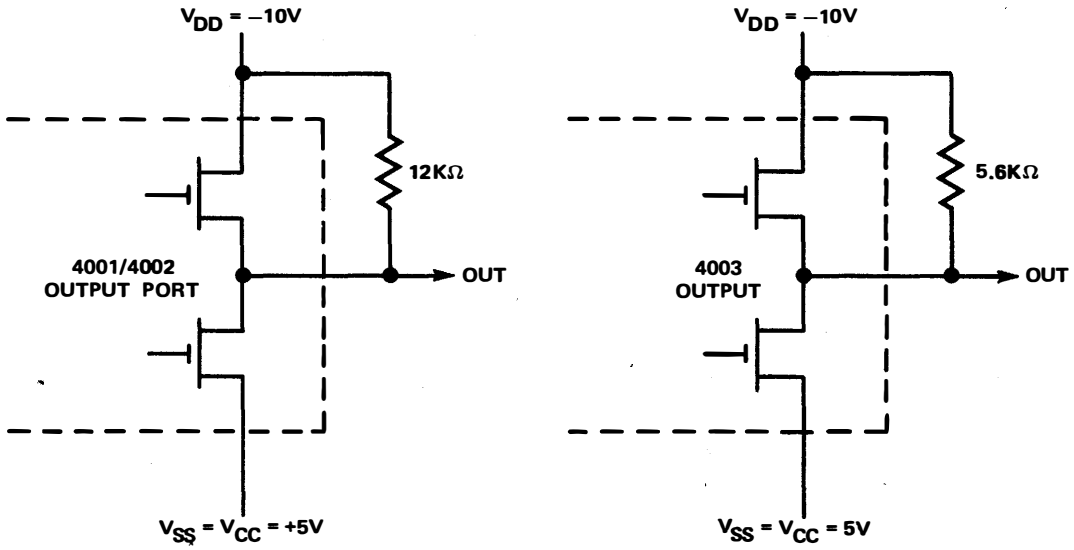


Figure 38. MCS-4 Output Configuration for TTL Compatibility

The input options for the 4001 are shown with the detailed description of the 4001 I/O ports. All other inputs are high impedance MOS inverters. Inputs to the 4001 and 4003 are TTL compatible (the 4001 non-inverting option is an exception).

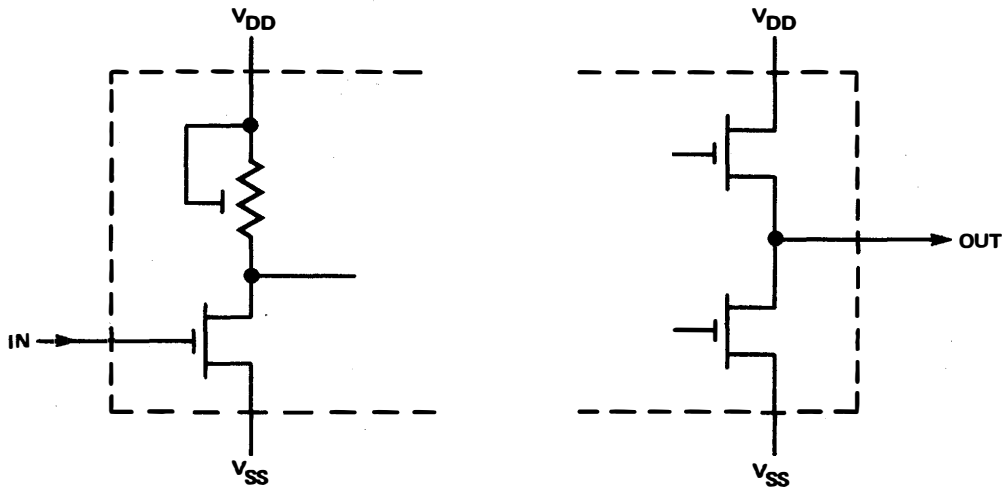


Figure 39. Typical MCS-4 Input and Output Circuitry

Absolute Maximum Ratings*

Ambient Temperature Under Bias	0°C to +70°C
Storage Temperature	−55°C to +150°C
Input Voltages and Supply Voltage With Respect to V_{SS}	+0.5 to −20V
Power Dissipation	1.0 W

*COMMENT

Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other condition above those indicated in the operational sections of this specification is not implied.

D. C. and Operating Characteristics – 4001, 4002, 4003, 4004

T_A = 0°C to +70°C; V_{DD} = −15V ±5%, V_{SS} = GND, $t_{\phi PW} = t_{\phi D1}$ = 400 nsec, $t_{\phi D2}$ = 150 nsec, unless otherwise specified
Logic "0" is defined as the more positive voltage (V_{IH}, V_{OH}), Logic "1" is defined as the more negative voltage (V_{IL}, V_{OL})

SUPPLY CURRENT

PRODUCT	SYMBOL	PARAMETER	LIMIT MIN.	LIMIT TYP.(1)	LIMIT MAX.	UNIT	TEST CONDITIONS
4001	I_{DD1}	AVERAGE SUPPLY CURRENT		15	30	mA	T_A = 25°C
4002	I_{DD2}	AVERAGE SUPPLY CURRENT		17	33	mA	T_A = 25°C
4003	I_{DD3}	AVERAGE SUPPLY CURRENT		5.0	8.5	mA	$t_{WL} = t_{WH}$ = 8 μsec; T_A = 25°C
4004	I_{DD4}	AVERAGE SUPPLY CURRENT		30	40	mA	T_A = 25°C

INPUT CHARACTERISTICS (ALL INPUTS EXCEPT I/O INPUT PINS)

4001/2/4	I_{LI}	INPUT LEAKAGE CURRENT			10	μA	$V_{IL} = V_{DD}$
4001/2/4	V_{IH}	INPUT HIGH VOLTAGE (EXCEPT CLOCKS)	V_{SS}−1.5		V_{SS}+0.3	V	
4001/2/4	V_{IL}	INPUT LOW VOLTAGE (EXCEPT CLOCKS)	V_{DD}		V_{SS}−5.5	V	
4001/2/4	V_{ILC}	CLOCK INPUT LOW VOLTAGE	V_{DD}		V_{SS}−13.4	V	
4001/2/4	V_{IHC}	CLOCK INPUT HIGH VOLTAGE	V_{SS}−1.5		V_{SS}+0.3	V	

OUTPUT CHARACTERISTICS (ALL OUTPUTS EXCEPT I/O OUTPUT PINS)

4001/2/4	I_{LO}	DATA BUS OUTPUT LEAKAGE CURRENT			10	μA	V_{OUT} = −12V, Chip disabled
4001/2/4	V_{OH}	OUTPUT HIGH VOLTAGE		V_{SS}	V_{SS}−0.5	V	Driving 4000 Series loads only
4001/2/4	I_{OL1}	DATA LINES SINKING CURRENT "1" LEVEL	10	18		mA	V_{OUT} = 0V
4004	I_{OL5}	CM-ROM SINKING CURRENT "1" LEVEL	6.5	12		mA	V_{OUT} = 0V
4004	I_{OL6}	CM-RAM LINES SINKING CURRENT "1" LEVEL	2.5	4		mA	V_{OUT} = 0V
4001/2/4	V_{OL1}	DATA LINES, CM LINES, SYNC OUTPUT LOW VOLTAGE	V_{SS}−12	V_{SS}−10	V_{SS}−6.5	V	I_{OL1} = 500 μA
4001/2/4	R_{OH1}	OUTPUT RESISTANCE DATA LINES "0" LEVEL		150	250	Ω	V_{OUT} = −0.5V
4004	R_{OH5}	CM-ROM OUTPUT RESISTANCE "0" LEVEL		320	600	Ω	V_{OUT} = −0.5V
4004	R_{OH6}	CM-RAM LINES OUTPUT RESISTANCE "0" LEVEL		1.1	1.8	KΩ	V_{OUT} = −0.5V

I/O INPUT CHARACTERISTICS

4001/3	I_{LI}	INPUT LEAKAGE CURRENT			10	μA	$V_{IL} = V_{DD}$
4001/3	V_{IH}	INPUT HIGH VOLTAGE	V_{SS}−1.5		V_{SS}+0.3		
4001/3	V_{IL}(2)	INPUT LOW VOLTAGE	V_{DD}		V_{SS}−4.2	V	
4001	R_I	I/O PINS INPUT RESISTANCE	10	18	35	KΩ	Internal input resistor is optional

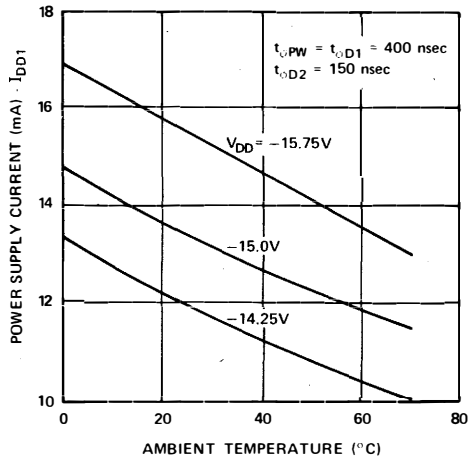
I/O OUTPUT CHARACTERISTICS

4001/2	I_{OL2}	I/O OUTPUT LINES SINKING CURRENT, "1" LEVEL	2.5	5		mA	V_{OUT} = 0V. For T²L compatibility a 12KΩ (±10%) resistor between output and V_{DD} should be added(3).
4003	I_{OL3}	PARALLEL OUT PINS SINKING CURRENT, "1" LEVEL	0.6	1.0		mA	V_{OUT} = 0V. For T²L compatibility a 5.6KΩ (±10%) resistor between output and V_{DD} should be added(3).
4003	I_{OL4}	SERIAL OUT SINKING CURRENT, "1" LEVEL	1.0	2.0		mA	V_{OUT} = 0V
4001/2	V_{OL2}	I/O OUTPUT LINES OUTPUT LOW VOLTAGE	V_{SS}−12	V_{SS}−7.5	V_{SS}−6.5	V	I_{OL2} = 50μA
4003	V_{OL3}	OUTPUT LOW VOLTAGE	V_{SS}−11	V_{SS}−7.5	V_{SS}−6.5	V	I_{OL3} = 10 μA
4001/2	R_{OH2}	OUTPUT RESISTANCE I/O LINES "0" LEVEL		1.2	1.8	KΩ	V_{OUT} = −0.5V
4003	R_{OH3}	PARALLEL OUT PINS OUTPUT RESISTANCE "0" LEVEL		400	750	Ω	V_{OUT} = −0.5V
4003	R_{OH4}	SERIAL OUT OUTPUT RESISTANCE "0" LEVEL		650	1200	Ω	V_{OUT} = −0.5V

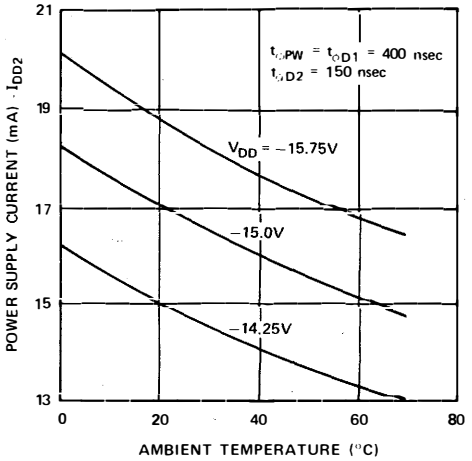
(1) Typical values are for T_A = 25°C and Nominal Supply Voltages.
(2) If non-inverting input option is used, V_{IL} = −6.5 Volts maximum.
(3) For T²L compatibility on the I/O lines the supply voltages should be V_{DD} = −10V ± 5% V_{SS} = +5V ± 5%

Typical D. C. Characteristics

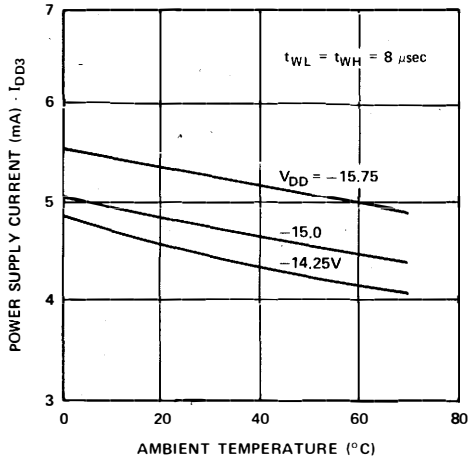
POWER SUPPLY CURRENT VS. TEMPERATURE (4001)



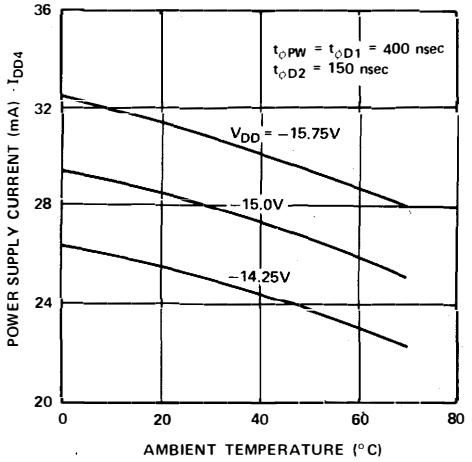
POWER SUPPLY CURRENT VS. TEMPERATURE (4002)



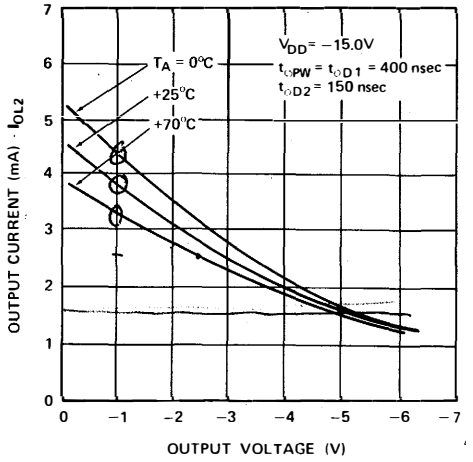
POWER SUPPLY CURRENT VS. TEMPERATURE (4003)



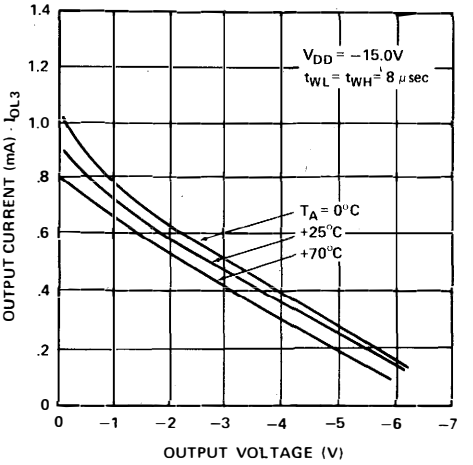
POWER SUPPLY CURRENT VS. TEMPERATURE (4004)



OUTPUT CURRENT VS. OUTPUT VOLTAGE (4001, 4002)



OUTPUT CURRENT VS. OUTPUT VOLTAGE (4003)



4001, 4002, 4004 A.C. Characteristics

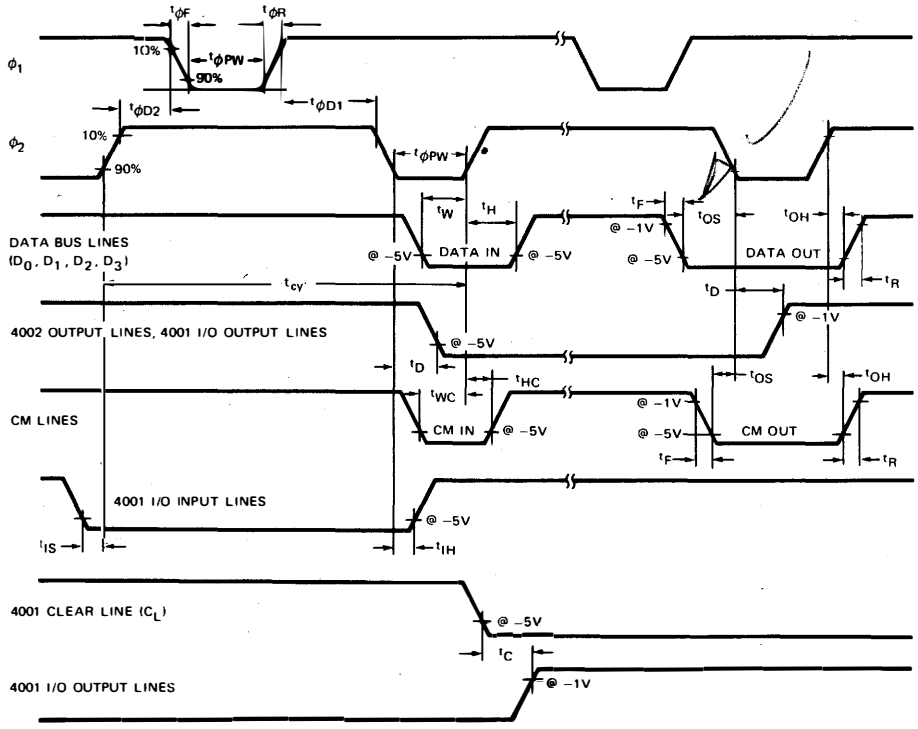
T_A = 0°C to +70°C; V_{DD} = −15V ± 5%, V_{SS} = GND

PROOUCT	SYMBOL	TEST	LIMIT MIN.	LIMIT MAX.	UNIT	CONDITIONS
4001/2/4	t_{cy}	CLOCK PERIOD	1.35	2	μsec	
	$t_{\phi R}$ $t_{\phi F}$	CLOCK RISE AND FALL TIMES		50	nsec	
	$t_{\phi PW}$	CLOCK WIDTH	380	480	nsec	
	$t_{\phi D1}$	CLOCK DELAY FROM ϕ_1 TO ϕ_2	400	550	nsec	
	$t_{\phi D2}$	CLOCK DELAY FROM ϕ_2 TO ϕ_1	150		nsec	
	t_W	DATA-IN WRITE TIME	350		nsec	
	t_H	DATA-IN HOLD TIME	40		nsec	
	t_{OS}(1)	SET TIME FOR DATA OUT, SYNC, CM-ROM,(2) CM-RAM,(2) LINES	0		nsec	C_{OUT} = 500pF for data lines 500pF for SYNC 160pF for CM-ROM 50pF for CM-RAM
	t_{OH}	HOLD TIME FOR DATA OUT, SYNC, CM-ROM, CM-RAM, LINES	50		nsec	C_{OUT} = 20pF
	t_R, t_F	RISE AND FALL TIMES FOR DATA OUT, SYNC, CM-ROM, CM-RAM, LINES		500	nsec	C_{OUT} = 500pF for data lines 500pF for SYNC 160pF for CM-ROM 50pF for CM-RAM
4001/2	t_D	I/O OUTPUT LINES DELAY		600	nsec	C_{OUT} = 20pF
	t_{WC}	CM WRITE TIME	350		nsec	
	t_{HC}	CM HOLD TIME	10		nsec	
4001	t_{IS}	I/O INPUT LINES SET TIME	50		nsec	
	t_{IH}	I/O INPUT LINES HOLD TIME	100		nsec	
	t_C(3)	I/O OUTPUT LINES DELAY ON CLEAR		200	nsec	C_{OUT} = 20pF

NOTES: (1) Data out, SYNC, CM-ROM, and CM-RAM$_i$ lines are clocked out with the trailing edge of the ϕ_2 clock.
(2) The CM-ROM and the selected CM-RAM$_i$ lines are always activated during A_3 time. They are also activated during M_2 time if an I/O and RAM instruction was fetched by the CPU, and during X_2 time if an SRC instruction was fetched by the CPU.
(3) Pin C_L on 4001 is used to asynchronously clear the output flip-flops associated with the I/O lines.

4001, 4002, 4004 Timing Diagram

Outputs with loading conditions specified on A.C. Characteristics table.



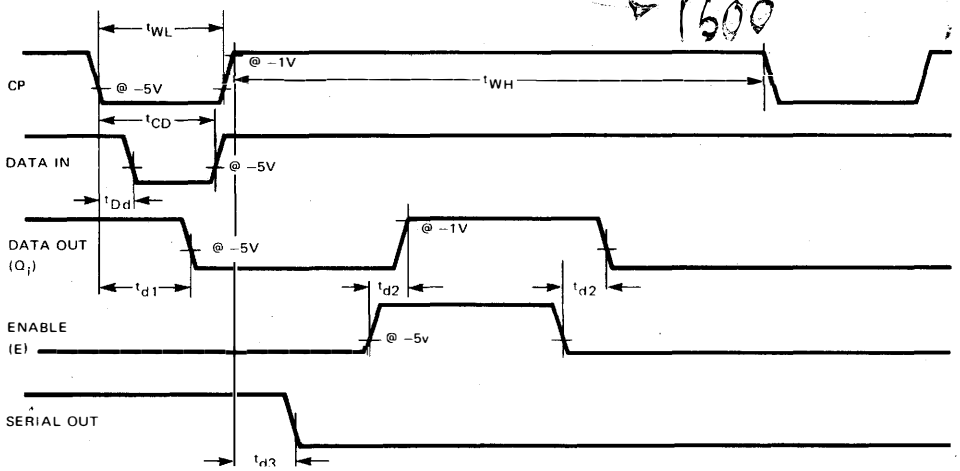
4003 A. C. Characteristics

T_A = 0°C to +70°C; V_{DD} = −15 ± 5%, V_{SS} = GND

SYMBOL	TEST	LIMIT MIN.	LIMIT MAX.	UNIT	CONDITIONS
t_{WL}	CP LOW WIDTH	6	10,000	μsec	
t_{WH}	CP HIGH WIDTH	6	Note (1)	μsec	
t_{CD}	CLOCK-ON TO DATA-OFF TIME	3		μsec	
t_{Dd}	CP TO DATA SET DELAY	Note (2)	250	nsec	
t_{d1}	CP TO DATA OUT DELAY	250	1,750	nsec	
t_{d2}	ENABLE TO DATA OUT DELAY		350	nsec	C_{OUT} = 20 pF
t_{d3}	CP TO SERIAL OUT DELAY	200	1,250	nsec	C_{OUT} = 20 pF

NOTES: (1) t_{WH} can be any time greater than 6 μsec.
(2) Data can occur prior to CP.

4003 Timing Diagram



Capacitance

f = 1 MHz; V_{IN} = 0 V; T_A = 25° C ; Unmeasured Pins Grounded.

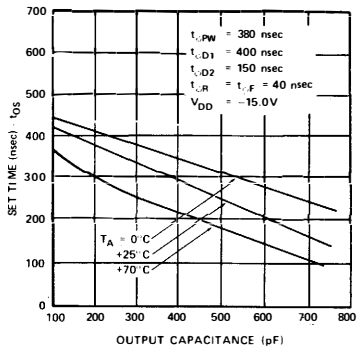
PRODUCT	SYMBOL	TEST	LIMIT (pF) TYP.	LIMIT (pF) MAX.
4001/2/3/4	C_{IN}	INPUT(1) CAPACITANCE	5	10
4001/2	$C_{\phi 1}$, $C_{\phi 2}$	CLOCK INPUT CAPACITANCE	8	15
4004	$C_{\phi 1}$, $C_{\phi 2}$	CLOCK INPUT CAPACITANCE	14	20

PRODUCT	SYMBOL	TEST	LIMIT (pF) TYP.	LIMIT (pF) MAX.
4002/4	C_{D1}	DATA BUS I/O LINES CAPACITANCE	6.5	10
4001	C_{D2}	DATA BUS I/O LINES CAPACITANCE	9.5	15

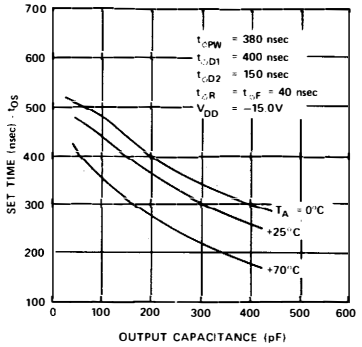
NOTE: (1) Refers to all input pins except data bus I/O and ϕ_1 and ϕ_2.

Typical Load Characteristics

SET TIME VS. OUTPUT CAPACITANCE
(DATA LINES FOR 4001, 4002, 4004 & SYNC FOR 4004)



SET TIME VS. OUTPUT CAPACITANCE
(CM-ROM 4004)



Absolute Maximum Ratings*

Ambient Temperature Under Bias	0°C to +70°C
Storage Temperature	−55°C to +150°C
Input Voltages and Supply Voltage With Respect to V_{SS}	+0.5 to −20V
Power Dissipation	1.0 W

*COMMENT

Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other condition above those indicated in the operational sections of this specification is not implied.

4008, 4009

D. C. and Operating Characteristics

T_A = 0°C to 70°C, $V_{SS}-V_{DD}$ [1] = 15V ± 5%, $t_{\phi PW} = t_{\phi D1}$ = 400ns, $t_{\phi D2}$ = 150ns unless otherwise specified.

Symbol	Parameter	Product	Min.	Typ.[2]	Max.	Unit	Test Conditions
I_{LI}	Input Leakage Current	4008/9			10	μA	$V_{in} = V_{SS}$ −16V, Pins 1-8 (4008) Pins 1-8, 11, 13-15 (4009)
I_{DD}	Average Supply Current	4008		10	20	mA	T_A = 25°C Unloaded
		4009		13	30	mA	
V_{IH}	Input High Voltage	4008/9	V_{SS} −1.5		V_{SS} +0.3	V	
V_{ILC}	Clock Input Low Voltage	4008/9	V_{DD}		V_{SS} −12.5	V	
V_{IL1}	Input Low Voltage (Except I/O)	4008/9	V_{DD}		V_{SS} −5.5	V	Pins 1-6 (4008), Pins 11, 15, 20-23 (4009)
V_{IL2}	I/O Input Low Voltage	4009	V_{DD}		V_{SS} −4.2	V	Pins 1-8, 16-19
V_{OL}	Output Low Voltage	4008/9	V_{SS} −12	V_{SS} −10	V_{SS} −6.5	V	Capacitive Load Only
I_{OL1} [3]	Address Line Sinking Current	4008	8	12		mA	$V_{out} = V_{SS}$
I_{OL2}	Chip Select and F/L Sinking Current	4008	9	13		mA	$V_{out} = V_{SS}$
			1.6	2.5		mA	$V_{out} = V_{SS}$ −4.85V
I_{OL3} [4]	W Output Sinking Current	4008	2.5	5.0		mA	$V_{out} = V_{SS}$
I_{OL4}	Data Bus Sinking Current	4009	9	15		mA	$V_{out} = V_{SS}$: Pins 20-23
I_{OL5}	I/O and Strobe Output Sinking Current	4009	5	12		mA	$V_{out} = V_{SS}$
			1.6	4		mA	$V_{out} = V_{SS}$ −4.85V
R_{OH1}	Output on Resistance	4008		0.6	1.2	kΩ	$V_{out} = V_{SS}$ −0.5V
R_{OH2}	Data Bus Output On Resistance	4009		130	250	Ω	$V_{out} = V_{SS}$ −2V, Pins 20-23
R_{OH3}	I/O and Strobe Output on Resistance	4009		250	1000	Ω	$V_{out} = V_{SS}$ −2V, Pins 9,10,16-19
I_{CF}	Output Clamp Current	4008/9			16	mA	$V_{out} = V_{SS}$ −6V. All outputs on 4008. Pins 9,10,16-19 (4009)

NOTES:

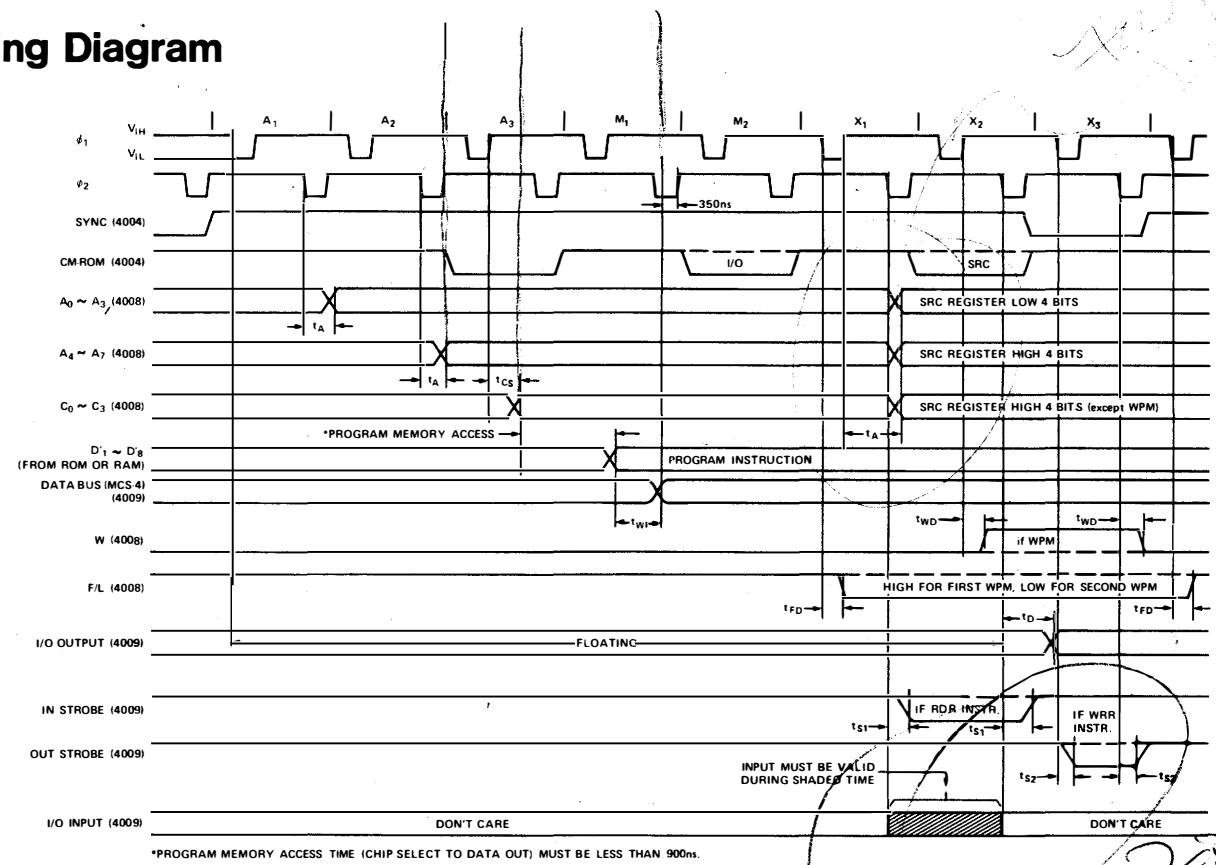
1. For TTL compatibility on the I/O lines, the supply voltages should be V_{SS} = +5V ± 5%, V_{DD} = −10V ± 5%.
2. Typical values are for T_A = 25°C and nominal supply voltages.
3. The address lines will drive a TTL load if a resistor of 470 ohms is connected in series between the address output and the TTL input.
4. A 6.8kohm resistor must be connected between Pin W and V_{DD} for TTL capability.

4008, 4009 A.C. Characteristics

T_A = 0°C to 70°C, V_{SS}–V_{DD} = 15V ± 5%. All clock, sync, CM ROM, data bus, and I/O timing specifications are identical with the 4001 and 4004.

Symbol	Parameter	Product	Limit		Unit	Test Conditions
			Min.	Max.		
t_{CY}	Clock Period	4008/4009	1.35	2.0	μs	
$t_{\phi R}$,$t_{\phi F}$	Clock Rise and Fall Time	4008/4009		50	ns	
$t_{\phi PW}$	Clock Width	4008/4009	380	480	ns	
$t_{\phi D1}$	Clock Delay from ϕ_1 to ϕ_2	4008/4009	400	500	ns	
$t_{\phi D2}$	Clock Delay from ϕ_2 to ϕ_1	4008/4009	150		ns	
t_A	Address to Output Delay at A_1,A_2, X_1	4008		1	μs	C_L = 350pF
t_{CS}	Chip Select Output Delay at A_3	4008		300	ns	C_L = 50pF
t_{WD}	W Output Delay	4008		600	ns	C_L = 100pF
t_{FD}	F/L Output Delay	4008	0.1	1	μs	C_L = 100pF
t_{WI}	Data In Write Time	4009	600		ns	C_L = 200pF on data bus
t_D	I/O Output Delay	4009		1.0	μs	C_L = 300pF
t_{S1}	$\overline{IN}$ Strobe Delay	4009		450	ns	C_L = 50pF
t_{S2}	$\overline{OUT}$ Strobe Delay	4009		1.0	μs	C_L = 50pF

Timing Diagram



*PROGRAM MEMORY ACCESS TIME (CHIP SELECT TO DATA OUT) MUST BE LESS THAN 900ns.

Capacitance

f = 1MHz, V_{IN} = V_{SS}, T_A = 25°C.

Symbol	Parameter	Product	Limit (pF)		Symbol	Parameter	Product	Limit (pF)	
			Typ.	Max.				Typ.	Max.
C_{IN}	Input Capacitance	4008 4009	5 8	10 15	$C_{I/O}$	Data Bus and I/O Capacitance	4008/9	8	10
C_{OUT}	Output Capacitance	4008/9	8	10	C_ϕ	Clock Capacitance	4008/9	12	20

APPLICATION OF THE 4008/4009 IN AN MCS-4 SYSTEM

The standard memory and I/O interface set (4008/4009) provides the complete control functions performed by the 4001 in MCS-4 systems. The 4008/4009 are completely compatible with other members of the MCS-4 family. All activity is still under control of the 4004 CPU. One set of 4008/4009 and several TTL decoders is sufficient to interface to 4k words of program memory, sixteen four-bit input ports and sixteen four-bit output ports.

It should be noted that in any MCS-4 system the program memory is distinct from the read/write data storage (4002 RAM). Using the 4008/4009, programs can now be stored and executed from RAM memory, but this RAM memory is distinct from the 4002 read/write data storage. RAM program memory will be organized in eight bit words and 256 word pages, just like the memory array inside the 4001. Any combination of PROM, ROM, and RAM will be referred to as program memory.

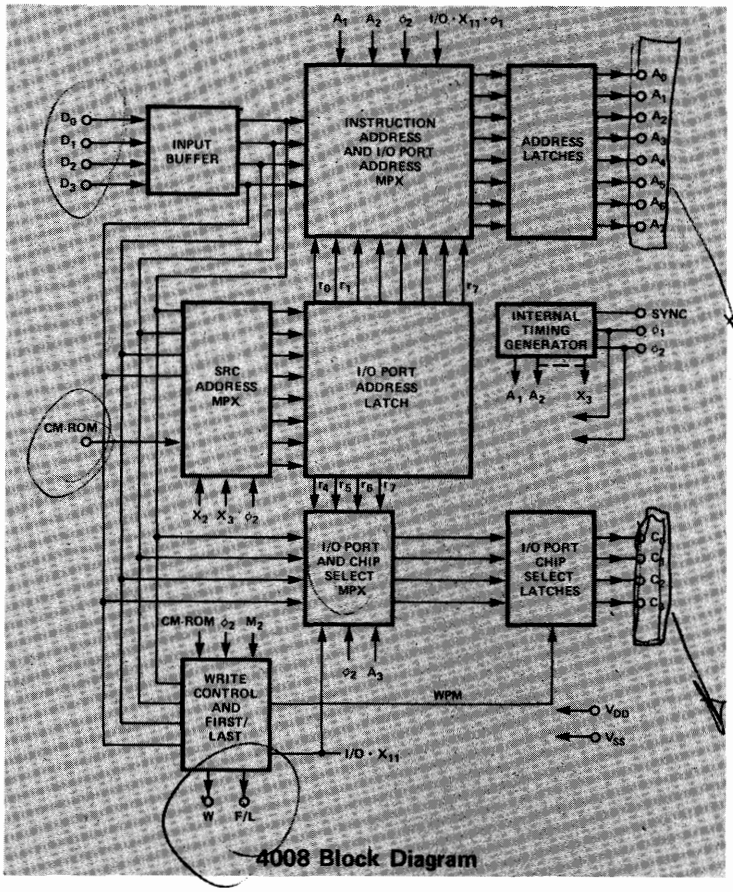
The accompanying diagrams show the internal organization of both the 4008 and 4009.

The 4008 is the address latch chip which interfaces the 4004 to standard PROMs, ROMs and RAMs used for program memory. The 4008 latches the eight bit program address sent out by the CPU during A1 and A2 time. During A3 time it latches the ROM chip number from the 4004. The eight bit program address is then presented at pins A0 through A7 and the four bit chip number (also referred to as page number) is presented at pins C0 through C3. These four bits must be decoded externally and one page of program memory is selected.

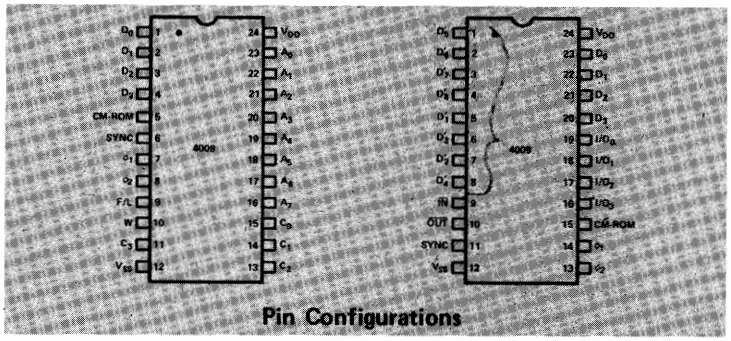
The 4009 then transfers the eight bit instruction from program memory to the 4004 four bits at a time at M1 and M2. The command signal sent by the CPU activates the 4009 and initiates this transfer.

When the CPU executes an SRC (Send Register Control) instruction, the 4008 responds by storing the I/O address in its eight bit SRC register. The content of this SRC register is always transferred to the address lines (A0 through A7) and the chip select lines (C0 through C3) at X1 time. The appropriate I/O port is then selected by decoding the chip select lines. The $\overline{IN}$ and $\overline{OUT}$ lines of the 4009 indicate whether an input or output operation will occur.

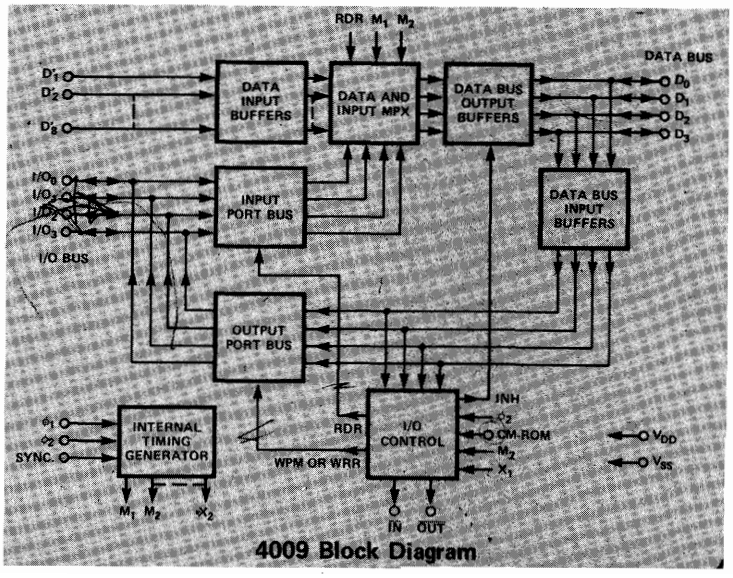
The 4009 is primarily an instruction and I/O transfer device. When the CPU executes an RDR (Read ROM Port) instruction, the 4009 will send an input strobe (pin 9) to enable the selected input port. It also enables I/O input buffers to transfer the input data from the I/O bus to the data bus. When the 4009 interprets a WRR (Write ROM Port) instruction, it transfers output data from the CPU to the I/O bus and sends an output strobe (pin 10) to enable the selected output port.



4008 Block Diagram



Pin Configurations



4009 Block Diagram

A formerly undefined instruction is now used in conjunction with the 4008/4009 to write data into the RAM program memory. This new instruction is called WPM (Write Program Memory – 1110 0011). When an instruction is to be stored in RAM program memory, it is written in two four-bit segments. The F/L signal from the 4008 keeps track of which half is being written. When the CPU executes a WPM instruction, the chip select lines of the 4008 are jammed with "1111". In the system design this should be designated as the RAM channel. The W line on the 4008 is also activated by the WPM instruction. The previously selected SRC address on line A0 through A7 of the 4008 becomes the address of the RAM word being written. By appropriately decoding the chip select lines, the W line, and F/L, the write strobes can be generated for the memory. The F/L line is initially high when power comes on. It then pulses low when every second WPM is executed. A high on the F/L line means that the first four bits are being written, and a low means that the last four bits are being written. The 4009 transfers the segment of the instruction to the I/O bus at X2 of the WPM instruction. The SRC address sent to RAM is only 8 bits. When more than one page of RAM (256 bytes) is being written, an output port must be used to supply additional address lines for higher order addresses.

Definition of Write Program Memory Instruction

Mnemonic: WPM
OPR OPA: 1110 0011
Symbolic:
1111 → $C_3C_2C_1C_0$ of 4008
ACC → $I/O_3I/O_2I/O_1I/O_0$ of 4009
SRC Address → $A_0 - A_7$ of 4008

Description: The chip select lines of the 4008 are forced to "1111" at X1 time and the content of the accumulator is available on the 4009 I/O bus at X2. RAM program memory can be loaded four bits at a time. The previous SRC address is sent out on lines A0 through A7 of 4008.

System Illustrations Using the 4008 and 4009

Four systems are shown where the MCS-4 components are used with standard Intel memory elements as the program memory. Notice that several different approaches to chip select, port decoding, and the I/O elements are shown.

Example 1: Four 1702A PROMs and Four I/O Ports.

This configuration is equivalent to the SIM4-01 system. Four 1702As are used for program storage and four four-bit I/O ports are used. In this case D-type output latches are used and a one of eight decoder (3205) is used to decode both the input and output strobes. Note that the I/O bus is buffered from the outputs. Buffers are needed only when the current sinking requirement on the bus exceeds 1.6mA. In small systems low power TTL could be used and buffers could be avoided.

Example 2: Read/Write Memory for Program Storage.

This example shows only the RAM portion of a system when RAM is used for program memory. Note that the chip selects are tied together in groups of four. The chip selects are gated with the F/L control line for writing only four bits at a time when executing a WPM instruction. They are also gated with the decoding of the chip selects from the 4008 for normal program execution. The 1101 (256 words x 1 bit) is shown. A similar system using the 2102 (1k words x 1 bit) could be developed.

Example 3: Seven 1702A PROMs, one RAM block, and seven I/O Ports.

This example uses a single page of RAM program memory shown in Example 2 in a complete system. In this case the input ports are 8:1 multiplexes which are buffered from the I/O bus by a quad three state buffer. The input port selection is then the function of the multiplexers. The output ports are Intel 3404 latches and the port selection is done using an Intel 3205 decoder.

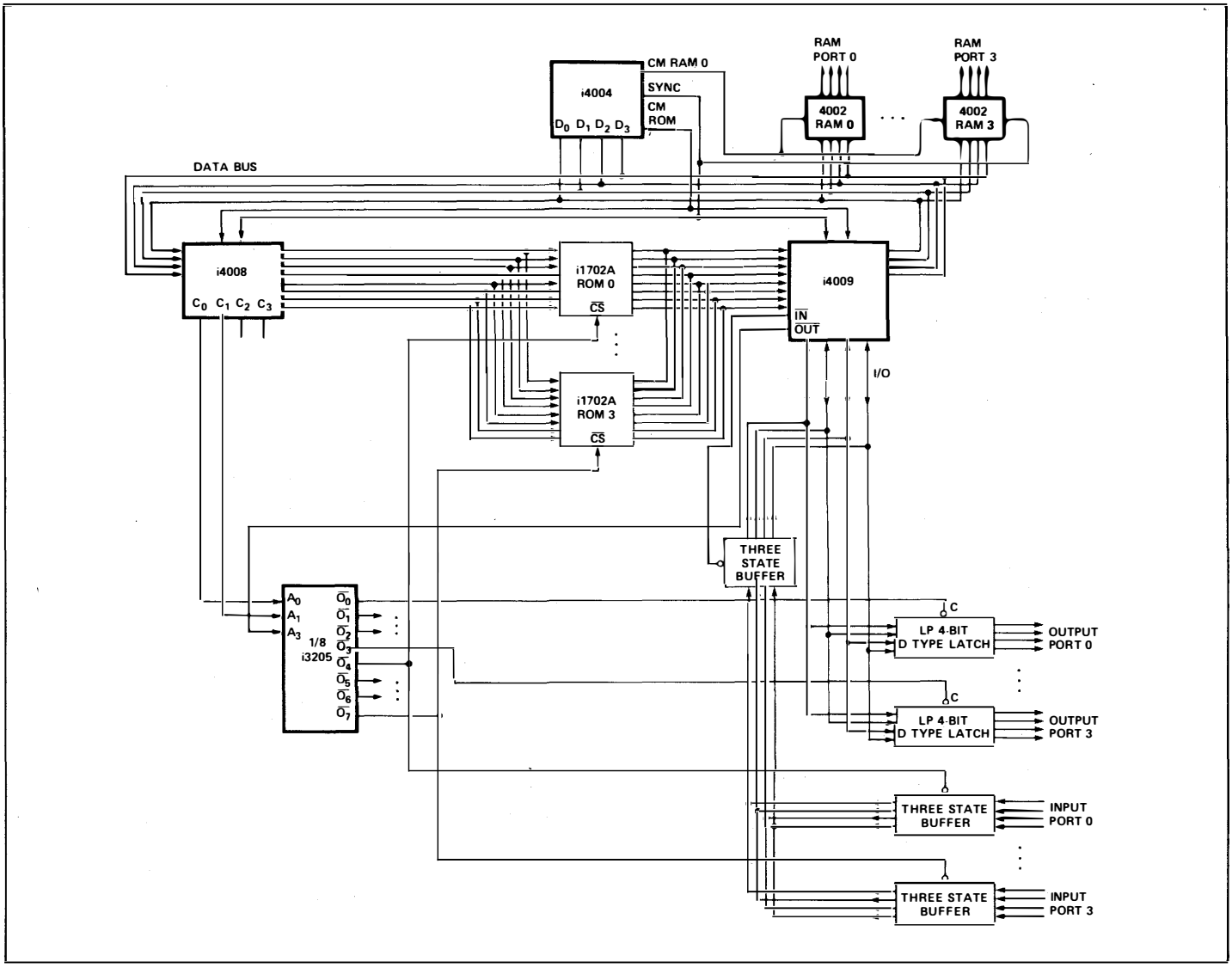
Example 4: Eight 1702A PROMs, eight RAM Blocks, and eight I/O Ports.

Program memory organized with 2k bytes in ROM and 2k bytes in RAM. Each basic RAM block can be organized as in Example 2. When more than one block of RAM is used, the write chip select (WCS) for each RAM block is generated by properly gating chip select 15 with special decoding for page selection. Output port eight is dedicated to this selection function. This is only necessary when the RAM program memory is being written. In this example standard TTL logic elements are used for I/O port selection rather than decoders as shown in previous examples. In this case all input ports are three state buffers.

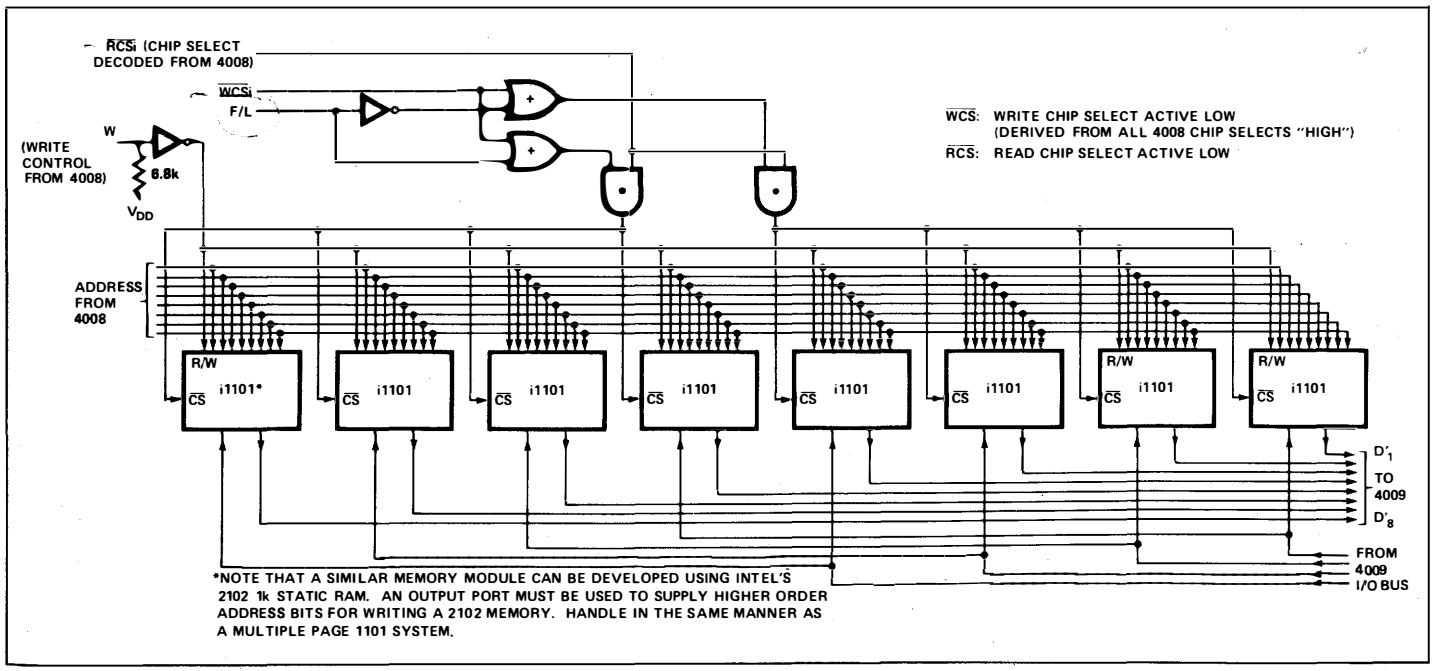
IMPORTANT:

The following differences exist between an MCS-4 system using 4001 program memory and a system using 4008/4009 program memory.

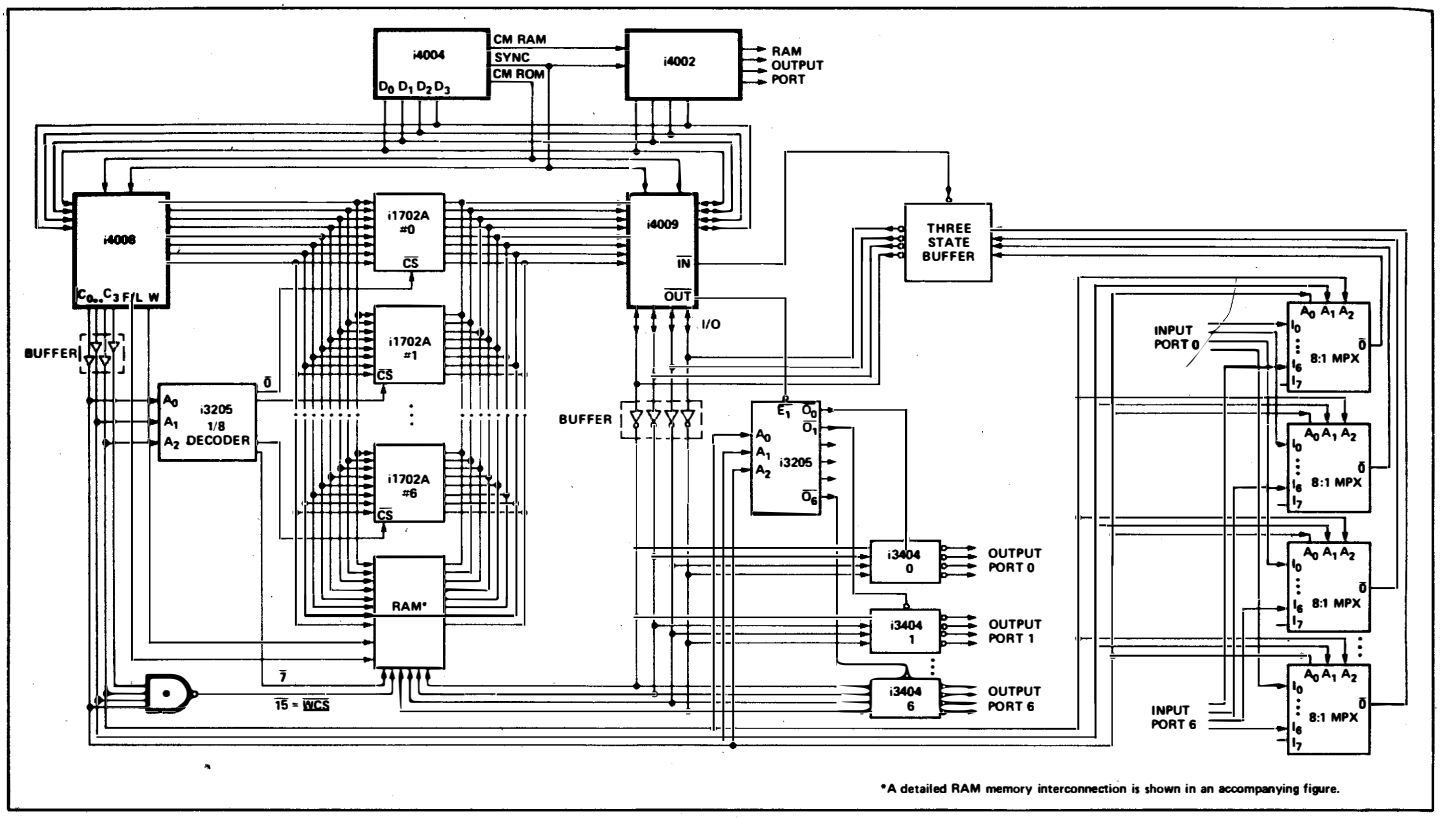
1. For normal operation, 4001 ROMs cannot be used in the same system with 4008/4009.
2. Memory address, memory data, I/O bus, and control lines from both 4008 and 4009 are defined with respect to positive logic. The MCS-4 data and control lines from the 4004 are defined with respect to negative logic. As a result, in program memory used with the 4009, programs should be coded with logic "1" = high level and logic "0" = low level (i.e., NOP = 0000 0000 = NNNN NNNN). Note that programs are defined for the 4001 in terms of negative logic such that NOP = 0000 0000 = PPPP PPPP. Carefully check all tapes submitted for metal mask ROMs to be sure that the correct logic definitions are used.
3. Input and output data from the 4009 I/O bus is defined in terms of positive logic. If these interface devices are used for prototyping a 4001 program memory, care should be taken to be sure that the I/O ports for the 4001s are defined consistent with the 4008/4009 system.
4. An I/O port associated with the 4009 can have lines with both input and output capability. On the 4001 each I/O line may have only a single function, either input or output.
5. The RAM program memory cannot be used as a substitute for the 4002 read/write data storage. They perform distinctly different functions.
6. CM-ROM and CM-RAM_0 cannot be used to control 4002s when CM-ROM is used for 4008/4009 and the WPM instruction is being used. The reason is that the WPM instruction is interpreted as a Write Memory (WRM) by 4002s connected to the same CM line as 4008/4009. CM-RAM_0 in absence of a DCL behaves exactly like CM-ROM.



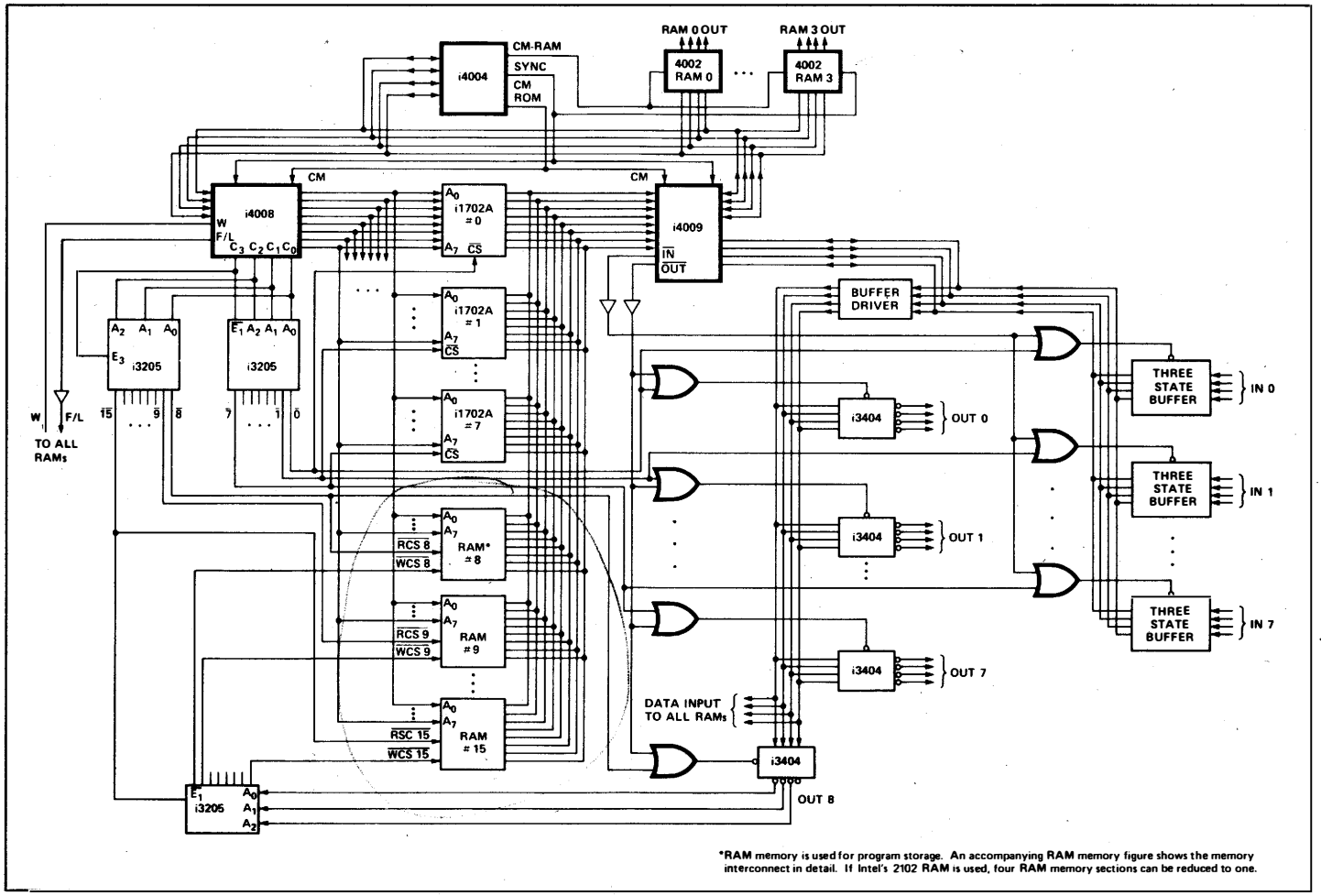
Example 1. Four 1702As and Four I/O Ports (SIM4-01 Equivalent)



Example 2. Read/Write Memory for Program Storage



Example 3. Program Memory with Seven Pages of PROM and One Page of RAM



Example 4. Program Memory with Eight Pages of PROM and Eight Pages of RAM

APPENDIX C

MCS-4 CUSTOM ROM ORDER FORM

SAMPLE

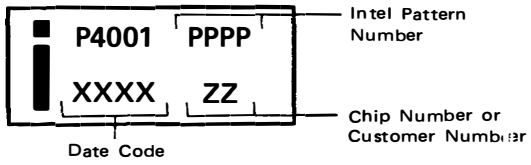
4001 Metal Masked ROM

All custom ROM orders must be submitted on forms provided by Intel. Programming information should be sent in the form of computer punched cards or punched paper tape. In either case, a print-out of the truth table must accompany the order. Refer to Intel's Data Catalog for complete pattern specifications. Alternatively, the accompanying truth table may be used. Additional forms are available from Intel.

	For Intel use only
CUSTOMER _____	S# _____ PPPP _____
P.O. NUMBER _____	STD _____ ZZ _____
	APP _____ DD _____
DATE _____	DATE _____ I/O _____

INTEL STANDARD MARKING

The marking as shown at right must contain the Intel logo, the product type (P4001), the four digit Intel pattern number (PPPP), a date code (XXXX), and the two digit chip number (DD). An optional customer identification number may be substituted for the chip number (ZZ).



Optional Customer Number (Maximum 6 characters or spaces) _____

MASK OPTION SPECIFICATIONS

A. CHIP NUMBER _____ (Must be specified – any number from 0 through 15 – DD)

B. I/O OPTION – Specify the connection numbers for each I/O pin (next page). Examples of some of the possible I/O options are shown below:

EXAMPLES – DESIRED OPTION/CONNECTIONS REQUIRED

1. Non-inverting output – 1 and 3 are connected.
2. Inverting output – 1 and 4 are connected.
3. Non-inverting input (no input resistor) – only 5 is connected.
4. Inverting input (input resistor to V_{SS}) – 2, 6, 7, and 9 are connected.
5. Non-inverting input (input resistor to V_{DD}) – 2, 7, 8, and 10 are connected.
6. If inputs and outputs are mixed on the same port, the pins used as the outputs must have the internal resistor connected to either V_{DD} or V_{SS} (8 and 9 or 8 and 10 must be connected). This is necessary for testing purposes. For example, if there are two inverting inputs (with no input resistor) and 2 non-inverting outputs the connection would be made as follows:

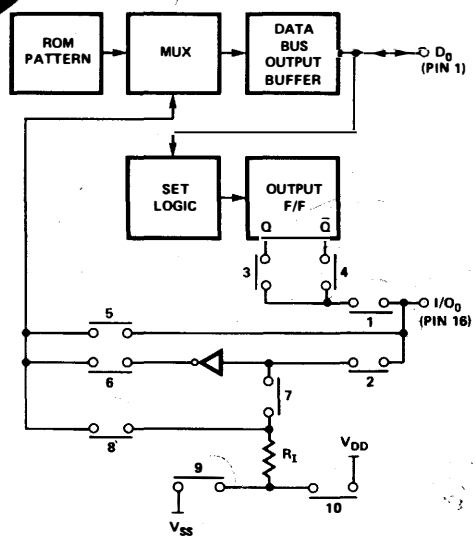
   Inputs – 2 and 6 are connected
   Outputs – 1, 3, 8 and 9 are connected or
   1, 3, 8 and 10 are connected

   If the pins on a port are all inputs or all outputs the internal resistors do not have to be connected.

C. 4001 CUSTOM ROM PATTERN – Programming information should be sent in the form of computer punched cards or punched paper tape. In either case, a print-out of the truth table must accompany the order. Refer to Intel's Data Catalog for complete pattern specifications. Alternatively, the accompanying truth table may be used. Based on the particular customer pattern, the characters should be written as a "P" for a high level output = n-logic "0" (negative logic "0") or an "N" for a low level output = n-logic "1" (negative logic "1").

Note that NOP = BPPPP PPPPF = 0000 0000

4.001 I/O Options

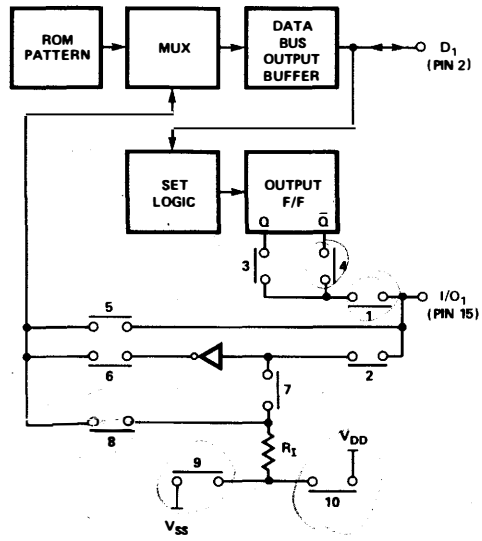


I/O_0 (PIN 16)

CONNECTIONS DESIRED (LIST NUMBERS & CIRCLE CONNECTIONS ON SCHEMATIC)_____

a. For T^2L compatibility on the I/O lines the supply voltages should be $V_{DD} = -10V \pm 5\%$, $V_{SS} = +5V \pm 5\%$

b. If non-inverting input option is used, $V_{IL} = -6.5$ Volts maximum (not TTL).

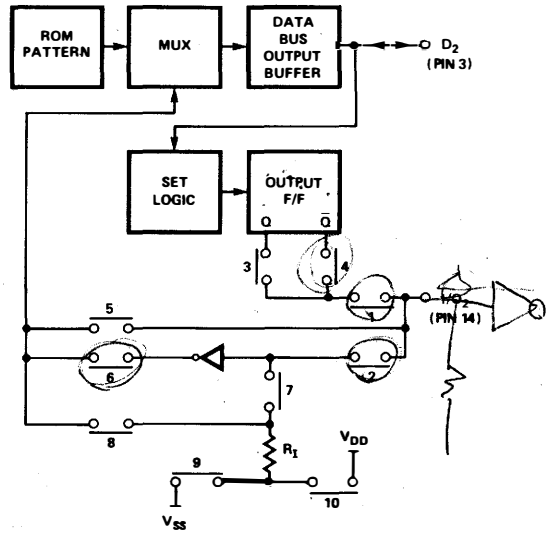


I/O_1 (PIN 15)

CONNECTIONS DESIRED (LIST NUMBERS & CIRCLE CONNECTIONS ON SCHEMATIC)_____

a. For T^2L compatibility on the I/O lines the supply voltages should be $V_{DD} = -10V \pm 5\%$, $V_{SS} = +5V \pm 5\%$

b. If non-inverting input option is used, $V_{IL} = -6.5$ Volts maximum (not TTL).

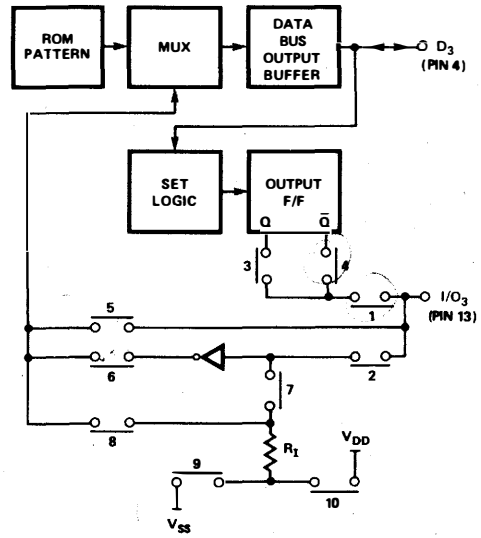


I/O_2 (PIN 14)

CONNECTIONS DESIRED (LIST NUMBERS & CIRCLE CONNECTIONS ON SCHEMATIC)_____

a. For T^2L compatibility on the I/O lines the supply voltages should be $V_{DD} = -10V \pm 5\%$, $V_{SS} = +5V \pm 5\%$

b. If non-inverting input option is used, $V_{IL} = -6.5$ Volts maximum (not TTL).



I/O_3 (PIN 13)

CONNECTIONS DESIRED (LIST NUMBERS & CIRCLE CONNECTIONS ON SCHEMATIC)_____

a. For T^2L compatibility on the I/O lines the supply voltages should be $V_{DD} = -10V \pm 5\%$, $V_{SS} = +5V \pm 5\%$

b. If non-inverting input option is used, $V_{IL} = -6.5$ Volts maximum (not TTL).

4001 CUSTOM ROM TRUTH TABLE

Customer _____

P.O. No. _____

Chip No. _____

Date _____

SAMPLE

The customer truth pattern should be placed in the blue screen area. The white section above the screen area will be used by Intel to verify the customer pattern.

Based on the particular customer pattern, the characters should be written as a "P" for a high level output = n-logic "0" (negative logic "0") or an "N" for a low level output = n-logic "1" (negative logic "1").

Word Number	INSTRUCTION OPR $D_3D_2D_1D_0$	OPA $D_3D_2D_1D_0$	Word Number	INSTRUCTION OPR $D_3D_2D_1D_0$	OPA $D_3D_2D_1D_0$	Word Number	INSTRUCTION OPR $D_3D_2D_1D_0$	OPA $D_3D_2D_1D_0$	Word Number	INSTRUCTION OPR $D_3D_2D_1D_0$	OPA $D_3D_2D_1D_0$
0			32			64			96		
1			33			65			97		
2			34			66			98		
3			35			67			99		
4			36			68			100		
5			37			69			101		
6			38			70			102		
7			39			71			103		
8			40			72			104		
9			41			73			105		
10			42			74			106		
11			43			75			107		
12			44			76			108		
13			45			77			109		
14			46			78			110		
15			47			79			111		
16			48			80			112		
17			49			81			113		
18			50			82			114		
19			51			83			115		
20			52			84			116		
21			53			85			117		
22			54			86			118		
23			55			87			119		
24			56			88			120		
25			57			89			121		
26			58			90			122		
27			59			91			123		
28			60			92			124		
29			61			93			125		
30			62			94			126		
31			63			95			127		

INTEL CORP. 3065 Bowers Avenue, Santa Clara, California 95051 • (408) 246-7501

SAMPLE

4001 CUSTOM ROM TRUTH TABLE

Customer _____

P.O. No. _____

Chip No. _____

Date _____

The customer truth pattern should be placed in the blue screen area. The white section above the screen area will be used by Intel to verify the customer pattern.

Based on the particular customer pattern, the characters should be written as a "P" for a high level output = n-logic "0" (negative logic "0") or an "N" for a low level output = n-logic "1" (negative logic "1").

Word Number	INSTRUCTION OPR $D_3D_2D_1D_0$	INSTRUCTION OPA $D_3D_2D_1D_0$	Word Number	INSTRUCTION OPR $D_3D_2D_1D_0$	INSTRUCTION OPA $D_3D_2D_1D_0$	Word Number	INSTRUCTION OPR $D_3D_2D_1D_0$	INSTRUCTION OPA $D_3D_2D_1D_0$	Word Number	INSTRUCTION OPR $D_3D_2D_1D_0$	INSTRUCTION OPA $D_3D_2D_1D_0$
128			160			192			224		
129			161			193			225		
130			162			194			226		
131			163			195			227		
132			164			196			228		
133			165			197			229		
134			166			198			230		
135			167			199			231		
136			168			200			232		
137			169			201			233		
138			170			202			234		
139			171			203			235		
140			172			204			236		
141			173			205			237		
142			174			206			238		
143			175			207			239		
144			176			208			240		
145			177			209			241		
146			178			210			242		
147			179			211			243		
148			180			212			244		
149			181			213			245		
150			182			214			246		
151			183			215			247		
152			184			216			248		
153			185			217			249		
154			186			218			250		
155			187			219			251		
156			188			220			252		
157			189			221			253		
158			190			222			254		
159			191			223			255		

INTEL CORP. 3065 Bowers Avenue, Santa Clara, California 95051 • (408) 246-7501

APPENDIX D
TELETYPE MODIFICATIONS FOR SIM4-01/SIM4-02

The SIM4-01 and SIM4-02 micro computer systems and associated software have been designed for interface to a model ASR 33 teletype wired in accordance with the following description.

The ASR 33 teletype must receive the following internal modifications and external connections:

Internal Modifications

1. The current source resistor value must be changed to 1450 ohms. This is accomplished by moving a single wire. (See Figures 5 and 6.)
2. A full duplex hook-up must be created internally. This is accomplished by moving two wires on a terminal strip. (See Figures 4 and 6.)
3. The receiver current level must be changed from 60mA to 20mA. This is accomplished by moving a single wire. (See Figures 4 and 6.)
4. A relay circuit must be introduced into the paper tape reader drive circuit. The recommended circuit consists of a relay, a resistor, a capacitor and suitable mounting fixture. An alternate circuit utilizes a thyractor for suppression of inductive spikes. This change requires the assembly of a small "vector" board with the relay circuit on it. It may be mounted in the teletype by using two tapped holes in the mounting plate shown in Figure 1. The relay circuit may then be added without alteration of the existing circuit. (See Figures 2, 3, and 6.) That is, wire "A", to be connected to the brown wire in Figure 2, may be spliced into the brown wire near its connector plug. The "line" and "local" wires must then be connected to the mode switch as shown. Existing reader control circuitry within the teletype need not be altered.

External Connections

1. A two-wire receive loop must be created. This is accomplished by the connection of two wires between the teletype and the "SIM" board in accordance with Figure 6.
2. A two-wire send loop similar to the receive loop must be created. (See Figure 6.)
3. A two-wire tape reader loop connecting the reader control relay to the "SIM" board must be created. (See Figure 6.)

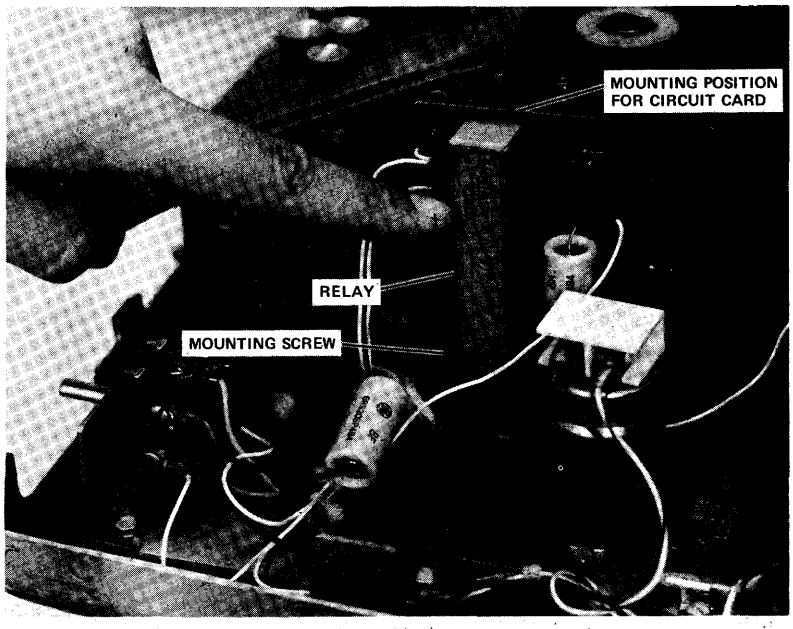


Figure 1. Relay Circuit (Alternate)

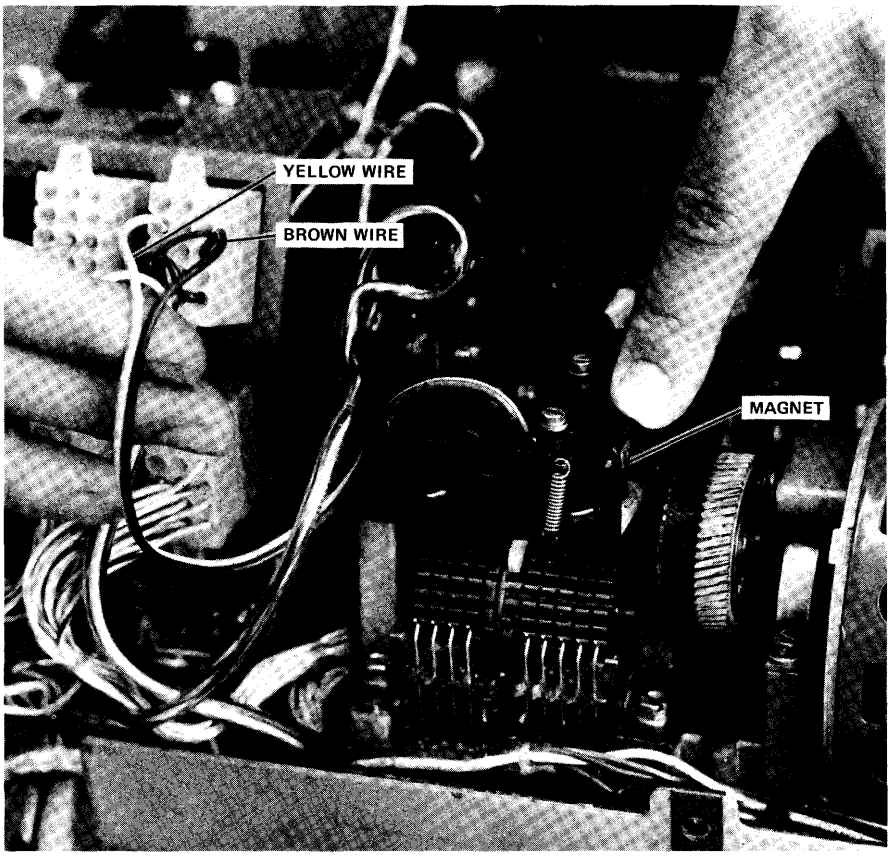


Figure 2. Distributor Trip Magnet

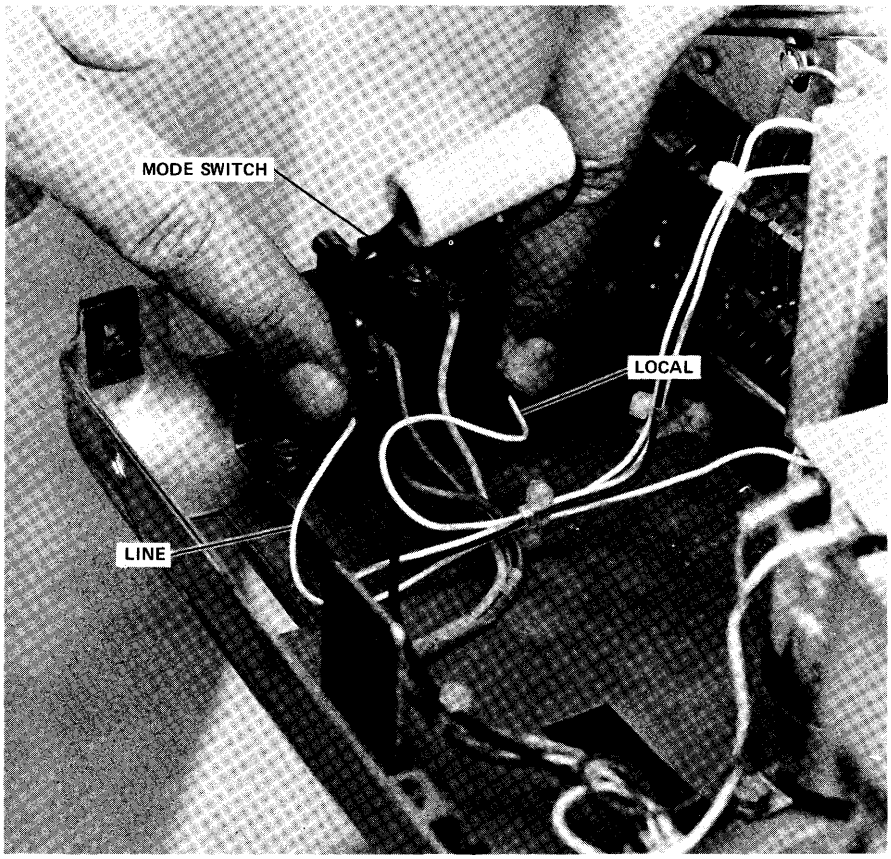


Figure 3. Mode Switch (Rear View)

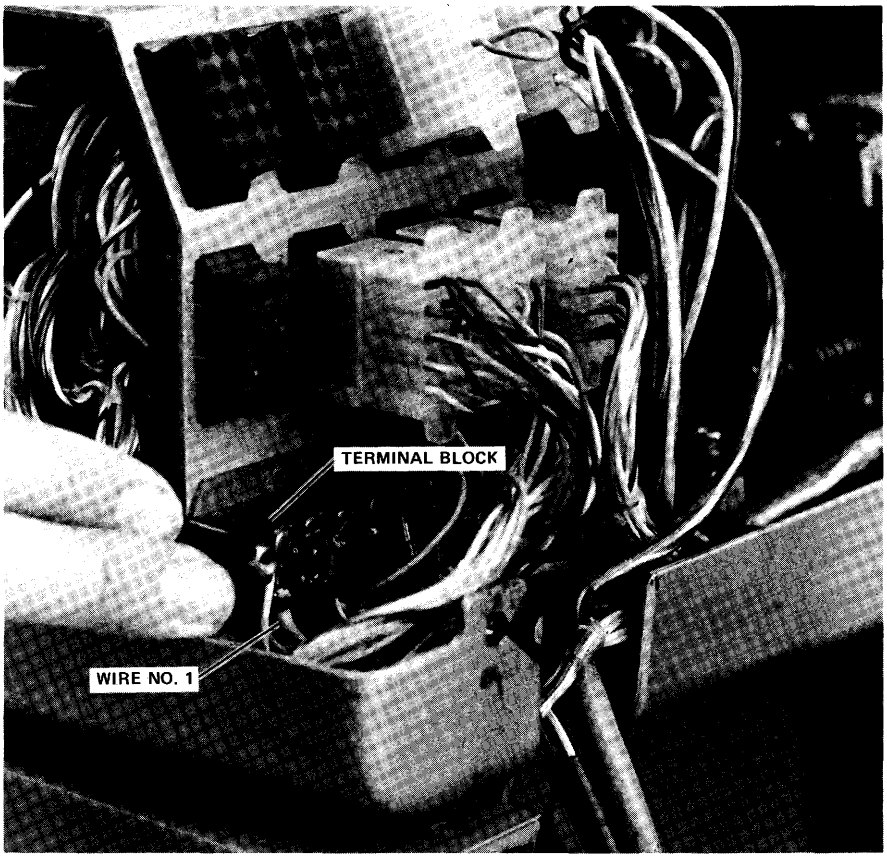


Figure 4. Terminal Block

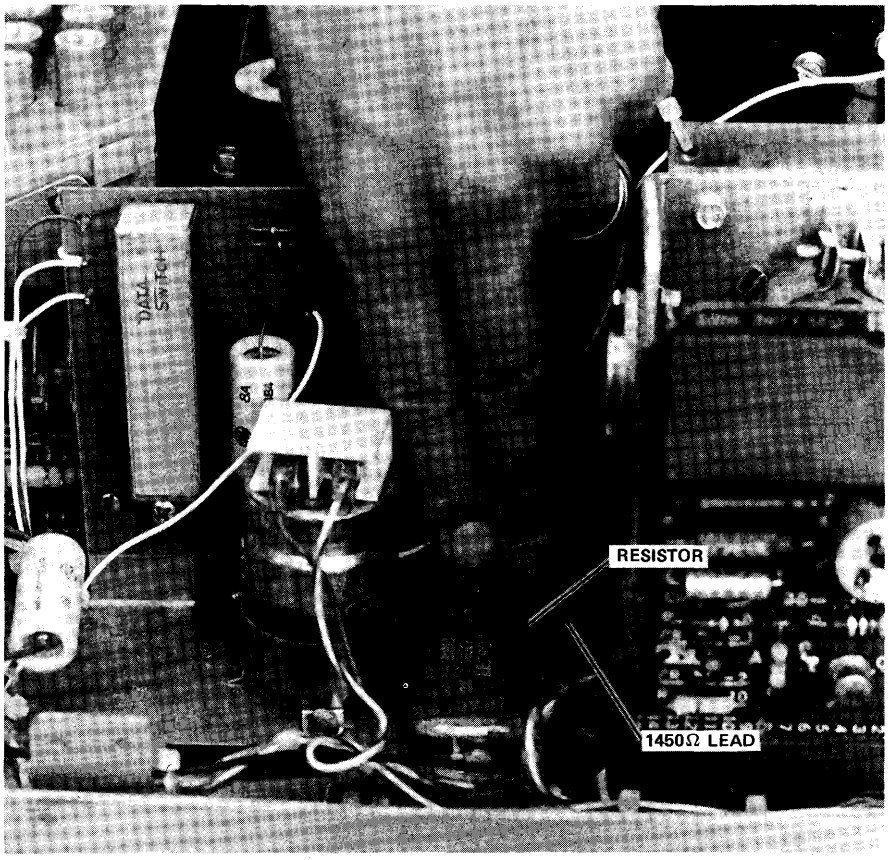


Figure 5. Current Source Resistor

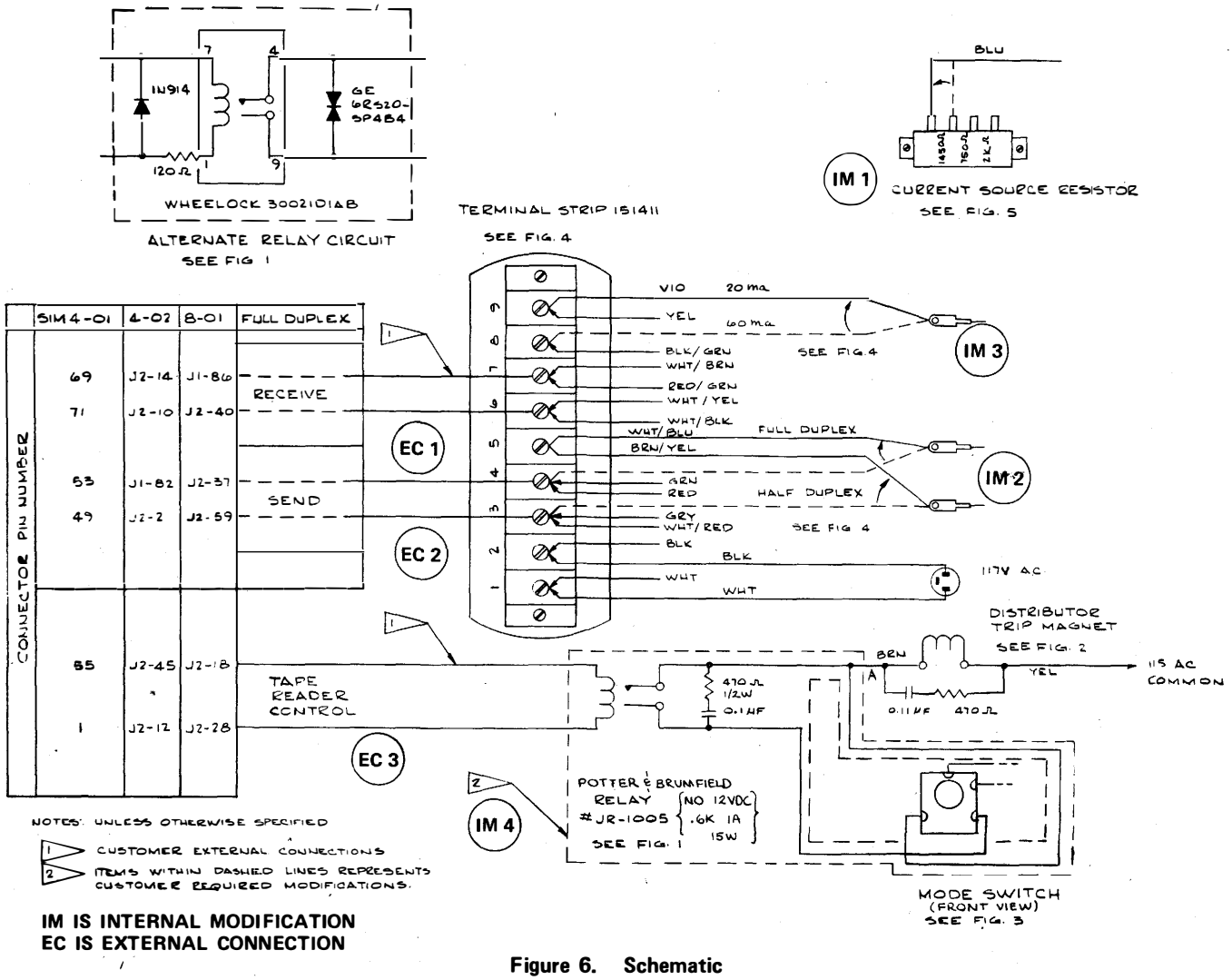


Figure 6. Schematic

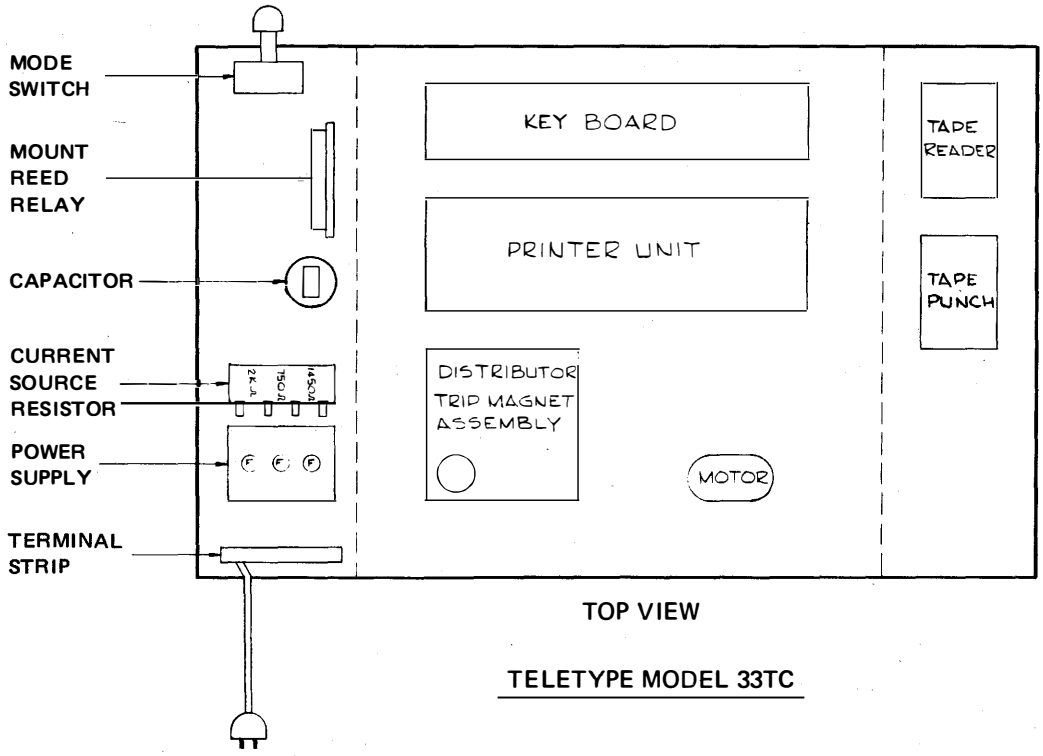


Figure 7. Block Diagram

APPENDIX E
SYSTEM INTERFACE AND CONTROL MODULES

MCB4-10

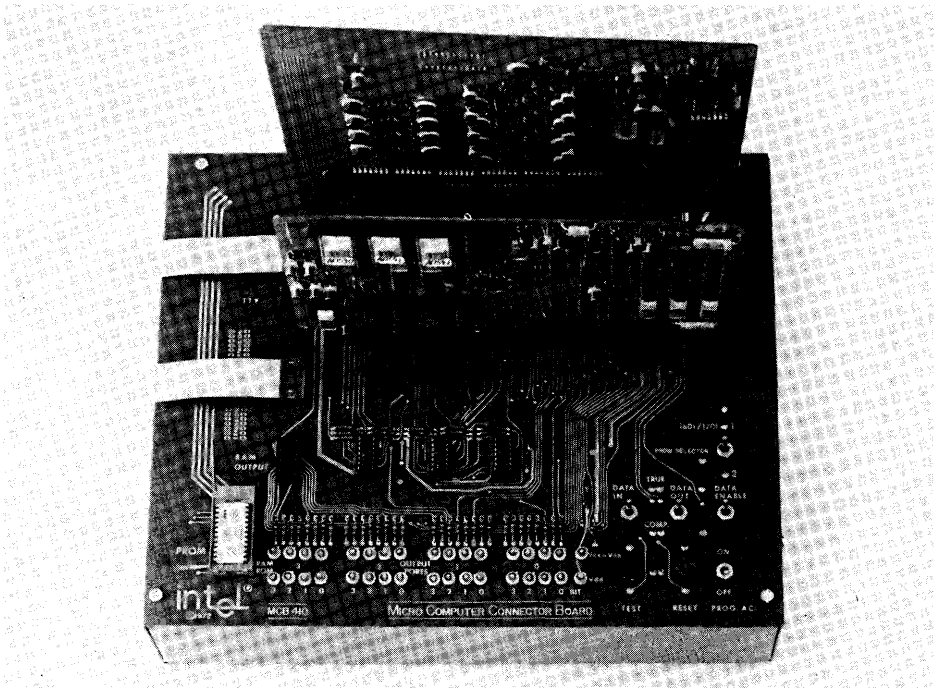
The MCB4-10 is a completely assembled interconnect, display and control switch module which eliminates all hand wiring associated with an MP7-03/SIM4-01 setup. With the additions noted below, it becomes a self-contained system featuring the following:

1. **Automatic PROM Programming** (with SIM4-01, PROM set A0540, A0541, A0543, MP7-03 power supplies, TTY).
2. **General Purpose Micro Processor with I/O and Display** (with SIM4-01, power supplies).
3. **Test System for checkout of PROMs** (with SIM4-01, power supplies).

The MCB4-10 includes the following:

1. All interconnect circuitry necessary to implement the programming system described in section XIII of the "MCS-4 Users Manual".
2. Connectors for the SIM4-01 and MP7-03 boards.
3. A zero insertion force 24-pin socket for PROMs to be programmed. Appropriate connections to the MP7-03 connector are provided.
4. Teletype receive conditioning circuit, transmit source circuit, punch and reader control interface circuits. Access to these signals is provided by a 16 pin socket.
5. Control switches (2) and logic necessary for complementation of programmer input or output data.
6. Breakout and buffering of computer signals to open sockets for ease of access. This includes 16 ROM outputs, 16 ROM inputs, and 16 RAM outputs.
7. SIM4-01 ROM and RAM I/O port binary display using light emitting diodes. This includes 32 bits of display.
8. Data enable control switch which enables the MP7-03 output buffer.
9. A PROM selector switch which facilitates addition of a select function on the MP7-03 board for future use.
10. Two momentary pushbutton switches which drive the "test" and "reset" input lines on the SIM4-01 board.
11. Two transformers, 115Vrms, capacitors, fuse holder and AC input jack wired to develop a raw supply and filtering for development of the programming voltage.
12. A control switch for disabling the programming voltage.
13. Input jacks for applying externally supplied +5V DC and −10V DC to the assembly. (Note: Internal supplies are not included.)

The setup for the PROM programming application requires an MP7-03 (rear) and a SIM4-01 board installed in the MCB4-10. Also shown are flat cables interfaced via two 16-pin DIP sockets to the system.



MCB4-10/MP7-03/SIM4-01 System

1. Micro Processor System

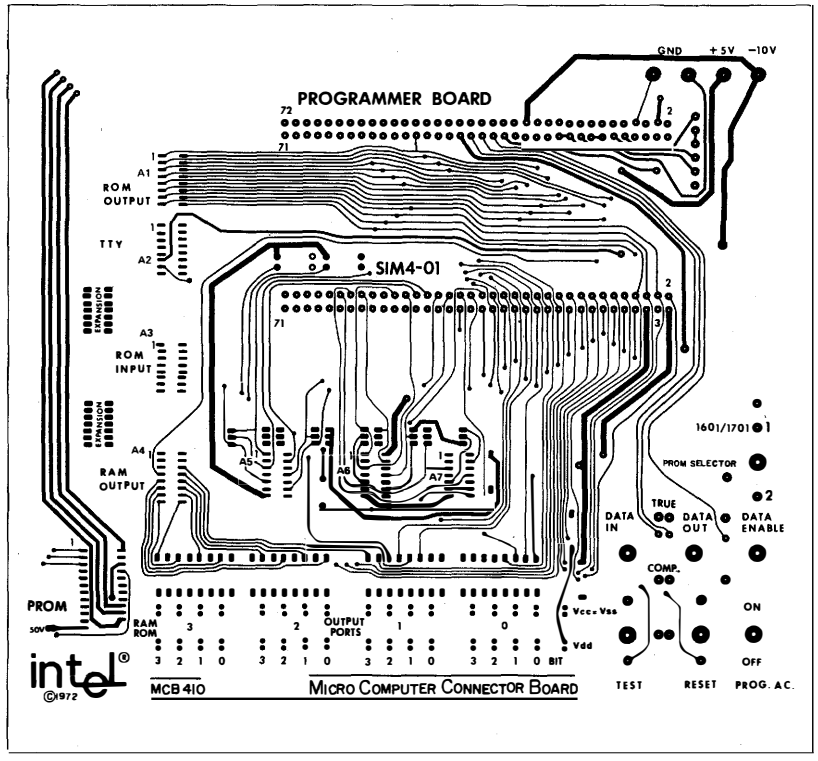
When the MCB4-10 is used as a micro processor, its features, such as the display for the output ports and input ports, may be utilized at the discretion of the user. As an example, consider the testing of SIM4-01 boards loaded with a PROM (1702) containing the following program: read ROM port 0 and ROM port 1, add the two values and store the result at RAM ports 0 and 1. The test could be implemented by connecting 8 switches to the "ROM input" socket. The actual switch circuit would consist of a single pole double throw switch wired with one pole to ground and the wiper wired to the appropriate socket connector pin in accordance with the MCB4-10 schematic (A3-1, A3-16 . . . A3-13). The SIM4-01 is then inserted into the "SIM4-01" connector and a bench supply connected to the +5V DC and −10V DC input jacks. The actual test may now be performed. The reset button is depressed, clearing the system's memories and registers and the program executes. The result appears at the LED display and may be verified for correctness. The display lights of interest are identified on the system's printed circuit board as "OUTPUT PORTS", "RAM 0", "RAM 1", "BITS" 0, 1, 2, and 3.

2. Programming System

Consider the actual programming (in the hardware sense) of the 1702A PROM in the example above. The system can perform this function with the addition of an MP7-03 board inserted into the "PROGRAMMER BOARD" connector. An automatic programming system which allows data entry from a keyboard or papertape, automatic verification, listing of ROM contents, and hands-off programming is provided by the further addition of a SIM4-01 board with three pre-programmed PROMs A0540, A0541, A0543 and a modified teletype. The switches added in the manual set-up are deleted. The teletype modification consists of the addition of simple relay network described by MCS-4 Users Manual. The procedure for programming a 1602A/1702A PROM, then, is as follows:

1. Insert MP7-03 and SIM4-01 boards (SIM4-01 loaded with PROM A0540, A0541, A0543).
2. Connect teletype to "TTY" socket.
3. Connect +5V DC, −10V DC and 115 Vrms.
4. Depress "RESET".
5. Set "PROG.AC" to "ON".
6. Set "DATA ENABLE" to "DATA ENABLE".
7. Set "PROM SELECTOR" to "1601A/1701A".
8. Place teletype in "ON-LINE" mode.
9. Depress "RESET".
10. Insert PROM.
11. Place paper tape in TTY reader and set reader to "START".
12. Type in program command "P" and beginning and ending address.

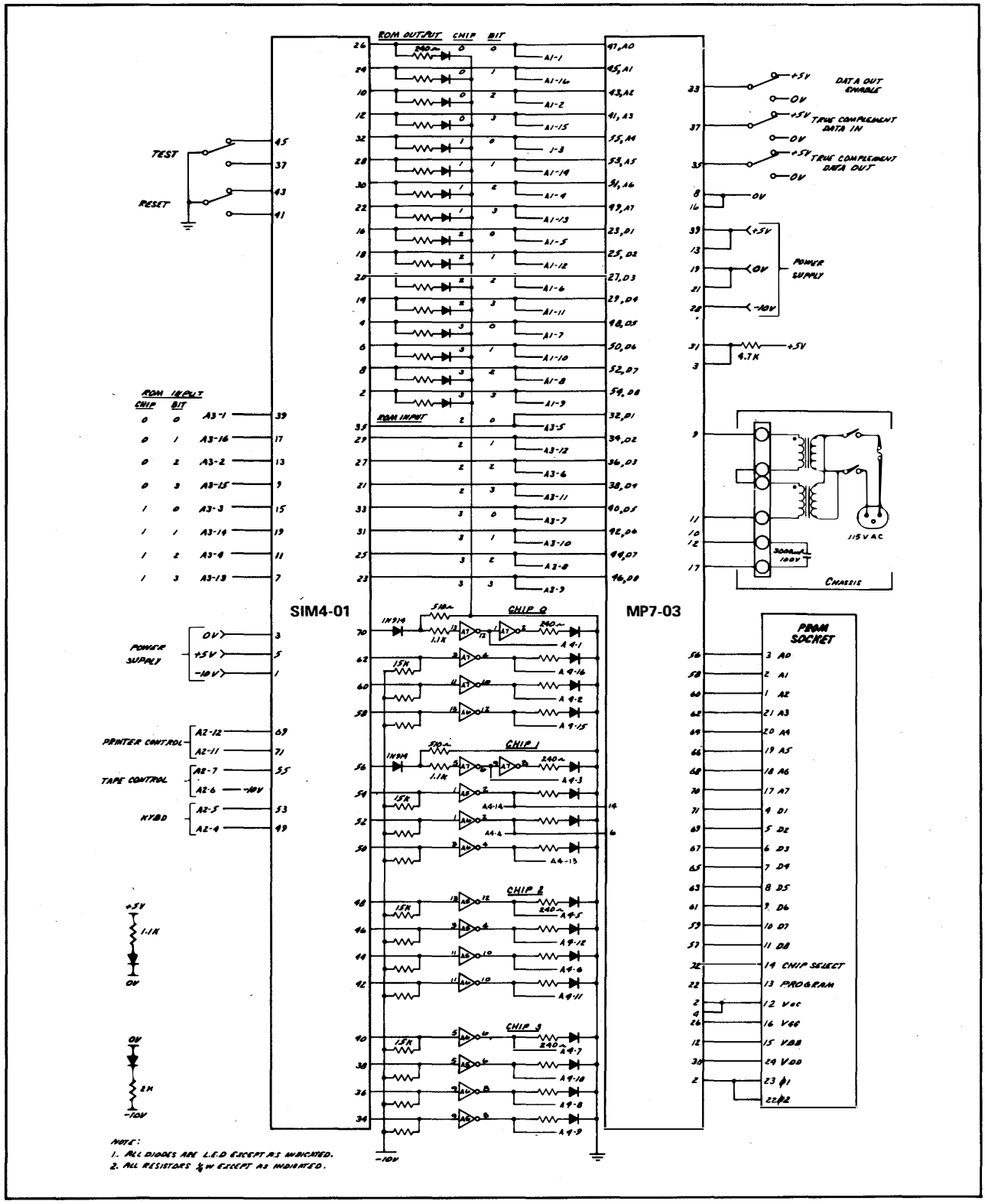
Refer to section XIII for a complete description of the MCS-4 micro computer controlled programming system. The PROM is then automatically programmed and checked for correct content.



MCB4-10 Interconnect and Control Module Printed Circuit Board

NOTE:
1. All signals are defined with respect to negative logic at the dual-in-line I/O socket (i.e., True [n-logic 1] = GND, False [n-logic 0] = +5V).
2. 1 TTL Load/Drive = 1.6mA @ 4V.

PORT	LOGIC	COMPATIBILITY
ROM Input Port	True	TTL In
ROM Output Port	False	TTL Out
RAM Output Port	False	TTL Out



MCB4-10 Schematic

CAUTION:
Permanent damage may result to MP7-03 board and PROM to be programmed if the DC POWER is turned OFF BEFORE the PRGM AC is turned off. The SIM4-01 and MP7-03 should never be inserted into their respective sockets with either the DC or AC power applied.
APPLY DC POWER BEFORE AC POWER AND REMOVE AC POWER BEFORE DC POWER.

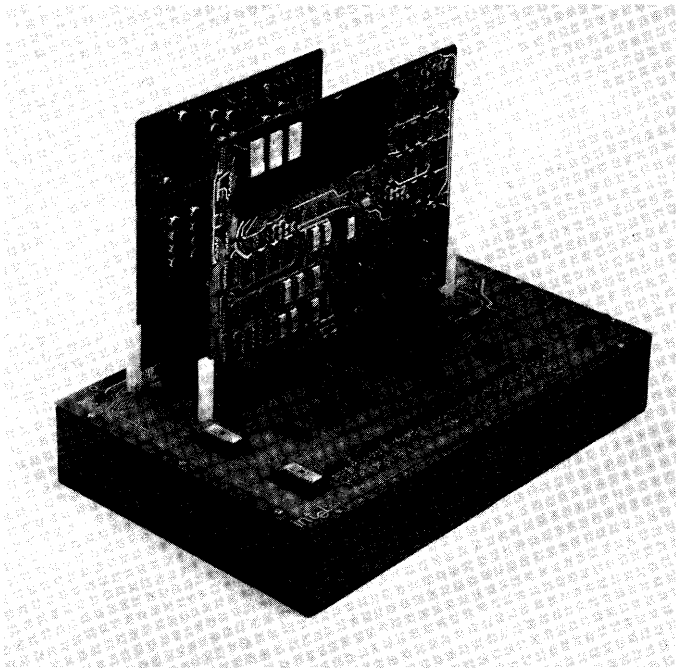
The MCB4-20 is a completely assembled interconnect, display and control switch assembly which eliminates all hand wiring associated with an MP7-03/SIM4-02 setup. With the additions noted below, it becomes a self-contained system featuring the following:

1. **Automatic PROM Programming** (with SIM4-02, PROM set A0540, A0541, A0543, MP7-03, power supplies, TTY).
2. **Automatic PROM Duplicating/Comparing** (with SIM4-02, MP7-03, A0544 PROM Program, power supplies).
3. **General Purpose Micro Processor With I/O and Display** (with SIM4-02, power supplies).
4. **Test System for checkout of PROM Programs** (with SIM4-02, power supplies).

The MCB4-20 includes the following:

1. All interconnect circuitry necessary to implement the programming system described in paragraph XIII of the MCS-4 Users Manual.
2. Connectors for the SIM4-02 and MP7-03 boards.
3. Two zero insertion force 24-pin sockets for PROMs. One socket for the "PROM to be programmed", one socket for a "Duplicating REF PROM", and appropriate connections to the MP7-03 connector.
4. Teletype receive conditioning circuit, transmit source circuit, punch and reader control interface circuits (on the SIM4-02) and a TTY connect/disconnect switch. Access to these signals is provided by a 16 pin dip socket labeled "TTY" socket (J4). Flat cable is provided for the connector to TTY.
5. Two control switches for complementing programmer input or output data.
6. Eleven 16 pin DIP sockets provide easy access to SIM4-02 input, output and miscellaneous control signals. This includes 32 ROM inputs (8 ports x 4 bits), 32 ROM outputs (8 ports x 4 bits), and 64 RAM outputs (16 ports x 4 bits).
7. SIM4-02 ROM and RAM output port binary display using light emitting diodes. This includes 32 bits of ROM output display and 32 bits of RAM output display. The additional 32 bits of RAM output are not displayed but brought out to 16 pin sockets J6 and J7.
8. Data out control switch enables or disables the data from the MP7-03 to the SIM4-02 input ports or provides CPU software control of data out when in the duplicating position.
9. A DC power control switch which connects the external +5V and −10V power supplies to the SIM4-02 and MP7-03.
10. Two momentary pushbutton switches which drive the "test" and "reset" input lines on the SIM4-02 board.
11. Two transformers, a switch for 115V AC/220V AC, capacitor, fuse holder and AC input jack wired to develop the unregulated 80V DC which in turn is regulated on MP7-03 to 47V DC programming voltage.
12. A PRGM AC switch which controls the programming AC voltage.
13. Input jacks for applying externally supplied +5V DC and −10V DC to the assembly. (Note: Internal supplies are not included, 5V @ 4A and −10V @ 2A supplies required [worst case].)

The setup for the PROM Programming application is shown below. The MP7-03 (rear) and the SIM4-02 board are installed in the MCB4-20.



MCB4-20/MP7-03/SIM4-02 System

1. Micro Processor System

When the MCB4-20 is used as a micro processor, its features, such as the output ports (with displays) and input ports, may be utilized at the discretion of the user. As an example, consider the testing of SIM4-02 boards loaded with a PROM containing the following program: read ROM port 0 and ROM port 1, add the two values and store the result at RAM ports 0 and 1. The test could be implemented by connecting eight switches to the "ROM input" socket. The actual switch circuit would consist of a single pole double throw switch wired with one pole to ground and the wiper wired to the appropriate socket connector pin in accordance with the MCB4-20 schematic (GND on input port equals a logic 1). The SIM4-02 is then inserted into the "SIM4-02" connector and a bench supply connected to the +5V DC and −10V DC input jacks. The actual test may now be performed. The DC power switch is turned on and the reset button is depressed, clearing the system's memories and registers. The program begins to execute. The result appears at the LED display and may be verified for correctness. The LED displays of interest are identified on the system's printed circuit board as "OUTPUT PORTS", "RAM 0", "RAM 1", "BITS" 0, 1, 2, and 3.

2. Programming System

Consider the actual programming (in the hardware sense) of the 1702A PROM in the example above. The system can perform this function with the addition of an MP7-03 board inserted into the "MP7-03" connector. An automatic programming system which allows data entry from a keyboard or papertape, automatic verification, listing of ROM contents, and hands-off programming is provided by the further addition of a SIM4-02 board with three pre-programmed PROMs A0540, A0541, A0543, and a modified teletype. The teletype modification consists of the addition of simply relay network described by MCS-4 Users Manual or TTY application note. The procedure for programming a PROM, then, is as follows:

1. Insert MP7-03 and SIM4-02 boards (SIM4-02 loaded with PROM A0540, A0541, A0543).
2. Connect teletype to "TTY" socket using the flat cable provided.
3. Connect +5V DC, −10V DC and 115V AC/220V AC.
4. Set "DC Power" switch to "ON" (See Caution below).
5. Depress "RESET".
6. Set "PRGM AC" to "ON".
7. Set Data Out to "Enable".
8. Set "Data In" and "Data Out", "True/Compl", to desired position.
9. Set TTY switch to "TTY Connect".
10. Place teletype in "ON-LINE" mode.
11. Depress "RESET".
12. Insert an erased 1702A PROM into the "PROM to be Programmed" 24 pin zero force socket.
13. Place paper tape in TTY reader and set reader to "START".
14. Type in program comman "P" and beginning and ending addresses.

The steps above are fully described in section XIII of the MCS-4 Users Manual. The PROM is then automatically programmed and checked for correct content.

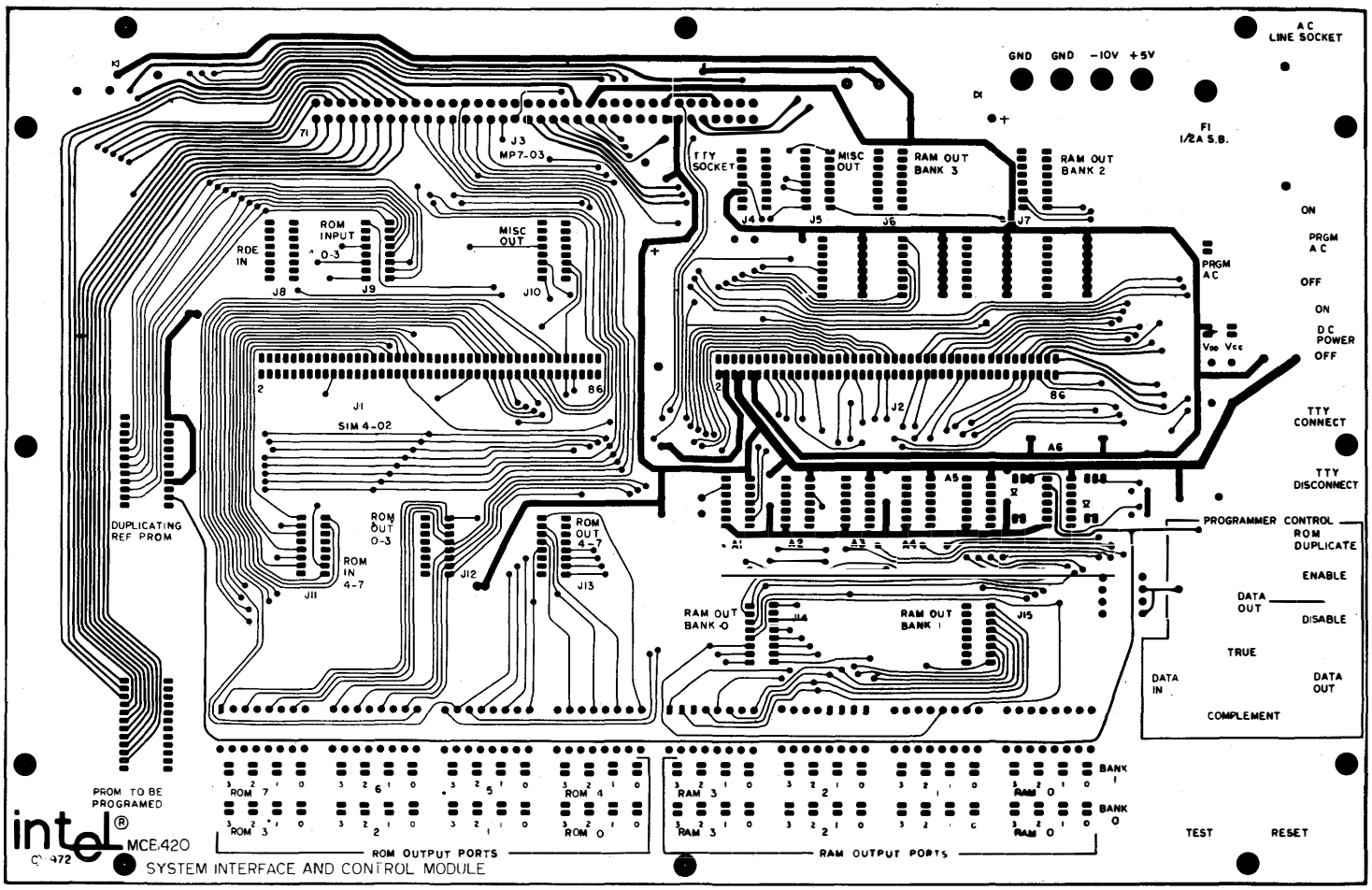
CAUTION:
Permanent damage may result to MP7-03 board and PROM to be programmed if the DC POWER is turned OFF BEFORE the PRGM AC is turned off. The SIM4-02 and MP7-03 should never be inserted into their respective sockets with either the DC or AC power applied.
APPLY DC POWER BEFORE AC POWER AND REMOVE AC POWER BEFORE DC POWER.

3. Automatic PROM Duplicating/Comparing

The MCB4-20 may be used to duplicate or compare 1702/1702A PROMs by the use of the SIM4-02, MP7-03, external power supplies and A0544 PROM. The following procedure should be followed for duplicating or comparing 1702/1702A PROMs.

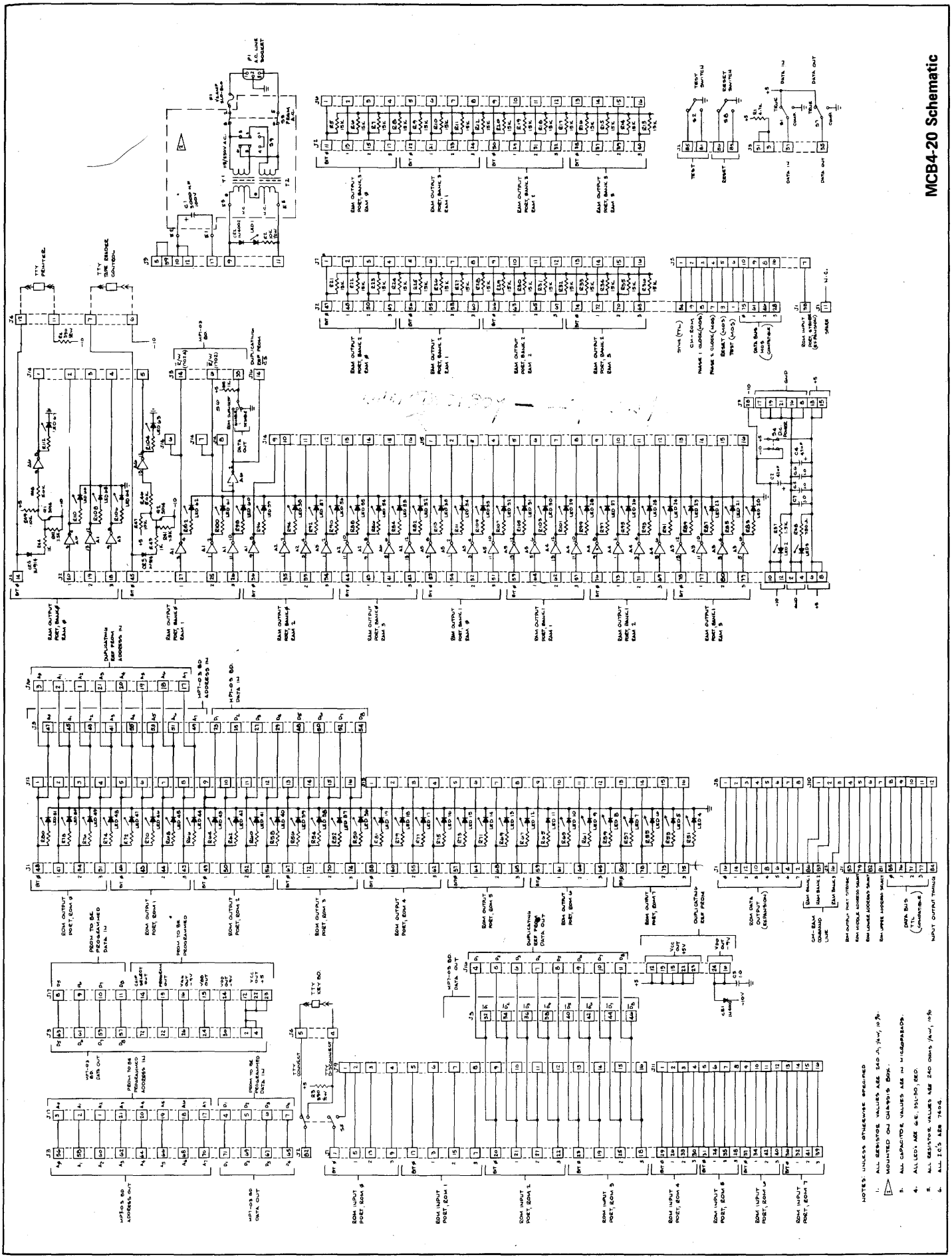
1. Insert SIM4-02 into the MCB4-20.
2. Install A0544 PROM in SIM4-02 PROM socket 0.
3. Insert MP7-03 into the MCB4-20.
4. Connect external power supplies and AC power cord.
5. Turn "DC Power" switch on (See Caution note).
6. Press "Rest" button.

7. Provide a ground potential to "ROM Input (0-3)" socket J9; pin number in accordance with desired operation:
   a. Duplicate 1702A PROMs: GND J9-5
   b. Duplicate 1702 PROMs: GND J9-6
   c. Compare 1702/1702A PROMs: GND J9-7
8. Set "Data Out" control switch to "ROM Duplicate" (for duplicate and compare).
9. Set both "Data In" and "Data Out" switches to either "True" (for exact duplicate) or "Complement" (for complement duplicate). For compare, always set the switches to "True".
10. Set TTY switch to "TTY Disconnect".
11. Press "Reset" button.
12. Turn "PRGM AC" switch "On" (for duplicating ONLY).
13. Insert REF PROM into "Duplicating REF PROM" socket and other PROM in "PROM to be Programmed" socket.
14. Press "Test" button. BANK 0, RAM 0, Bit 1 will light indicating "Start".
15. a. If an error occurs during duplicating or comparing, BANK 0 RAM 0, Bit 2 will light. To continue, press "Test", otherwise press "Reset" and remove "PROM to be Programmed". During duplicating or comparing, four passes are made before claiming an error. DURING DUPLICATE, PROMs ARE AUTOMATICALLY COMPARED AFTER EACH LOCATION IS PROGRAMMED.
    b. If the program finishes, Bank 0, RAM 0, Bit 3 will light. Remove "PROM to be Programmed".
16. Remove REF PROM from socket.
17. Turn "PRGM AC" switch Off.
18. Turn "DC Power" switch Off.



NOTE: 1. All signals are defined with respect to negative logic at the dual-in-line I/O socket (i.e., True [n-logic 1] = GND, False [n-logic 0] = +5V).
2. 1 TTL Load/Drive = 1.6mA @ .4V.

PORT	LOGIC	COMPATIBILITY
ROM Input Port 0-7	True	TTL In
ROM Output Port 0-7	False	TTL Out
RAM Output Port 0-3, Bank 0-1	False	TTL Out
RAM Output Port 0-3, Bank 2-3	True	TTL Out



MCB4-20 Schematic

The following table lists the indicators used in the A0544 PROM Duplicator/ Comparator Program to provide pertinent status, control, address, data in and data out information:

Bank Ø, RAM Ø, Bit 1:	"Start"	Status
Bank Ø, RAM Ø, Bit 2:	"Error"	
Bank Ø, RAM Ø, Bit 3:	"Finish"	
Bank Ø, RAM 1, Bit 1:	1702A $\overline{R}$/W Control	Control
Bank Ø, RAM 1, Bit 2:	1702 $\overline{R}$/W Control	
Bank Ø, RAM 1, Bit 3:	Prom DE/REF ROM CS	
Bank Ø, RAM 2, Bit Ø:	D1	PROM to be PRGM Data Out
Bank Ø, RAM 2, Bit 1:	D2	
Bank Ø, RAM 2, Bit 2:	D3	
Bank Ø, RAM 2, Bit 3:	D4	
Bank Ø, RAM 3, Bit Ø:	D5	
Bank Ø, RAM 3, Bit 1:	D6	
Bank Ø, RAM 3, Bit 2:	D7	
Bank Ø, RAM 3, Bit 3:	D8	
ROM Ø, Bit Ø:	AØ	PROM to be PRGM/REF ROM Address
ROM Ø, Bit 1:	A1	
ROM Ø, Bit 2:	A2	
ROM Ø, Bit 3:	A3	
ROM 1, Bit Ø:	A4	
ROM 1, Bit 1:	A5	
ROM 1, Bit 2:	A6	
ROM 1, Bit 3:	A7	
ROM 2, Bit Ø:	D1	PROM to be PRGM Data in/(REF ROM Data Out)
ROM 2, Bit 1:	D2	
ROM 2, Bit 2:	D3	
ROM 2, Bit 3:	D4	
ROM 3, Bit Ø:	D5	
ROM 3, Bit 1:	D6	
ROM 3, Bit 2:	D7	
ROM 3, Bit 3:	D8	

The complete program is shown in Appendix H.

APPENDIX F

SIM4 HARDWARE ASSEMBLER for SIM4-01 or SIM4-02

INTRODUCTION

The SIM4 Hardware assembler is a program stored in Intel PROMs A0740, A0741, A0742 and A0743 which translates a symbolic assembly language into bit patterns suitable for MCS-4 control storage programming. It operates on the SIM4-01 or the SIM4-02 micro computer system with at least 3 RAMs and an ASR-33 teletype. A block diagram is given in figure 1.1.

The assembler accepts input source text from the teletype keyboard or paper tape reader on each of two required passes. A name table and source listing are created on the first pass. On the second pass, the source text is reread and a programming paper tape and associated listing are generated. The programming tape is suitable for programming of the Intel 1702A erasable PROM (using the MP7-03 programmer system) or for the Intel 4001 metal mask ROM.

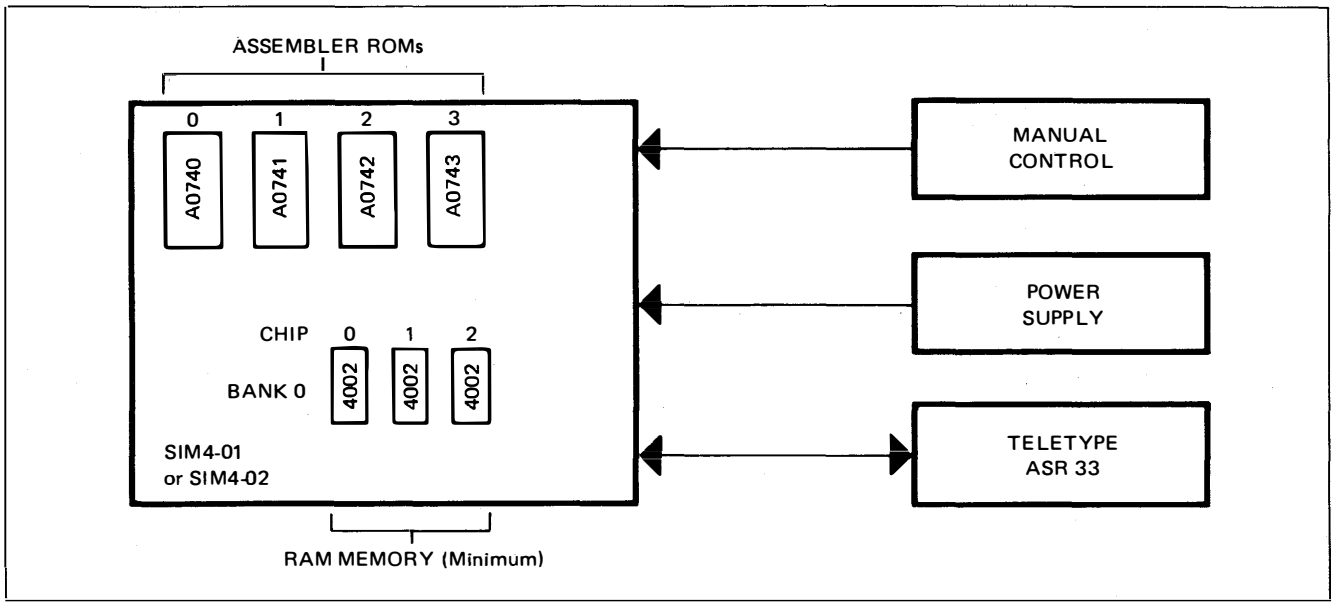


Figure 1.1. Assembler Hardware Block Diagram

DESCRIPTION

Assembly Passes

During pass 1, the assembler constructs a name table from the source text and generates a listing. The source text entries are prompted by an address location printed on each line of the listing. Paper tape reader on/off controls are issued by the assembler during the prompting.

Diagnostics are performed during pass 1. Errors such as duplicated labels, name table overflows, and unrecognized instruction mnemonics are flagged. Operator intervention of the reader operation and subsequent keyboard entries can be used to edit an erroneous source entry.

A programming tape and a listing of its contents are created during pass 2. Entry of the source tape and any editing during pass 1 must be repeated. The assembler decodes the instruction mnemonics, searches the name table for addresses, and forms binary representations of the machine instructions. Simultaneously, it controls the read operation, punches and lists the object tape and executes further diagnostics. The diagnostics will flag errors such as unrecognized instruction mnemonics, undefined names and off page references.

Operating Procedures

Two normal modes of operation are possible with the assembler. A source tape may be prepared in advance, off-line, using the perforator backspace and rubout character to correct minor errors. This tape may be fed, with the reader, to the assembler twice, with the perforator turned off during pass 1, and on during pass 2 (if a ROM programmer tape is desired).

Alternatively, a good teletypist may type the source program directly in on-line, with the perforator turned on, using the punched tape to feed the source back in on pass 2. This tape may also be edited manually off-line for purposes of updating the program for re-assembly. Most minor errors in typing may be corrected by cancelling the line (control-X or escape) and restarting, or by modification of the partially typed line.

A combination of the two methods is also possible, reading the tape from an earlier edition in on pass 1 with the perforator turned on, making corrections manually by stopping the reader, pulling portions to be deleted through, and keying in portions to be added. The repunched tape is used as text input on pass 2.

Samples of the listing generated during pass 1 and pass 2 are given in Figures 2.1 and 2.2. Another example with a step-by-step procedure is given in Appendix B.

```
0:/  TYPICAL ASSEMBLY FOR A VERY SMALL SAMPLE PROGRAM.
0:P5   =10              ASSIGN VALUE 10 TO NAME P5
0:      : 15
15:       NOP
16:LABA,    CLB
17:       FIM P5 89      /NOTE COMMENT
19:       SRC P5
20:LAB2,  RDR
21:       JTN LAB2
23:       JMS TR3
25:       JUN LABA
27:TR3,   LDM 8
28:       ADD 11
29:       XCH 5
30:       BBL 7
31:$
```

Figure 2.1. Pass 1 Listing

```
  0:    0:    0:    15:BPPPPPPPPF
  16:BNNNNPPPPF    17:BPPNPNPNPF BPNPNNPPNF     19:BPPNPNPNNF
  20:BNNNPNPNPF    21:BPPPNNPPNF BPPPNPNPPF     23:BPNPNPPPPF
BPPPNNPNNF    25:BPNPPPPPPF BPPPNPPPPF    27:BNNPNNPPPF
  28BNPPPNPNNF    29:BNPNNPNPNF    30:BNNPPPNNNF    31:
F
```

Figure 2.2. Pass 2 Listing

Assembly Language

The assembler operates with the 64-character subset of ASCII generated by the ASR-33 teletype, along with the control characters: carriage return, linefeed, escape, start of heading, start of text, and delete or rubout. The 31 character subset containing the lower case letters are ignored by the assembler, and are treated the same as the delete character.

Source instruction mnemonic statements include all of those specified in the MCS-4 Users Manual. This set is augmented by extended mnemonics for conditional jumps and a pseudo operand used for equating labels to values or modification of the assembly address.

Symbolic addressing is provided for by definition of labels consisting of one or more characters.

The use of comment fields and transparent headers is also provided for by the assembler.

CONTROL CHARACTERS

In the discussion to follow, the following generic terms will be used freely:

Control: Any of the first 32 codes in the full ASCII set, obtained on the teletype by one of the special keys, linefeed, return, or ESC; or by holding down the "CTRL" key while typing a letter key.

Separator: Any of the first 48 codes in the full ASCII set, including the control, space, comma, plus, minus, slash, etc. Any number of separators may be linked together wherever a separator is to be used, but the other separators should normally be used singly.

Digit: Any of the ten digits, 0-9.

Letter: Any of the 26 capital letters of the alphabet.

Special significance is attached to the following characters by the assembler:

LINEFEED: Source text lines are initiated by linefeed characters, which are recognized by the assembler to condition the prefixing of the address value of the current location to be assembled.

SOH (Start of Heading – Control A): If the first character after a linefeed is an SOH, all characters following it until the next STX are ignored by the assembler and not printed on pass 1 of the assembly.

STX (Start of Text – Control B): Heading information initiated by an SOH is terminated by a STX. Another SOH may follow the STX, with more heading information, which is in turn followed by another STX, and so on, as desired. The assembler delimits the address counter typeout with an SOH-STX pair, so that its presence on the tape will not interfere with the subsequent use of the tape as source text input.

ESC (Escape): An erroneous line of input source text may be cancelled by an escape character or a cancel (Control-X) if it is typed in before the terminal separator of the mnemonic or any required operands. The assembler responds to an escape or a cancel by typing the up arrow (↑) then ringing the bell. The line will be restarted after the next linefeed. If the up-arrow and bell response is not forthcoming, the cancel has been ignored, because either it is too late in the line (i.e., the terminal delimiter has already been accepted) or the line has been identified by the assembler as a comment line.

RUBOUT (Delete): This character is ignored by the assembler, and may occur anywhere in the input text except in front of an SOH. The preparation of tapes of source text is facilitated by the fact that erroneous characters accidentally punched into the tape may be effectively removed by backing the tape up in the perforator and repunching the frames with rubouts.

$: A dollar sign as the first character following the linefeed or STX signals the end of the source text to the assembler, and conditions the punching of the leader or trailer in the object tape.

+ –: A plus or minus sign serves simultaneously as an operational sign in the computation of an operand value, and as the terminal delimiter for the term immediately preceeding it.

* : The asterisk is recognized by the assembler in an operand field to have a value equal to the address of the (first byte of the) current instruction.

NUMBER SYSTEM

All numeric values in the source code are recognized by the assembler as decimal, and all numeric values generated by the assembler are in decimal. Since the internal representation of the numeric values is 12-bit binary, the largest value that may be accommodated is 4095 ($2^{12} - 1$). Decimal numbers in the source text must not exceed 4095 for correct operation of the assembler, although there is no such restriction on the computation of values, except that the value will truncate modulo 4096 at each step. Negative numbers may be calculated, and are handled in two's complement notation, e.g., –N = 4096 –N.

FORMAT

The assembler is a line-statement, free-format assembler in the following sense: each line of the source text (except for comment lines) assembles into one machine instruction or one byte of data; each field of the source code line is defined by its context and not by its fixed position in the line.

The typical line of source code begins with a linefeed. The next character determines the interpretation of the line. If it is a dollar sign, $, the previous line is considered as the end of the source text and the assembler proceeds to the next pass. If it is a control A (ASCII start of header, SOH) header information terminated by a Control B (ASCII start of text, STX) is anticipated. If it is an alphabetic character, it is interpreted as the first character of a label. A space is interpreted as the terminator for a null label field. The assembler then expects a mnemonic or a pseudo operand. If a "[", "]", "@", "/", "↑", or "←", is entered, a fatal error results. If the first character is any other than those mentioned above, the remainder of the line is treated as comment.

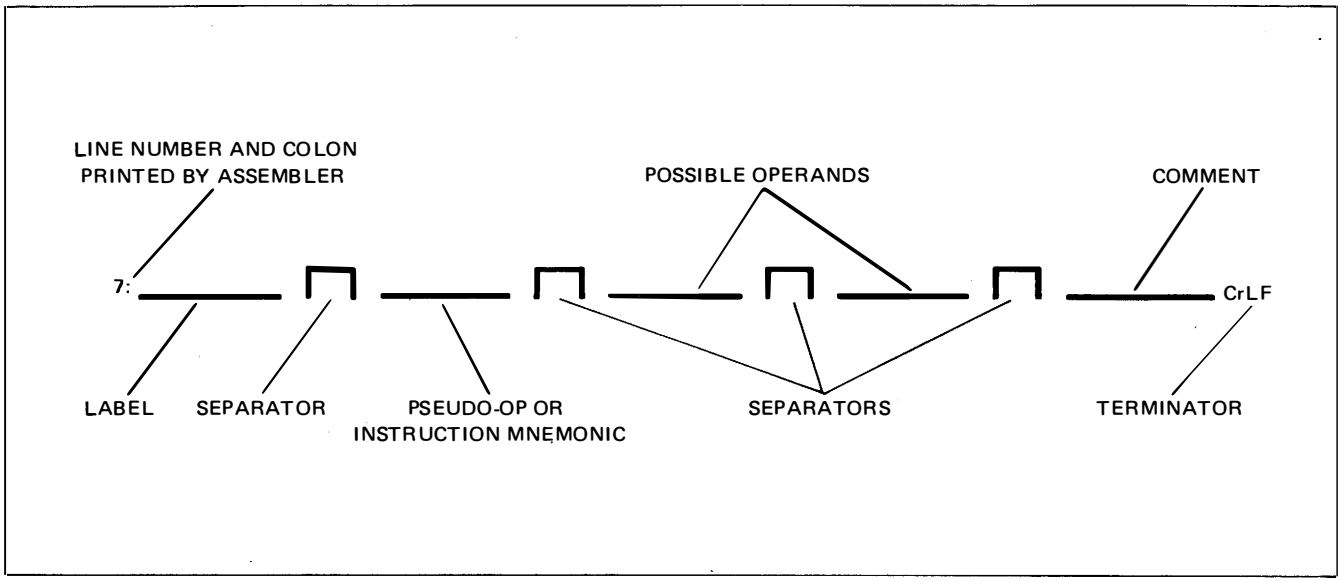


Figure 5.1. Statement

The division of a statement by fields is exemplified in Figure 5.1. The following sections describe the various elements in detail. Statements accordingly may assume several configurations.

Names

Eight entries in the name table are accomodated by each 4002 RAM chip. Maximums of 31 and 127 names are provided for the SIM4-01 and SIM4-02 boards respectively. A name consists of one or more characters, the first of which is a letter. The rest of the characters may be letters, digits, or any of the following special characters: ":", ";", "<", "=", ">", "?", "0", "[", "/", "]", "↑", "←". No imbedded separators are allowed.

To assure consistent operation, distinct names should be unique through the first three characters, although in some cases this may not be essential (the sum of the binary representations of all characters after the third, modulo 4, is used to distinguish names which are identical in the first three characters). The following are some examples of valid names:

CLB
CZ
POLO
A
G = 3A
XYZ
XYZW

Note that the first three are valid names, and have no pre-assigned values. The following are examples of invalid names:

35A (does not begin with a letter)
C/D (contains an imbedded separator)

Labels

The label field of the line of source text begins immediately after the linefeed (or STX) and ends with the first separator. If the label field is null (i.e., no label in the line) the first separator must be a space to avoid making the whole line a comment line. All labels must conform to the rules for the orthography of names. Every name used in an operand field in the program to be assembled must occur exactly once in a label field, and the numeric value of the name is equal to the address of the instruction or datum on the line with the lable, or the value assigned to the label by the "=" pseudo-op. If the name occurs in more than one label field, an error is indicated, and the most recent value applies. If the name does not occur in a label field an error is indicated, and the value zero (0) is used.

Instruction Mnemonics

All of the instruction mnemonics defined in the MCS-4 Users Manual are recognized by the assembler. In addition, the following extended mnemonics are recognized as particular cases of the conditional jump:

JTZ	Jump on test zero
JTN	Jump on test not zero
JTO	Jump on test one
JCZ	Jump on carry/line zero
JNC	Jump on no carry (i.e. = 0)
JCO	Jump on carry/link one
JOC	Jump on carry
JAZ	Jump on accumulator zero
JNZ	Jump on accumulator not zero
JAN	Jump on accumulator not zero

All of the memory and accumulator group instructions, and the NOP instruction require no operand field; the ISZ, FIM and JCN instructions require two operand fields; all other instructions, including the extended mnemonics for the conditional jump instruction, require one operand field.

Pseudo-Operators

The assembler is provided with one pseudo-operator having two functions. It serves to equate labels to values other than instruction addresses, and it enables assembly to begin at some address other than 0. The pseudo-op consists of one of the following characters in the mnemonic field of a source text line:

":", ";", "<", "=", ">", "?"

This special character is followed by a single operand field, which, when evaluated, becomes the value of the pseudo-op. If the label field of the pseudo-op line is null, the value of the operand field becomes the address of the next instruction to be assembled. If the label field contains a name it is assigned the value of the operand. It should be noted that the excessive use of this pseudo-op to define the address of the next instruction, other than at the beginning of individual ROM pages, will lead to discontinuities in the object code which are not recognized by the 1701 programmer, and the practice should be avoided. Note also that when using the pseudo-op to define the address of the next instruction to be assembled, the operand field may not contain (as yet) undefined names, since this will result in ambiguities which will not be flagged as erroneous. The following are some examples of the proper use of the pseudo-op:

```
CZ = 10
POLL = 6
: 0  (This is unnecessary at the beginning of a program, since the assembler always begins at zero anyway).
NEXT: 512
?Next    (Start at the beginning of ROM page 2)
```

Operand Fields

An operand field begins after the space(s) which terminates the mnemonic field, or after the separator which terminates the previous operand field. It consists of one or more terms separated by operational signs (+ or —) and is terminated by a separator other than an operational sign (such as a space, comma, or carriage return), with no imbedded separators other than the operations signs. Each term in the operand field may be a decimal number, a register pair designation, a register designation, a name, or the special symbol "*", which has a value equal to the address of the first byte of the current instruction. The value of the operand is equal to the two's complement algebraic sum of the terms, modulo 4096. If the operand is larger than the receiving field in the instruction, it is truncated on the left (i.e., the most significant bits are removed as necessary) to fit, with no error indication, except for off-page references in the address field of the ISZ and conditional jump instructions. Some examples of valid operand fields:

```
ABC
5,
B + 6
* —1     (The address of the byte immediately preceding the current instruction)
A—B+13 = Q
3P  Register Pair 3, value 6
        (Null operand, value zero)
```

It should be noted that null operand fields must be terminated by a parenthesis, asterisk, period, comma, or slash. Some invalid operand fields:

```
O/F      (Imbedded Separator)
=3A      (Invalid term may not begin with =)
$        (Separator is ignored, and search continues for operand)
```

Register Designators

Instructions which operate on register pairs must be coded with the even-numbered register number, or an equivalent expression, in the operand field. Thus "SRC 2" refers to registers 2 and 3, not pair 2 in the sense of registers 4 and 5. If it is desired to code register pair numbers, the same instruction may be coded "SRC 1P". Any register pair (or in fact any even numbered register) may be so defined by a single digit followed by a letter or other non-digit, non-separator. It is essential that the register operand field evaluate to an even number for those instructions which operate on register pairs, since if the operand evaluates to an odd number the instruction op code will be altered, with no indication of error. The following are some examples of valid register pair operands:

```
5P
3:
PO11    (assuming PO11 has been equated to an even number)
513-7   (truncated to 10)
```

Conditional Jumps

The first operand field in the JCN instruction is evaluated in a strictly numeric fashion, just as all other operand fields. Thus to effect a jump on the condition of zero carry, the numeric value of the first operand field of the JCN must be 10. If it is desired to use mnemonic condition names such as CZ or AN, etc., these names must be equated to their numeric equivalents:

```
CZ = 10
AN = 12
JCN CZ Label
```

To avoid using up valuable name table space on condition names, the extended mnemonics may be used for the conventional conditional jumps. The following three instructions will assemble to the same object code, provided CZ has been equated to 10:

```
JCN CZ LABEL
JCN 10 LABEL
JCZ LABEL
```

Data Constants

It is desirable to define numeric or address constants to be assembled into the ROM image independent of any instructions (such as might be accessed by a FIN instruction). This is possible by using a null mnemonic field. The assembler will then expect one operand field, the first term of which must be a positive number. In all other respects, the operand field is evaluated the same as any other operand field, and it is assembled into one byte of object code, truncating to eight bits if necessary with no error indication. The following are some examples of valid data constants:

```
0
4095    (truncated to 255)
00+ABC  (equal to the address ABC)
```

The following are some examples of invalid constant data fields:

```
ABC     (does not begin with a number)
-18     (number is not positive)
```

ERROR FLAGS

Five errors are recognized by the assembler, and each one is indicated by a single character typeout followed by a bell. Some of the errors are detected in pass 1, and some of them are not detected until pass 2. The typeout for errors detected in pass 2 occurs between the colon typed with the location address, and the "B" which begins the object code for that instruction. Corrective action is indicated where it is possible.

FLAG	PASS	DESCRIPTION
"	1	Duplicate label. The value assigned on this line supercedes the previous value for all subsequent references. If typing in source text manually, recovery may be effected by equating this label to its former value, then selecting a new name to restart the line. For example: 25: ABC SRC 2 Original Label. . . . 43: ABCD " = 25 Old Value Reassigned 43: ABCD1 ADM New Line Restarted
%	1	Name table overflow. May occur on pass 2 if the source text differs from that used in pass 1, a procedure not recommended. The label so flagged has not been added to the name table, and any reference to it will be flagged as undefined on pass 2. No remedy is possible except to add more 4002 RAM (up to the limit of 16 RAMs), or to remake the source program with fewer labels.
!	1 or 2	Unrecognized instruction mnemonic. One byte, with an op code of zero and a four-bit operand is assembled. The line may be cancelled before the terminal delimiter of the operand, and re-typed with the corrected mnemonic.
?	2	Undefined name in an operand field. The entire operand is assembled as a zero value to facilitate correction of a PROM (any remaining terms are not examined, and may be incorporated into a second operand field). Since this flag is issued only on pass 2, reference must be made to the corresponding line of the pass 1 listing to determine (by conjecture) the offending name. Unless the error is patchable, re-assembly is normally required.
&	2	Off page reference by JCN or ISZ instruction operand. Normally, no recovery is possible, and the source program must be rearranged and re-assembled to correct the error.

OUTPUT TAPE

The assembler generates a ROM programming tape in the "BNPF" format required by the MP7-02 programmer system, when connected to the SIM4 system. It may also be used as mask development on the 4001 ROM. The output tape is preceeded and followed by 12 inches of nulls, and has an average of four locations per line. Each instruction is identified with the address in decimal, followed by a colon, for correlation with the pass 1 listing.

Extensive use of pseudo-ops (which generate no object code, but which punch an address anyway) and cancelled lines will cause the listing of the object code to pile up on the right margin, but this will not adversely affect the operation of the ROM programming.

PROGRAMMING SUGGESTIONS

Users of the Fortran program, ASM4, will find it convenient to limit their programming habits in some of the following ways to enhance compatibility between the two assemblers:

1. Names should not exceed five characters in length, and should be constructed only of letters and digits.
2. Standard condition names and register identifier names should not be used for any other purpose.
3. Instruction mnemonics should not be used as names.
4. Labels should be terminated by a comma-space.
5. Operand fields should be delimited by spaces.
6. All numeric values should be in decimal.
7. All conditional jumps should be coded with the JCN mnemonic and the appropriate condition name.
8. Comments on source text lines should be preceded by slashes, and comment lines must have a slash in column 1 (i.e., the first character after the linefeed).
9. Pseudo-ops should be avoided, except to assign values to condition names at the beginning or end of the program tape (where they are easily removed).

LIST OF OPERATING INSTRUCTIONS

[Those instructions preceded by an asterisk (*) are 2 word instructions that occupy 2 successive locations in ROM]

MACHINE INSTRUCTIONS (Logic 1 = Low Voltage = Negative Voltage; Logic 0 = High Voltage = Ground)

MNEMONIC	OPR $D_3 D_2 D_1 D_0$	OPA $D_3 D_2 D_1 D_0$	DESCRIPTION OF OPERATION
NOP	0 0 0 0	0 0 0 0	No operation.
*JCN	0 0 0 1 $A_2 A_2 A_2 A_2$	$C_1 C_2 C_3 C_4$ $A_1 A_1 A_1 A_1$	Jump to ROM address $A_2 A_2 A_2 A_2$, $A_1 A_1 A_1 A_1$ (within the same ROM that contains this JCN instruction) if condition $C_1 C_2 C_3 C_4$[1] is true, otherwise skip (go to the next instruction in sequence).
*FIM	0 0 1 0 $D_2 D_2 D_2 D_2$	R R R 0 $D_1 D_1 D_1 D_1$	Fetch immediate (direct) from ROM Data D_2, D_1 to index register pair location RRR.[2]
SRC	0 0 1 0	R R R 1	Send register control. Send the address (contents of index register pair RRR) to ROM and RAM at X_2 and X_3 time in the Instruction Cycle.
FIN	0 0 1 1	R R R 0	Fetch indirect from ROM. Send contents of index register pair location 0 out as an address. Data fetched is placed into register pair location RRR.
JIN	0 0 1 1	R R R 1	Jump indirect. Send contents of register pair RRR out as an address at A_1 and A_2 time in the Instruction Cycle.
*JUN	0 1 0 0 $A_2 A_2 A_2 A_2$	$A_3 A_3 A_3 A_3$ $A_1 A_1 A_1 A_1$	Jump unconditional to ROM address A_3, A_2, A_1.
*JMS	0 1 0 1 $A_2 A_2 A_2 A_2$	$A_3 A_3 A_3 A_3$ $A_1 A_1 A_1 A_1$	Jump to subroutine ROM address A_3, A_2, A_1, save old address. (Up 1 level in stack.)
INC	0 1 1 0	R R R R	Increment contents of register RRRR.[3]
*ISZ	0 1 1 1 $A_2 A_2 A_2 A_2$	R R R R $A_1 A_1 A_1 A_1$	Increment contents of register RRRR. Go to ROM address A_2, A_1 (within the same ROM that contains this ISZ instruction) if result $\neq 0$, otherwise skip (go to the next instruction in sequence).
ADD	1 0 0 0	R R R R	Add contents of register RRRR to accumulator with carry.
SUB	1 0 0 1	R R R R	Subtract contents of register RRRR to accumulator with borrow.
LD	1 0 1 0	R R R R	Load contents of register RRRR to accumulator.
XCH	1 0 1 1	R R R R	Exchange contents of index register RRRR and accumulator.
BBL	1 1 0 0	D D D D	Branch back (down 1 level in stack) and load data DDDD to accumulator.
LDM	1 1 0 1	D D D D	Load data DDDD to accumulator.
JTZ	0 0 0 1 $A_2 A_2 A_2 A_2$	0 0 0 1 $A_1 A_1 A_1 A_1$	Jump if test zero (5)
JTN	0 0 0 1 $A_2 A_2 A_2 A_2$	1 0 0 1 $A_1 A_1 A_1 A_1$	Jump if test not zero (5)
JTO	0 0 0 1 $A_2 A_2 A_2 A_2$	1 0 0 1 $A_1 A_1 A_1 A_1$	Jump if test one (5)
JCZ	0 0 0 1 $A_2 A_2 A_2 A_2$	0 0 1 0 $A_1 A_1 A_1 A_1$	Jump if carry/link zero (5)
JNC	0 0 0 1 $A_2 A_2 A_2 A_2$	1 0 1 0 $A_1 A_1 A_1 A_1$	Jump if no carry (5)
JCO	0 0 0 1 $A_2 A_2 A_2 A_2$	0 0 1 0 $A_1 A_1 A_1 A_1$	Jump if on carry/link one (5)
JOC	0 0 0 1 $A_2 A_2 A_2 A_2$	0 0 1 0 $A_1 A_1 A_1 A_1$	Jump on carry (5)
JAZ	0 0 0 1 $A_2 A_2 A_2 A_2$	0 1 0 0 $A_1 A_1 A_1 A_1$	Jump if accumulator equalt to 0 (5)
JNZ	0 0 0 1 $A_2 A_2 A_2 A_2$	1 1 0 0 $A_1 A_1 A_1 A_1$	Jump if accumulator non zero (5)
JAN	0 0 0 1 $A_2 A_2 A_2 A_2$	1 1 0 0 $A_1 A_1 A_1 A_1$	Jump if accumulator non zero (5)

INPUT/OUTPUT AND RAM INSTRUCTIONS

(The RAM's and ROM's operated on in the I/O and RAM instructions have been previously selected by the last SRC instruction executed.)

MNEMONIC	OPR $D_3 D_2 D_1 D_0$	OPA $D_3 D_2 D_1 D_0$	DESCRIPTION OF OPERATION
WRM	1 1 1 0	0 0 0 0	Write the contents of the accumulator into the previously selected RAM main memory character.
WMP	1 1 1 0	0 0 0 1	Write the contents of the accumulator into the previously selected RAM output port. (Output Lines)
WRR	1 1 1 0	0 0 1 0	Write the contents of the accumulator into the previously selected ROM output port. (I/O Lines)
WPM(6)	1 1 1 0	0 0 1 1	Write the contents of the accumulator into the previously selected half byte of read/write program memory (for use with 4008/4009 only)
WRϕ (4)	1 1 1 0	0 1 0 0	Write the contents of the accumulator into the previously selected RAM status character 0.
WR1(4)	1 1 1 0	0 1 0 1	Write the contents of the accumulator into the previously selected RAM status character 1.
WR2(4)	1 1 1 0	0 1 1 0	Write the contents of the accumulator into the previously selected RAM status character 2.
WR3(4)	1 1 1 0	0 1 1 1	Write the contents of the accumulator into the previously selected RAM status character 3.
SBM	1 1 1 0	1 0 0 0	Subtract the previously selected RAM main memory character from accumulator with borrow.
RDM	1 1 1 0	1 0 0 1	Read the previously selected RAM main memory character into the accumulator.
RDR	1 1 1 0	1 0 1 0	Read the contents of the previously selected ROM input port into the accumulator. (I/O Lines)
ADM	1 1 1 0	1 0 1 1	Add the previously selected RAM main memory character to accumulator with carry.
RDϕ (4)	1 1 1 0	1 1 0 0	Read the previously selected RAM status character 0 into accumulator.
RD1(4)	1 1 1 0	1 1 0 1	Read the previously selected RAM status character 1 into accumulator.
RD2(4)	1 1 1 0	1 1 1 0	Read the previously selected RAM status character 2 into accumulator.
RD3(4)	1 1 1 0	1 1 1 1	Read the previously selected RAM status character 3 into accumulator.

ACCUMULATOR GROUP INSTRUCTIONS

CLB	1 1 1 1	0 0 0 0	Clear both. (Accumulator and carry)
CLC	1 1 1 1	0 0 0 1	Clear carry.
IAC	1 1 1 1	0 0 1 0	Increment accumulator.
CMC	1 1 1 1	0 0 1 1	Complement carry.
CMA	1 1 1 1	0 1 0 0	Complement accumulator.
RAL	1 1 1 1	0 1 0 1	Rotate left. (Accumulator and carry)
RAR	1 1 1 1	0 1 1 0	Rotate right. (Accumulator and carry)
TCC	1 1 1 1	0 1 1 1	Transmit carry to accumulator and clear carry.
DAC	1 1 1 1	1 0 0 0	Decrement accumulator.
TCS	1 1 1 1	1 0 0 1	Transfer carry subtract and clear carry.
STC	1 1 1 1	1 0 1 0	Set carry.
DAA	1 1 1 1	1 0 1 1	Decimal adjust accumulator.
KBP	1 1 1 1	1 1 0 0	Keyboard process. Converts the contents of the accumulator from a one out of four code to a binary code.
DCL	1 1 1 1	1 1 0 1	Designate command line.

NOTES: (1)The condition code is assigned as follows:

$C_1 = 1$ Invert jump condition
$C_1 = 0$ Not invert jump condition
$C_2 = 1$ Jump if accumulator is zero
$C_3 = 1$ Jump if carry/link is a 1
$C_4 = 1$ Jump if test signal is a 0

(2)RRR is the address of 1 of 8 index register pairs in the CPU.

(3)RRRR is the address of 1 of 16 index registers in the CPU.

(4)Each RAM chip has 4 registers, each with twenty 4-bit characters subdivided into 16 main memory characters and 4 status characters. Chip number, RAM register and main memory character are addressed by an SRC instruction. For the selected chip and register, however, status character locations are selected by the instruction code (OPA).

(5)Extended Mnemonic

(6)The SIM4 Hardware Assembler is currently being modified to accept the WPM instruction associated with the 4008/4009.

SAMPLE ASSEMBLY with a STEP-by-STEP PROCEDURE

As an example, assume that one wishes to perform a logical "and" function on the data at two 4 bit ROM input ports and display the result at a RAM output port.
The first step of course, is to write the program using the MCS-4 instruction set. The result may be as shown in Figure B-1.

```
/                    FOUR BIT "AND" ROUTINE
START, FIM 4P 0       /  LOAD ROM PORT 0 ADDRESS
 SRC 4P               /  SEND ROM PORT ADDRESS
 RDR                  /  READ INPUT A
 XCH 0                /  A TO REGISTER 0
 INC 8                /  LOAD ROM PORT 1 ADDRESS
 SRC 4P               /  SEND ROM PORT ADDRESS
 RDR                  /  READ INPUT B
 XCH 1                /  B TO REGISTER 1
 JMS AND              /  EXECUTE "AND"
 XCH 2                /  LOAD RESULT C
 WMP                  /  STORE AT MEMORY PORT 0
 JUN START            /  RESTART
 NOP
 =104
/                    "AND" SUBROUTINE
AND, CLB              /  CLEAR ACCUMULATOR AND CARRY
 XCH 2                /  CLEAR REGISTER 2
 LDM 4                /  LOAD LOOP COUNT (LC)
 XCH 0                /  LOAD A, LC TO REGISTER 0
 RAR                  /  ROTATE LEAST SIGNIFICANT BIT TO CARRY
 XCH 0                /  RETURN ROTATED A TO REG 0, LC TO ACC.
 JCN CZ ROTR1         /  JUMP TO ROTR1 IF CARRY ZERO
 XCH 1                /  LOAD B, LC TO ACCUMULATOR
 RAR                  /  ROTATE LEAST SIGNIFICANT BIT TO CARRY
 XCH 1                /  RETURN ROTATED B TO REG.1, LC TO ACC.
ROTR2, XCH 2          /  LOAD PARTIAL RESULT C, LC TO REGISTER 2
 RAR                  /  ROTATE CARRY INTO PARTIAL RESULT MSB
 XCH 2                /  LOAD LC, RETURN C TO REGISTER 2
 DAC                  /  DECREMENT THE ACCUMULATOR (LC)
 JCN ANZ AND+3        /  LOOP IF LC NON ZERO
 BBL 0                /  RETURN
ROTR1, XCH 1          /  LOAD B, LC TO REGISTER 1
 RAR                  /  ROTATE B
 XCH 1                /  RETURN ROTATED B TO REG. 1, LC TO ACC.
 CLC                  /  CLEAR CARRY
 JUN ROTR2            /  RETURN TO LOOP
CZ  =10
ANZ  =12
$
```

Figure 1. Source Listing

Figure B-1 was transcribed from a handwritten copy to a teletype print out and punched paper tape to facilitate loading during actual assembly. To accomplish this, an ASR 33 teletype was used and the following steps were taken:

1. The TTY was placed in the "offline" mode.
2. The paper tape punch control was placed in an "on" condition.
3. Handwritten data was keyed into the teletype keyboard.

Some typographical errors were edited by using the TTY's backspace punch control and rubout character. The rubout is an all "1"s character which effectively deletes any character over which it is superimposed. The procedure is as follows:

1. Determine the number of backspaces required to return the punch to the erroneous character.
2. Depress the paper tape punch backspace control until the erroneous character is reached.
3. Enter a "rubout" from the keyboard. If a new character must be inserted, the previous character and the remaining line or lines must be deleted with rubouts.
4. Enter the desired character and remaining lines.

The assembler's editing features may also be used to simplify the task of correcting errors. As an example, assume the 18th instruction of the listing "RAR", were incorrectly entered as "RBR" (it would be unrecognized when assembled). A control-X (control characters are entered by simultaneously depressing the control key and character) could be entered after the error. The assembler would detect this and respond by requesting the line again. The correct line is then entered. The assembler maintains control of the tape reader during these operations but when editing directly from the paper tape it is advisable to manually advance the reader. This may be accomplished by using the reader's manual start/ stop control switch. (Do not attempt to advance the tape directly).

Some comments regarding the format of the listing are present below.

1. Each line must be preceded by a carriage return and a linefeed.
2. Lines which are entirely comment must begin with any separator other than a space.
3. All unlabeled lines must begin with a space.
4. All condition mnemonics must be defined by decimal constants. For example, the 20th line of the listing contains a CZ which is later defined as a "10" on line 35.
5. The pseudo operator is used to adjust the assembly address. Line 13 consists of a space, equal sign, and the number 104. The assembler will interpret this as a command to define the next line as line 104. The subroutine, "and", will thereby be located, at address 104.
6. The pseudo operator is also used in the last two lines of the listing to define constants. If the equal sign is preceeded by a label the assembler will assign the subsequent value to the label.
7. The dollar sign is an assembly terminator.

Pass 1 of the assembly procedure may now be completed. The corrected paper tape will be read line by line by the assembler. During this operation a name table is generated and a listing (figure B-2). Preceding each line will be an

```
  0:/                  FOUR BIT "AND" ROUTINE
  0:START, FIM 4P 0     /  LOAD ROM PORT 0 ADDRESS
  2:  SRC 4P            /  SEND ROM PORT ADDRESS
  3:  RDR               /  READ INPUT A
  4:  XCH 0             /  A TO REGISTER 0
  5:  INC 8             /  LOAD ROM PORT 1 ADDRESS
  6:  SRC 4P            /  SEND ROM PORT ADDRESS
  7:  RDR               /  READ INPUT B
  8:  XCH 1             /  B TO REGISTER 1
  9:  JMS AND           /  EXECUTE "AND"
 11:  XCH 2             /  LOAD RESULT C
 12:  WMP               /  STORE AT MEMORY PORT 0
 13:  JUN START         /  RESTART
 15:  NOP
 16:  =104
104:/                  "AND" SUBROUTINE
104:AND, CLB           /  CLEAR ACCUMULATOR AND CARRY
105:  XCH 2            /  CLEAR REGISTER 2
106:  LDM 4            /  LOAD LOOP COUNT (LC)
107:  XCH 0            /  LOAD A, LC TO REGISTER 0
108:  RAR              /  ROTATE LEAST SIGNIFICANT BIT TO CARRY
109:  XCH 0            /  RETURN ROTATED A TO REG 0, LC TO ACC.
110:  JCN CZ ROTR1     /  JUMP TO ROTR1 IF CARRY ZERO
112:  XCH 1            /  LOAD B, LC TO ACCUMULATOR
113:  RAR              /  ROTATE LEAST SIGNIFICANT BIT TO CARRY
114:  XCH 1            /  RETURN ROTATED B TO REG. 1, LC TO ACC.
115:ROTR2, XCH 2       /  LOAD PARTIAL RESULT C, LC TO REGISTER 2
116:  RAR              /  ROTATE CARRY INTO PARTIAL RESULT MSB
117:  XCH 2            /  LOAD LC, RETURN C TO REGISTER 2
118:  DAC              /  DECREMENT THE ACCUMULATOR (LC)
119:  JCN ANZ AND+3    /  LOOP IF LC NON ZERO
121:  BBL 0            /  RETURN
122:ROTR1, XCH 1       /  LOAD B, LC TO REGISTER 1
123:  RAR              /  ROTATE B
124:  XCH 1            /  RETURN ROTATED B TO REG. 1, LC TO ACC.
125:  CLC              /  CLEAR CARRY
126:  JUN ROTR2        /  RETURN TO LOOP
128:CZ  =10
128:ANZ  =12
128:$
```

Figure 2. Pass 1 Listing

address and a colon printed by the assembler. The line itself is read by the assembler and echoed back to the printer. The listing thereby acts as a rough check on the serial link between the TTY and SIM4 hardware. The procedure is as follows:

1. Assemble the appropriate hardware consting of the following
   a. SIM4-01 or SIM4-02 board with a minimum of 3 RAMs (Intel 4002), and the 4 assembler ROMs A0740, A0741, A0742 and A0743.
   b. An ASR 33 (with reader, punch, and ASCII keyboard) teletype wired with a reader on/off control relay and serial data link described in the MCS-4 Users Manual.
   c. SIM4 Power Supplies. Ref. MCS-4 Users Manual.
   d. Customer supplied interconnect wiring. Ref. MCS-4 Users Manual.
2. Connect the hardware as shown in block diagram of figure 1.1.
3. Place the TTY punch in an "off" mode.
4. Place the TTY reader in a "free" or "off" condition and position the tape on the leader.
5. Reset the SIM4 board. A customer-supplied reset switch is assumed.
6. Start the TTY reader.
7. If an error occurs (flagged by a bell and a special character) stop the reader and take corrective action with the keyboard if possible. Tape position must be carefully controlled during editing to avoid further erroneous input.
8. The tape will be read until the dollar sign $ is reached. At this point pass 1 of the assembly is complete and the assembler is awaiting the second entry of the tape for pass 2.
9. Turn the punch "on".
10. Place the tape reader in the "free" or "off" condition and position the tape on the leader.
11. The assembler will punch approximately 12 inches of leader and wait for reader input.
12. Place the reader in the "start" mode.

The assembler will read the tape, ignore comments, and output line addresses and "BNPF" formatted instruction codes. With the punch on a programming tape is created and a listing of the tape such as that shown in figure B-3 is generated. Any editing performed during pass 1 must be repeated during pass 2.

```
   0:   0:BPPNPNPPPF BPPPPPPPNF    2:BPPNPNPPNF    3:BNNNPNPNPF
   4:BNPNNPPPPF     5:BPNNPNPPPF    6:BPPNPNPPNF    7:BNNNPNPNPF
   8:BNPNNPPPNF     9:BPNPNPPPPF BPNNPNPPPF   11:BNPNNPPNPF
  12:BNNNPPPPNF    13:BPNPPPPPPF BPPPPPPPPF   15:BPPPPPPPPF
  16: 104: 104:BNNNNPPPPF  105:BNPNNPPNPF  106:BNNPNPNPPF  107:BNPNNPPPP
 108:BNNNNPNNPF  109:BNPNNPPPPF  110:BPPPNNPNPF BPNNNNPNPF
 112:BNPNNPPPNF  113:BNNNNPNNPF  114:BNPNNPPPNF  115:BNPNNPPNPF
 116:BNNNNPNNPF  117:BNPNNPPNPF  118:BNNNNNPPPF  119:BPPPNNNPPF
BPNNPNPNNF  121:BNNPPPPPPF  122:BNPNNPPPNF  123:BNNNNPNNPF
 124:BNPNNPPPNF  125:BNNNNPPPNF  126:BPNPPPPPPF BPNNNPPNNF
 128: 128: 128:
F
```

Figure 3. Programming Tape Listing

Completion of pass 2 is signalled by the printing of an "F" and the punching of a 12 inch trailer.

At this point the assembly is complete and PROM or ROM programming may proceed.

APPENDIX G
SIM4 HARDWARE SIMULATOR

INTRODUCTION

The SIM4-02 Hardware Simulator is a program written for the MCS-4 ™ series Micro Computer System. This program will provide interactive control over the debugging of other MCS-4 ™ programs.

The minimum configuration required is a SIM4-02 prototype card with three 4002 RAMs and a Teletype. When fully stuffed with 16 RAMs, test programs up to 512 bytes (locations) in length may be accomodated. The hardware simulation program itself occupies nine full ROMs.

The Hardware Simulation Program has two basic functions:

1. To simulate the execution of a test program, tracing its progress, and apprehending gross errors.
2. To allow the user to dynamically interact with and/or modify his test program, in order to facilitate the debugging process.

These two functions are implemented by means of a set of directives or commands which the user types in at the teletype keyboard. Some of the directives call for typeouts by the simulator program, some of the directives signal the input of data or program modifications, and some of the directives involve both typeouts and input response or data.

A directive is identified by a single letter of the alphabet (except the arithmetic conversion directives = and "). If the directive is associated with output only, the typing (or punching) will commence immediately. If input is allowed or required with the directive, the simulation program will enable the paper tape reader control, and wait for valid input data.

NUMBER SYSTEMS

Two number radices are standard with the hardware simulation program: binary and decimal. Index register values, program counter and instruction location values, chip numbers, and some pointers are handled in decimal for convenience. ROM instructions, the accumulator value, and one-bit indicators are handled in binary. Any input number may be entered in either radix by prefixing it with a suitable indentifier ("D" for decimal, "B" for binary), regardless of the expectations of the program. Unless so identified, however, all input should be in the radix used in the corresponding typeout.

To facilitate working with program tapes in the "BNPF" format, the hardware simulation program will accept binary numbers coded either as strings of ones and zeroes, or as strings of "P"s and "N"s, where the letter P is interpreted as a zero, and the letter N is interpreted as a one.

All input numbers are right-justified into the receiving register or field. If the number is smaller than the receiving field, leading zeroes are implied as necessary. If the number is larger than the receiving field, the excess bits are lost from the most-significant end of the number. Thus, if it is attempted to load an index register with the value 20, the result will be 4 in the register. This may be used to advantage in the event of an inadvertant error typein, by typing in as many zeroes as there are bits in the receiving field, then re-typing the number, all as one string of digits. A number typed in may end with a carriage return, a comma, a space, or the letter "F", or in the case of the = directive, with plus or minus sign. Any other characters will give unpredictable results, and should be avoided. Rubouts are the only non "numeric" characters which may be imbedded within the input number strings with no adverse effects. Rubouts are ignored in all cases.

DESCRIPTION

The hardware simulation program allocates a user-selected block of RAM main memory locations to hold the ROM instructions to be simulated, assigning two RAM locations for each simulated ROM location. Thus, to simulate 512 locations of ROM, all 16 RAMs must be used. Any RAM locations not allocated for program storage may be accessed in the normal way by the test program. In addition, the hardware simulation program uses the status characters in twelve consecutive RAM registers (equivalent to three RAM chips) to hold simulation parameters. RAM is assumed to be organized as four consecutive banks (with wraparound) of sixteen registers each, so that if less than 16 RAMs are used, those allocated to program and parameter storage must be in one block of contiguous banks and registers within banks.

The program to be tested may have an address anywhere in the 4096 locations of addressable ROM, since the hardware simulator program adds a bias value to all addresses which reference the simulated ROM. If the program attempts to jump or increment to outside the range of the simulated ROM, an error interrupt occurs.

Another error interrupt occurs in the event of an illegal instruction op code during simulated execution. The op codes which cause this interrupt are: 11111110, 11111111, 11100011, and all instructions with OPR = 0000 except for 00000000 (NOP).

A breakpoint register is associated with the simulated execution mode of operation, allowing the user to pre-set a location which will cause an interrupt before execution. The BREAK key on the teletype may also be used to interrupt execution and some other types of output.

During simulated execution, a count is kept of the number of simulated machine cycles (i.e., sync pulses) used by the test program, to assist in checking out programs with critical timing problems.

DIRECTIVES

"n **Binary conversion**
This directive accepts a single (decimal) number, and types out its equivalent in binary. A maximum of 12 bits (numbers to 4095) may be accomodated at once.

=n **Decimal adder**
This directive accepts a string of (decimal) numbers separated by plus and minus signs, and types out the algabraic sum, modulo 4096. If the algebraic sum is negative, 4095 is logically added to it to give a positive result. This directive may be used to perform binary to decimal conversion thus: =Bnnnn

Qr,s,e **RAM/ROM chip assignment**
This directive must be entered before any other letter directive. If the first character after the Q is a space or comma, the current values are typed out. Then, or immediately after the Q, three parameters separated by commas or spaces are required. If any of the three parameters is omitted, or if a RETURN is typed instead of the first parameter, the current values will be unchanged. **r** is the (decimal) RAM register number (0-63) which is used as the lowest in the block allocated to ROM. The simulation program has no way of preventing the test program from accessing RAM locations allocated to simulated ROM, so the user must use care in selecting a value for this parameter which will reduce the likelihood of improper access. If **r** is greater than 63, the previous value is used. **s** is the starting address of the ROM segment to be simulated. Any attempt to execute an instruction with an address less than this number will result in an out-of-bounds interrupt. **e** is the (decimal) ending address of the ROM segment to be simulated. Any program access to ROM locations greater than this address will result in an out-of-bounds interrupt. This directive clears the option word to zeroes.

Z **Zero**
This directive simulates the hardware reset function, and clears to zero all simulated registers, counters, and all RAMs not allocated to program. The Q directive executes a Z each time the parameters are changed.

In **Input**
This directive accepts a sequence of (binary) numbers and stores them in consecutive simulated ROM locations, beginning with location n. Spaces, commas, returns, and linefeeds may occur with any frequency or pattern between the individual numbers. Input is terminated by a free-standing letter F in the sequence. An ASCII SOH (control A) may be used to introduce a (decimal) number which, like n, becomes the new starting address for subsequent instruction bytes. The program counter in the current stack level is altered by this directive.

Pn,m **Punch**
This directive will punch out, in "BNPF" format with location numbers, the contents of the simulated ROM beginning at location n (decimal), and ending with location m. The currently selected program counter, and the breakpoint register are altered by this directive. Four inches of leader and trailer are punched on the tape, with an "F" after the last location. If the BREAK key on the teletype is depressed between locations, the typeout will be aborted, with no trailer. Both the breakpoint register and the program counter in the current stack level are altered by this directive.

Mn,i **Memory input**
This directive accepts a sequence of (decimal) numbers (0-15) and stores them sequentially in consecutive RAM locations, beginning in register n (decimal, 0-63) and location i (decimal, 0-19). If the starting location is in main memory (i less than 16), only main memory locations are filled. The next number after the one which goes into register **r**, digit 15, goes into register **r** + 1, digit 0. If the starting location is a status character (i greater than 15), only status characters are filled. The sequence ends with a carriage return following the terminal delimiter of the last number.

Dr, n **Dump RAM**
This directive types out in decimal the contents of each RAM location (both main memory and status) of n registers, beginning with register **r**. The typeout may be ended prematurely by depressing the BREAK key.

A **Accumulator**

This directive may be used to display and/or alter the contents of the simulated accumulator. A space or comma following the A will display in binary, the contents of the simulated accumulator. This or the A may be followed either by a new value to be entered, or by a carriage return to end the directive.

C **Carry/Link**

This directive may be used to display and/or alter the contents of the simulated Carry/Link bit. The use of this directive is the same as for A.

Xn **Index**

This directive may be used to display and/or alter the contents of any one or pair of the simulated index registers. If a space or comma follows the X, all 16 index registers are displayed (in decimal). Otherwise, if the (decimal) number n is followed by a space, index register n may be displayed and/or altered as in A. If the number n is followed by a comma, slash, or period, index **pair** number n may be displayed (in decimal) and/or altered as a unit. (Index registers 2n and 2n + 1 are handled together as a single 8-bit number.)

S **Stack pointer**

This directive may be used to display and/or alter the contents of the subroutine stack pointer. The current location counter is the one pointed to by this pointer. This pointer is incremented by JMS instructions and decremented by BBL instructions.

L **Location Counter**

This directive may be used to display and/or alter the contents of the current location counter. Note that altering the value of the stack pointer will cause a different register to be current location counter.

E **Examine Everything**

This directive combines the display functions of the C, A, S, L, R, and X directives. All four program counters in the stack are displayed. No modification is possible.

R **RAM/ROM selection**

This directive may be used to display and/or modify the simulated memory chip/location selection. A space or comma after the R types out an 11-bit binary number, of which the most-significant 3 bits represent the command line selection effected by the last DCL instruction, and the least-significant 8 bits represent the contents of the index pair as last used by an SRC instruction.

B **Breakpoint**

This directive may be used to display and/or modify the contents of the breakpoint register. The simulated execution will always be interrupted before processing the instruction pointed to by the breakpoint. If the breakpoint points to the second byte of a two-byte instruction, no breakpoint action will occur during instruction simulation.

W **When**

This directive may be used to display and/or alter the contents of the simulated sync cycle counter. This is a 12-bit (decimal) counter used to tally the number of instruction cycles used during instruction simulation.

T **Trace**

This directive causes the simulation program to begin simulated execution of the test program, beginning at the address in the current location counter. If instruction execution simulation is interrupted by a breakpoint, the keyboard BREAK key, or an illegal instruction, the T directive will cause the program to resume where it left off, just as if the program was never interrupted, except insofar as program parameters or registers were modified while interrupted. The basic format of the trace listing is

pppp:iiiiiiii c aaaa rr

where **pppp** is the decimal location counter; **iiiiiiii** is the binary representation of the (first byte of the) instruction being simulated; **c** is the resultant carry/link bit; aaaa is the resultant accumulator value; and **rr** is the resultant (decimal) index or index pair used in the instruction. (value, not index number) Any or all of the last three numbers may be omitted on instructions which do not reference their respective registers.

N **Non-trace**

This directive is identical to the T directive, except that execution proceeds without tracing.

O **Options**

This directive may be used to display and/or alter the current option status bits. This is a 4-bit binary number with the following significance:

1 Input, Output, and CPU test instructions are executed directly when this bit is on, instead of typing out on the teletype the port number and then typing or accepting the data.

10 No interrupt for subroutine stack overflow or underflow will occur when this bit is on.

1000 Unconditional jumps and subroutine jumps to ROM page 0 (chip 0) are executed directly instead of interpretively, permitting direct byte I/O during checkout.

ERROR MESSAGES

Most of the errors which can be detected by the simulation program are identified by a single character typeout, followed by ringing the bell once. Six different types of errors are identified this way:

CODE	SIGNIFICANCE
?	This is not a valid directive. Any printed graphic normally generated by the ASR33, which is not a valid directive, evokes this response. A question mark-bell combination also calls attention to a simulated input request.
#	Break condition recognized. This occurs normally, either when the location counter reaches the value in the break register in execution simulation, or when the BREAK key is depressed in simulation or ROM or RAM dumping.
>	Location counter out of range. This error occurs in simulation or ROM punching, if an attempt is made to access an address out of the range specified in the most recent Q directive.
!	Invalid op code. This error occurs and is recognized during execution simulation, after the instruction byte is typed out, but before the location counter is incremented, so that if it occurs under the control of the N directive, the T directive may be entered to examine the error by trying to execute it again.
%	Location counter stack overflow or underflow. This error is unique in that the interruption occurs after the instruction has been executed in simulation. A T or an N directive will resume execution with the next instruction (jumped or returned to).
↑	Cancel. This is the program response to a Cancel (Control "X" or ESCAPE) typein, during data input. Except for I, M, T, and N directives, it cancels the entire directive. If used in the I or M directives, only the current datum is canceled, and the directive is terminated at that point. Previous values, if any, have already been stored in memory. If used while the simulation program is requesting input data from a simulated ROM port or the simulated CPU Test line, it is equivalent to a break at the beginning of the instruction. In each case the simulation program returns to accept another directive.

OPERATING INSTRUCTIONS

1. Assemble Program

First assemble the test program on the Hardware Assembler (A0740 to A0743). *Important:* the Hardware Simulation Program will not accept ROM program tapes created by the FORTRAN assembler, ASM4, as these tapes have bit patterns and addresses together, with no identifier for the addresses. It is not necessary to assemble the program in one contiguous block of ROM locations, since the I directive in the simulation program is able to recognize the address fields (by the Control "A", SOH preceding them), and place the instruction patterns into the proper simulated ROM locations.

2. Prepare SIM4-02 Hardware

Remove ROM chips A0740 through A0743 and plug in the Hardware Simulation Program chips, A0750-A0758. Press RESET. The teletype should type out a carriage return-linefeed, and an asterisk to show that the simulation program is ready to accept a directive.

Determine how much simulated ROM is needed to test the program, and which RAMs are least likely to be accessed by the test program, using if necessary the = and " directives. Then type in the Q directive for this program. From now until the testing of this program has been completed and the amended program tape has been punched out, *DO NOT touch the RESET button. If the RESET button is pressed, the simulation parameters and any program in the simulated ROM will be destroyed.*

3. Load Program

Place the object tape in the teletype reader, and type in I. The simulation program will read both the addresses and the instructions from the tape and store them in the proper locations. Reading will be terminated by any error or by the terminal F punched by the assembler. Note that any instructions or data which fall outside the limits defined in the Q directive will be ignored. Note also that if the Q directive defines the ROM limits to be more than 512 bytes, wraparound overlap is possible. Thus, location 100 would be overwritten by instructions going into location 612. When the program is loaded, a simulated RESET may be effected by the Z directive. If starting at other than location zero, or with registers preloaded with data, the appropriate directives may be used to set these up. A breakpoint may be set if desired, and RAM may be loaded up with data if desired. If a subroutine or a part of a subroutine is being tested, the stack may be loaded with a

return address using the L directive. That may then be pushed down with the S directive, so that the starting address may be loaded into the first subroutine level, or the process may be repeated up to three times. If it is desired to force an interrupt at the first occurrence of a JMS instruction, the stack pointer may be set to 3 initially, so that the first JMS instruction causes a stack overflow. If it is desired to achieve more than one breakpoint, illegal instructions may be assembled or inserted into the program at the desired points. When the simulation attempts execution of one of these locations, an interrupt occurs, and the instruction may be replaced, or the program counter incremented around it to proceed.

4. Execute Program Simulation

To start the execution simulation, type a T (for Trace mode) or an N (for non-Trace mode). If at any time it is desired to stop execution simulation, whether because of program errors, to examine register contents, or to make corrections, the BREAK key may be depressed, and the simulation will be interrupted at the completion of the current instruction. Execution will resume as if uninterrupted, if the T or N directive is typed in after a break.

5. Edit Program

To make corrections to the program, the I directive is used, giving an address, and the value(s) to be entered. The I directive alters the contents of the current location counter. Thus, it should either be noted and restored, or the stack pointer may be incremented first and decremented afterwards – (unless of course, the simulation is interrupted at subroutine nest level 3).

6. Punch New "BNPF" Tape

After the program works correctly, an amended ROM tape may be punched in the "BNPF" format using the P directive. Four inches of leader and trailer are punched by this directive. If more is needed, rubouts or nulls (shift-control-P) may be punched while the simulation program is waiting for a directive. This will not in any way interfere with normal operation of the program. The user should remember to turn on the paper tape punch after typing in the second address in the P directive if a tape is to be made. If it is desired only to examine the contents of a simulated ROM location, this is not necessary.

7. Simulation of Segmented Programs

If a program is not very large, but is scattered over a wide range of addresses, it may be possible to accomodate the program in segments. Suppose the program occupies the first 32 locations in each of four ROMs. 128 locations must be reserved by the Q directive to hold all of this. Suppose further that the program accesses only bank zero in RAM. The Q directive would be something like this:

```
Q16, 0, 127
```

Then the first 32 locations of the program tape are read in using the I directive. The entire tape may be read with no deleterious effects, if that is convenient, or an F may be typed in manually at the end of the first 32 locations' worth of data. Then the Q directive is used again, to re-assign the same locations to the next block of addresses:

```
Q99, 224, 355
```

Note that the address limits have been offset by 32, to prevent the obliteration of the first 32 locations. The object tape may be read in again, or at least that part of it which includes the next block of data or instructions. Then the area is reassigned again:

```
Q99, 448, 575
```

The process is repeated until the whole program is loaded. To execute, the Q directive for the starting block of code is typed in again. If the segments are placed correctly, each time a jump is made to another segment, an out-of-range interrupt occurs. The Q directive for the segment jumped to is entered, and the program may proceed. This technique may also be used to relocate a program in ROM: for example, the following sequence of commands will effectively move (shift) a program up one position in ROM:

```
Q0, 0, 255
I0 (program)
Q0, 1, 256
P1, 256
```

JUMPS TO PAGE 0

Because of the nuisance of doing serial-to parallel conversion, and properly timing the bit frames in teletype input and output, the simulation program is provided with an option to perform subroutine calls and unconditional jumps to ROM page 0 directly, returning to simulation mode upon return. ROM page 0 contains subroutines to perform teletype reader and keyboard input, 7 bits wide (the parity bit is ignored), teletype output 8 bits wide, binary to decimal conversion and output,

the typing of some specialized sequences of characters, partial decimal to binary conversion on input, and 6-bit teletype character input with control character checking. A test program may use these subroutines to facilitate checkout of complex programs, or the ROM may be included in the final program if teletype interface and the same ancillary routines are needed.

The following is a summary of the subroutines and their calling parameters:

NAME	ADDRESS (X)	FUNCTION
KEY	120	(11-15) This routine inputs one 7-bit character from the teletype keyboard, and returns it, left-justified, in index registers 14 and 15. Index registers 12 and 13 are cleared to zero. The least significant bit of register 11 determines whether the character is echoed back (0 = yes, 1 = no). The carry is set if the character typed in is printable.
TTI	117	(11-15) This routine inputs one 7-bit character from the teletype paper tape reader or keyboard (the reader control is enabled), and is otherwise exactly the same as KEY.
TXX	234	(10, 11, 14, 15) This routine examines the carry bit set by KEY or TTI as well as the accumulator value and the character in registers 14 and 15 to determine if one of the following conditions obtains: 1. The character is some printable graphic between "Ø" and "←". If so, the character is biased to a six-bit value, centered in the byte. The carry is turned on if it is a letter. A normal return is taken, acc=0. 2. The character is a control character between null and ETB (control-Y) or a printable graphic between space and slash. An indirect jump to the address in ROM page 1 contained in index registers 10 and 11 is taken. 3. The character is a control between CAN (Control-X) and US (Shift-Control-0). An unconditional jump to location 256 is taken. 4. The character is one of those not generated by a KSR33 teletype, or a rubout. A normal return is taken, with the accumulator non-zero.
T6R	205	(10-15) This routine combines TTI and TXX such that normal return occurs only on characters "0" through "←". Characters in group (4) above are ignored, and the alternate exits are taken for control characters and delimiters. Note that if the address to which the delimiter return is to be made is odd, no echo occurs. On normal return, the character is right-justified in registers 14 and 15, and the carry is set if the character is a letter or higher.
T6L	220	(10-15) This routine is the same as T6R, except that on normal return the carry is always zero, and the character is left-justified in registers 14 and 15, leaving the lower two bits zero. Both T6R and T6L contain subroutine calls nested three deep, and may only be called from the main program, except during simulation.
D10	183	(4-7, 10-15) This subroutine multiplies the 12-bit binary number in registers 5-7 times ten, and adds the number in register 15 to the product, then goes to T6R to input another digit. This routine may be called repeatedly to input and convert to binary a decimal number. A terminal delimiter takes the alternate exit in registers 10 and 11. Register 4 is used for scratch.
Z47	6	(4-7) This routine clears registers 4 through 7 to binary zero in preparation for D10.
PUN	80	(11-15) This routine prints or punches the character in registers 14 and 15 out on the teletype. Registers 11 through 15 are cleared to zero on return.
IPR	70	(11-15) This routine does the same as PUN, except that if register 11 is initially even (echo mode on input), a 15 ms delay occurs to allow the teletype printer to settle.
RETN	107	(11-15) This routine types out a carriage return, a null, and a linefeed. It may only be called from the main routine.
MSG	66	(11-15) This routine types out the character in registers 14 and 15, then follows it with a bell.
SPACE	63	(11-15) This routine types one space.
DIGIT	53	(10-15) This routine types an ASCII digit corresponding to the BCD number in the accumulator. If it is zero, and the register 10 contains 15, a space is typed instead. Unless a space is typed, register 10 is incremented.
PDN	40	(4-7, 10-15) This routine will print, with zero suppression, the four-digit decimal number in registers 4 through 7.
BCD	11	(1-7, 10-15) This routine converts the 12-bit binary number in registers 1-3 into decimal, and prints the four digits with zero suppression.

RAM USAGE

The simulation program, to facilitate full usage of the RAM, has organized it into a nominal block of 64 registers, each containing 16 main memory locations and four status locations. Directives which reference RAM as such (i.e., Q, M, and D), always address it by a register number, and sometimes by a character position within the register. The following chart illustrates this addressing scheme:

REGISTER (selected by even index in SRC instr.)	DIGIT (selected by odd index in SRC instr.)																[Status]			
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Bank 0 RAM 0 (Port 0) — 0																				
1																				
2																				
3																				
Bank 0 RAM 1 — 4																				
5																				
6																				
7																				
8																				
9																				
•																				
•																				
•																				
Bank 1 RAM 0 (Port 4) — 16																				
17																				
18																				
19																				
20																				
•																				
•																				
•																				
Bank 2 RAM 3 (Port 11) — 44																				
45																				
46																				
47																				
Bank 4 RAM 0 (Port 12) — 48																				
49																				
50																				
51																				
52																				
•																				
•																				
•																				
Bank 4 RAM 3 (Port 15) — 60																				
61																				
62																				
63																				

The bank number in the chart above is the value of the accumulator during a DCL instruction, needed to address that bank of RAMs. The port number given corresponds to the number typed out during simulation of the WMP instruction. Register positions 16-19 (i.e., the status locations) are normally addressed in the program by the RD0/WR0, RD1/WR1, etc., instructions, respectively.

The Q directive is used to define and set aside some part of RAM for use by the simulation program as simulated ROM and other registers and parameters. Whole RAM registers are taken by the Q directive, beginning with the register identified in the first parameter. The status locations from exactly 12 registers are used by the simulator. The number of main memory locations used is determined by the difference between the second and third parameters of the Q directive. Where **s** and **e** represent the values in the second and third parameters of the Q directive, the number of registers used is determined by the formula: $n = (s + e)/8 + 1$
This value may be more or less than 12, the number of registers whose status locations are used, with no ill effects.

RAM main memory locations reserved by the Q directive are used solely for program storage. The instruction with an address equal to the second parameter of the Q instruction is loaded into digits 0 and 1 of the register designated by the first parameter of the Q directive; subsequent instructions are loaded into the following digit pairs, according to the addresses.

The RAM status locations reserved by the simulation program are allocated to the following functions:

Relative REGISTER	LOC'N.	FUNCTIONS
r + 0	0	Simulated Accumulator
0	1-3	Low ROM address limit
1	0	Option word
1	1-3	High ROM address limit
2	0	Execution parameters
2	1-3	Breakpoint address
3	0	Simulated Carry
3	1-3	Simulation Cycle counter
4	0	Simulated Stack pointer
4-7	1-3	Simulated Stack
5	0	Simulated Command line selection
6,7	0	Simulated ROM or RAM chip selection
8-11	0-3	Simulated index registers

EXAMPLES

Figures 1, 2, and 3 are annotated listings generated during actual simulation. Figure 1 is a simulation of the "AND" program described in figures 4 and 5. Figures 2 and 3 represent the simulator's response to various directives entered via a TTY keyboard.

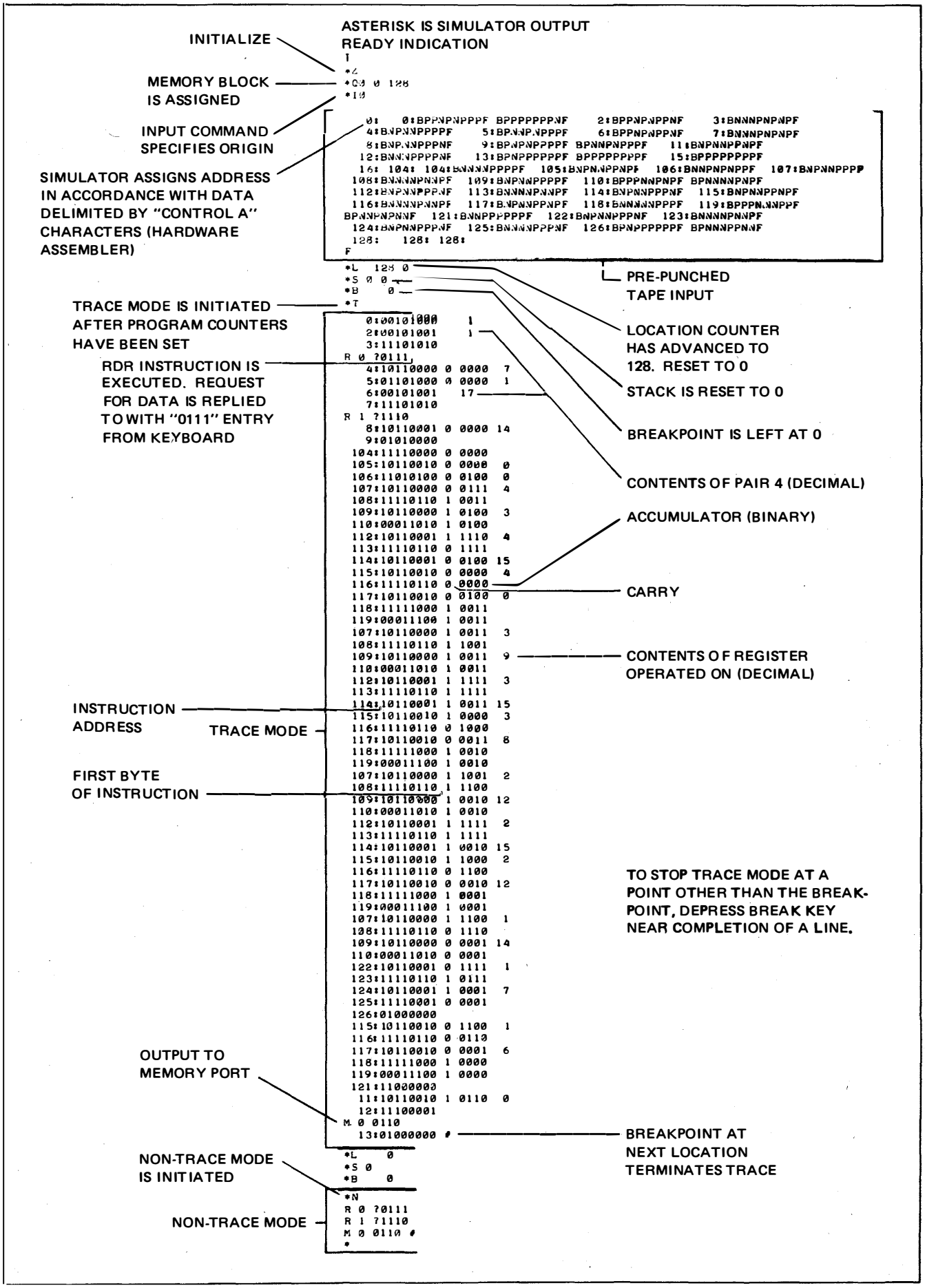


Figure 1. Trace and Nontrace Modes.

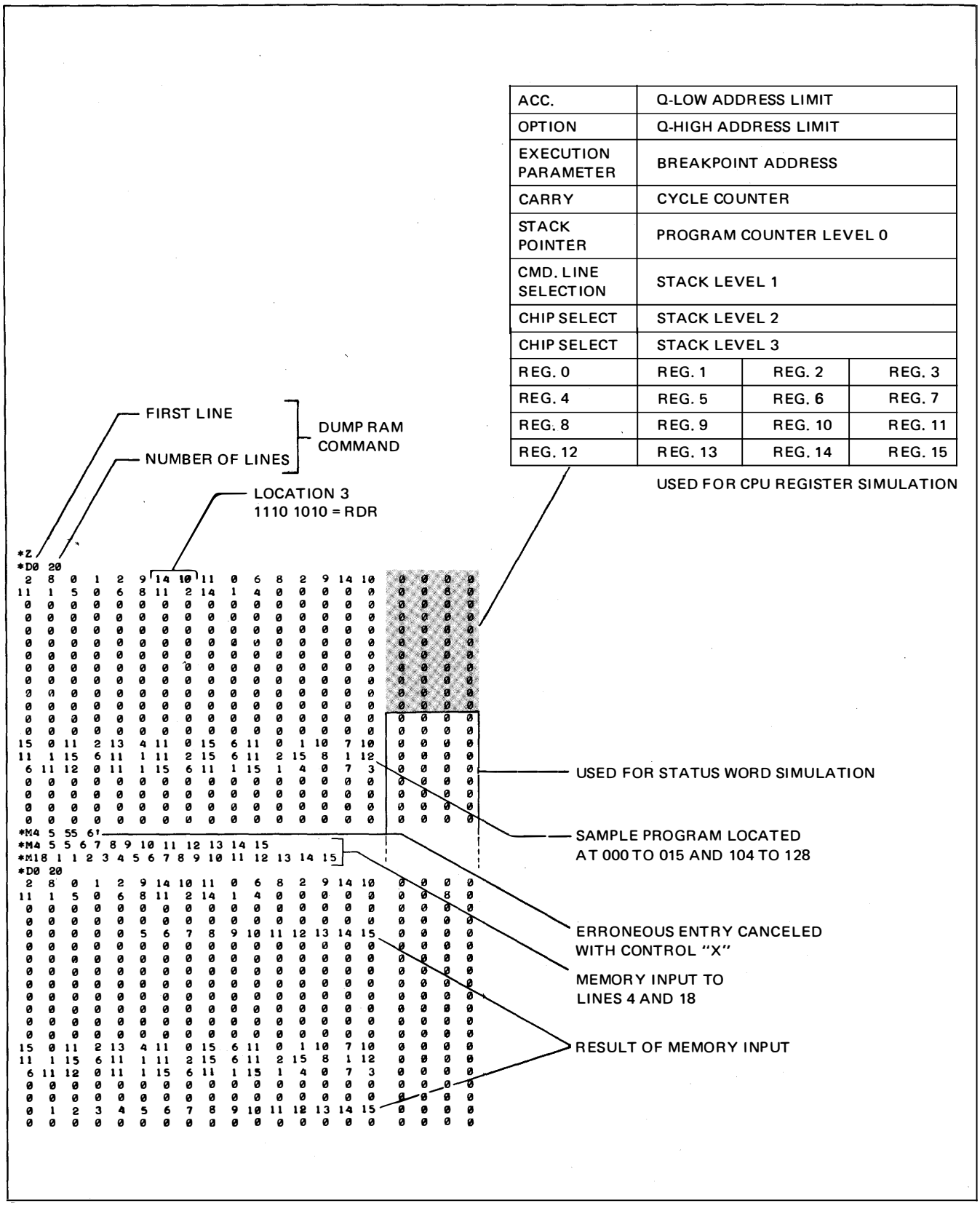


Figure 2. Memory Dump/Input.

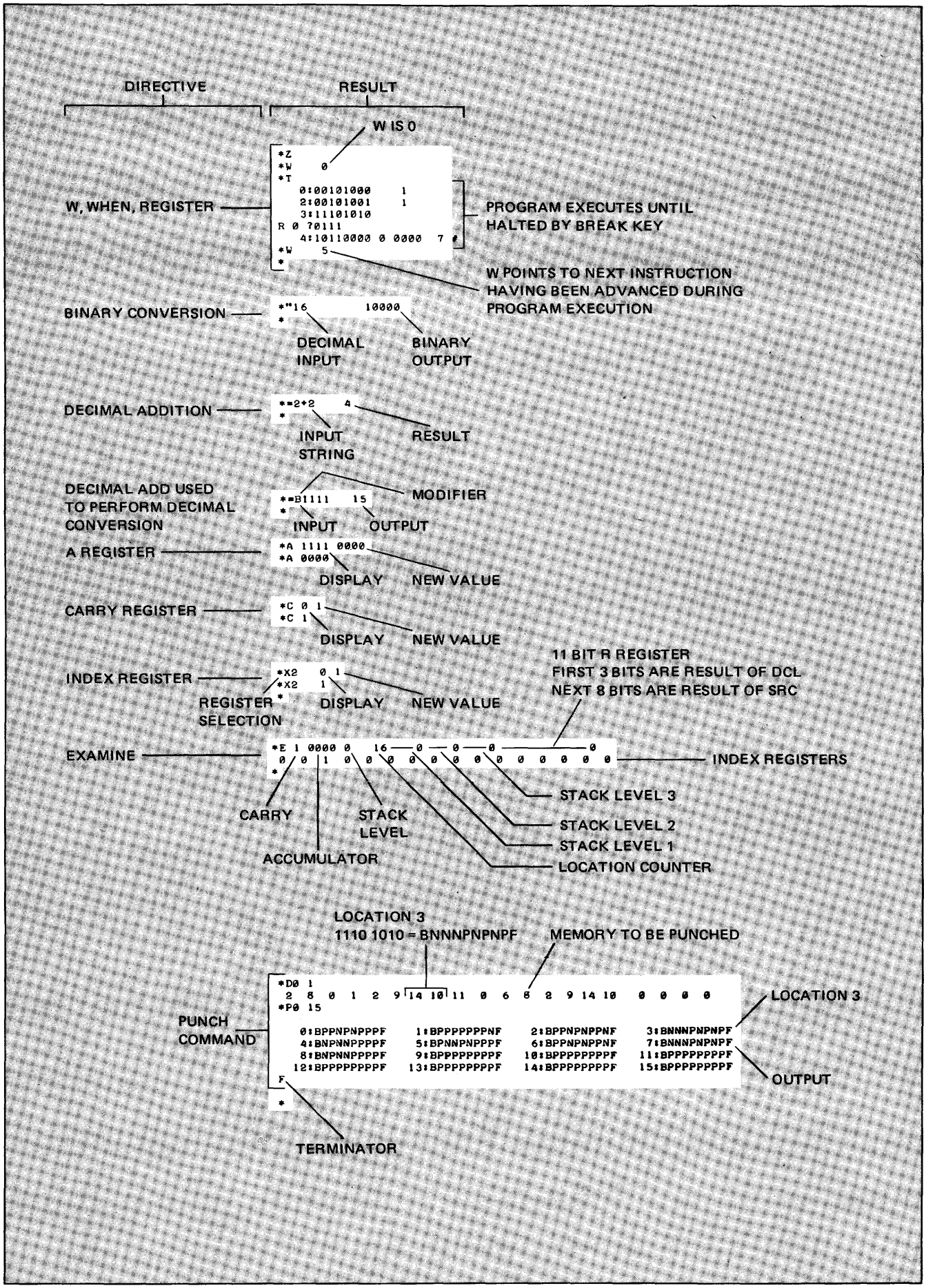


Figure 3. Miscellaneous Directives.

```
   0:/                  FOUR BIT "AND" ROUTINE
   0:START, FIM 4P 0     /  LOAD ROM PORT 0 ADDRESS
   2: SRC 4P             /  SEND ROM PORT ADDRESS
   3: RDR                /  READ INPUT A
   4: XCH 0              /  A TO REGISTER 0
   5: INC 8              /  LOAD ROM PORT 1 ADDRESS
   6: SRC 4P             /  SEND ROM PORT ADDRESS
   7: RDR                /  READ INPUT B
   8: XCH 1              /  B TO REGISTER 1
   9: JMS AND            /  EXECUTE "AND"
  11: XCH 2              /  LOAD RESULT C
  12: WMP                /  STORE AT MEMORY PORT 0
  13: JUN START          /  RESTART
  15: NOP
  16: =104
 104:/                  "AND" SUBROUTINE
 104:AND, CLB           /  CLEAR ACCUMULATOR AND CARRY
 105: XCH 2             /  CLEAR REGISTER 2
 106: LDM 4             /  LOAD LOOP COUNT (LC)
 107: XCH 0             /  LOAD A, LC TO REGISTER 0
 108: RAR               /  ROTATE LEAST SIGNIFICANT BIT TO CARRY
 109: XCH 0             /  RETURN ROTATED A TO REG 0, LC TO ACC.
 110: JCN CZ ROTR1      /  JUMP TO ROTR1 IF CARRY ZERO
 112: XCH 1             /  LOAD B, LC TO ACCUMULATOR
 113: RAR               /  ROTATE LEAST SIGNIFICANT BIT TO CARRY
 114: XCH 1             /  RETURN ROTATED B TO REG. 1, LC TO ACC.
 115:ROTR2, XCH 2       /  LOAD PARTIAL RESULT C, LC TO REGISTER 2
 116: RAR               /  ROTATE CARRY INTO PARTIAL RESULT MSB
 117: XCH 2             /  LOAD LC, RETURN C TO REGISTER 2
 118: DAC               /  DECREMENT THE ACCUMULATOR (LC)
 119: JCN ANZ AND+3     /  LOOP IF LC NON ZERO
 121: BBL 0             /  RETURN
 122:ROTR1, XCH 1       /  LOAD B, LC TO REGISTER 1
 123: RAR               /  ROTATE B
 124: XCH 1             /  RETURN ROTATED B TO REG. 1, LC TO ACC.
 125: CLC               /  CLEAR CARRY
 126: JUN ROTR2         /  RETURN TO LOOP
 128:CZ  =10
 128:ANZ  =12
 128:$
```

Figure 4. Pass 1 Listing.

```
   0:    0:BPPNPNPPPF BPPPPPPPNF     2:BPPNPNPPNF     3:BNNNPNPNPF
   4:BNPNNPPPPF     5:BPNNPNPPPF     6:BPPNPNPPNF     7:BNNNPNPNPF
   8:BNPNNPPPNF     9:BPNPNPPPPF BPNNPNPPPF    11:BNPNNPPNPF
  12:BNNNPPPPNF    13:BPNPPPPPPF BPPPPPPPPF    15:BPPPPPPPPF
  16: 104: 104:BNNNNPPPPF  105:BNPNNPPNPF  106:BNNPNPNPPF  107:BNPNNPPPP
 108:BNNNNPNNPF  109:BNPNNPPPPF  110:BPPPNNPNPF BPNNNNPNPF
 112:BNPNNPPPNF  113:BNNNNPNNPF  114:BNPNNPPPNF  115:BNPNNPPNPF
 116:BNNNNPNNPF  117:BNPNNPPNPF  118:BNNNNNPPPF  119:BPPPNNNPPF
BPNNPNPNNF  121:BNNPPPPPPF  122:BNPNNPPPNF  123:BNNNNPNNPF
 124:BNPNNPPPNF  125:BNNNNPPPNF  126:BPNPPPPPPF BPNNNPPNNF
 128: 128: 128:
F
```

Figure 5. Programming Tape Listing.

APPENDIX H
MCS-4™ ASSEMBLER/SIMULATOR SOFTWARE PACKAGE

This appendix describes the use and operation of the assembler and simulator software package for the MCS-4 micro computer set. This assembler-simulator package is offered through Tymshare, Inc., G.E. Timeshare, and AL/COM, nationwide computer timesharing services. The software package allows the user to: (1) prepare and edit his program; (2) assemble these programs into MCS-4 compatible binary code; (3) simulate the MCS-4 system's execution of this code; and (4) generate tapes in a format suitable for programming 1602A or 1702A field programmable and erasable read-only memories. The output tapes so generated can also be delivered to Intel for preparation of masks for the 4001 mask-programmed read-only memory.

The assembler performs a number of error diagnostics, checking for code overlapping, invalid mnemonics, illegal off-page references and several other common errors.

The simulator allows one to operate in a trace mode or to introduce a break point in a program and to interrogate the contents of registers when the program arrives at the break point.

The MCS-4 Assembly Language

Table 1 lists the instructions associated with the MCS-4 computer set. Listed for each instruction is the standard mnemonic code used to describe that instruction to the assembler and its binary equivalent. The number of bytes (one or two) occupied by the instruction and the types of modifiers which must accompany the instruction are also indicated in Table 1. The types of modifiers required may be none, as in the case of NOP, CLB, CLC, etc.; a register designator indicating one of the sixteen registers on the central processor chip as in the case of Add, Load, Exchange, etc.; a 4-bit data item as in the case of the BBL or LDM instruction, an 8-bit data item as in the case of the FIM instruction; a register pair designator as in the case of the SRC or JIN instruction; an 8-bit address as in the case of the ISZ, or a 12-bit address as in the case of the JUN instruction. In general, such addresses will be expressed symbolically by use of labels.

The JCN, FIM and ISZ instructions require two modifiers. The JCN must include a condition code in addition to the address code. The ISZ instruction must include a single register designator as well as address code, while the FIM requires a register pair designation and an 8-bit data item. In each case, the modifiers must occur in the sequence shown with address or data items following condition codes, register or register pair designations.

For use by the assembler, numeric data may be entered in binary, decimal or octal format. See below under "Pseudo Operators".

TABLE 1.

INSTRUCTION	MNEMONIC	BINARY EQUIVALENT 1st byte	2nd byte	MODIFIERS
No Operation	NOP	ØØØØØØØØ	-	none
Jump Conditional	JCN	ØØØ1CCCC	AAAAAAAA	condition, address
Fetch Immediate	FIM	ØØ1ØRRRØ	DDDDDDDD	register pair, data
Send Register Control	SRC	ØØ1ØRRR1	-	register pair
Fetch Indirect	FIN	ØØ11RRRØ	-	register pair
Jump Indirect	JIN	ØØ11RRR1	-	register pair
Jump Unconditional	JUN	Ø1ØØAAAA	AAAAAAAA	address
Jump to Subroutine	JMS	Ø1Ø1AAAA	AAAAAAAA	address
Increment	INC	Ø11ØRRRR	-	register
Increment and Skip	ISZ	Ø111RRRR	AAAAAAAA	register, address
Add	ADD	1ØØØRRRR	-	register
Subtract	SUB	1ØØ1RRRR	-	register
Load	LD	1Ø1ØRRRR	-	register
Exchange	XCH	1Ø11RRRR	-	register
Branch Back and Load	BBL	11ØØDDDD	-	data
Load Immediate	LDM	11Ø1DDDD	-	data
Write Main Memory	WRM	11100000	-	none
WRite RAM Port	WMP	11100001	-	none
Write ROM Port	WRR	11100010	-	none
Write Status Char Ø	WRØ	11100100	-	none
Write Status Char 1	WR1	11100101	-	none
Write Status Char 2	WR2	11100110	-	none
Write Status Char 3	WR3	11100111	-	none
Subtract Main Memory	SBM	11101000	-	none
Read Main Memory	RDM	11101001	-	none
Read ROM Port	RDR	11101010	-	none
Add Main Memory	ADM	11101011	-	none
Read Status Char Ø	RDØ	11101100	-	none
Read Status Char 1	RD1	11101101	-	none
Read Status Char 2	RD2	11101110	-	none
Read Status Char 3	RD3	11101111	-	none
Clear Both	CLB	11110000	-	none
Clear Carry	CLC	11110001	-	none
Increment Accumulator	IAC	11110010	-	none
Complement Carry	CMC	11110011	-	none
Complement	CMA	11110100	-	none
Rotate Left	RAL	11110101	-	none
Rotate Right	RAR	11110110	-	none
Transfer Carry and Clear	TCC	11110111	-	none
Decrement Accumulator	DAC	11111000	-	none
Transfer Carry Subtract	TCS	11111001	-	none
Set Carry	STC	11111010	-	none
Decimal Adjust Accumulator	DAA	11111011	-	none
Keyboard Process	KBP	11111100	-	none
Designate Command Line	DCL	11111101	-	none

NOTE: The MCS-4 Assembler is currently being modified to accept the WPM instruction associated with the 4008/4009.

A register pair may be designated by including the numeric value corresponding to the even numbered register of the pair, by a P followed immediately by the number (Ø through 7) of the register pair in decimals, or by a P followed immediately by a 3-bit binary code. Examples of acceptable forms are: Ø, 2, PØØ1, P1, P7. A register may be designated by including the register number or by the form RØ through R15.

A condition code may be satisfied by the inclusion of a numeric value corresponding to the desired condition code, or by using one of the condition codes shown in Table 2. Note that Table 2 does not include all possible condition codes for the JCN instruction.

A label is a tag attached to a particular line of code. The label will take on value corresponding to the address assigned to that line of code by the assembler. To associate a tag with a line of code, the tag or label is made the first item of the line and is followed by a comma. Figure 1 below shows examples of labelled and unlabelled instructions. Addresses, as used with the JUN, ISZ or JCN instructions and data, as used with the FIM instruction, can refer to these labels. Labels attached to double word instructions always refer to the address of the first word.

Acceptable forms for use as addresses in this assembler are a numeric value, a label, or a label plus or minus a numeric value. However, when the form label plus or minus a numeric value is used, the numeric value corresponds to the number of bytes displaced from the label, not the number of instructions. Thus, one must remember which instructions occupy one byte and which occupy two bytes when using this displaced form. The form * plus or minus a numeric value refers to a displacement equal to the numeric value from the current address.

For more detailed descriptions of the functions of each of the instructions of the MCS-4 instruction set, the user is referred to the MCS-4 Micro Computer Set User's Manual.

Pseudo Operators

A number of pseudo operations are available in the MCS-4 assembler system. For example, the number system used with the MCS-4 assembler is initially decimal. However, the user may change the number system to be used by inserting one of the three pseudo operators: .B for binary, .O (letter O) for octal, .D for decimal. For example, the pseudo operator, .B, will signal the assembler to interpret any number (not otherwise indicated as being in another number system) as a binary number.

At any point in the program, the current number system can be overridden by following the typed number with the letter B, D,

TABLE 2

Condition Codes for JCN Instruction

Condition (Jump If:)	Mnemonic*	Binary Equivalent
no condition	NC	0000
test equals zero	TZ, TØ	0001
test equals one	TN, T1	1001
carry equals one	CN, C1	0010
carry equals zero	CZ, CØ	1010
accumulator equals zero	AZ, AØ	0100
accumulator non zero	AZ, NZA	1100

*When more than one mnemonic is shown, any one of the forms is acceptable.

a.)

```
        L1,     CLB
                DCL
```

b.)

```
                JCN     TZ      *
```

c.)

```
        TAG,    JMS     L1+1
```

Figure 1., Examples of labelled and unlabelled lines

a.) L1 is a label, but the second line is unlabelled.
b.) The line is unlabelled, but the * implies that the jump occurs back to this instruction as long as the CPU test line is at a logic zero.
c.) TAG is a label, and the Jump to subroutine will enter the routine at L1+1 (ie., the DCL instruction of Figure 1a.)

or O in the position immediately following the last digit of the number. This input will not alter the number system used for numbers not so labelled.

Numbers designating register pairs or registers of the form RO-R15 PØØØ-P111, PØ-P7 are not subject to this number system input convention.

If the assembler comes across a number which will not fit within the number system chosen, it will indicate an error.

To set the starting address for a group of instructions, the group may be preceded by a define origin pseudo operator. This pseudo operator takes the form of an asterisk (*) followed by the numeric value of the desired address. This numeric value follows the number sytem conventions described above.

Figure 2 shows how number systems, pseudo-operators, and comments are used with the assembler.

Any statements following a slash (/) are interpreted as comments They do not alter or affect the other instructions in the system. A comment line may stand alone or a comment may be appended at the end of a line.

No end statement is necessary. The end of the file signifies the end of the program.

```
          .D
          *1024                    /DECIMAL 1024
STRT,     LDM  1101B               /BINARY 1101 = DECIMAL 13
          FIM  P1ØØ   100          /REGISTERS 8,9 DATA=100 DECIMAL
    /THE NEXT PSEUDO OPERATOR CHANGES THE NUMBER SYSTEM
    /TO BINARY
          .B
          LD   1100                /REGISTER 12
          XCH    4D                /REGISTER  4
```

Figure 2. Example of Pseudo Operators and Comments

Running the Assembly

Once the source file with the MCS-4 assembly language code has been prepared, it may be assembled into MCS-4 binary code by running the assembler. To perform this operation, the user should make sure he is in PDP-10 monitor mode. He then types RUN (*) ASF4↵. The system will then indicate that the assembler program is being loaded. Once execution of the assembly begins, the program will type the following:

SOURCE NAME FILE =

The user should then type the name of the file containing his symbolic code followed by a carriage return. As all valid names are of the form: name.DAT, only the portion left of the period is typed. The assembler will then request whether or not the user desires a listing of his program as it is assembled by typing the statement: LIST?(Y,N): If you wish to have a listing generated, type Y. If the user desires to have a listing generated, the assembler program will then request which output device, the disk, teletype or line printer, the output is to be generated on. The user should then type the three letter code corresponding to the desired output device (DSK, TTY, LPT) followed by a carriage return.

If the disk has been chosen as the output device, the program will then request a name for the output file. The system will also request which number system is to be used for the address and data portion of the output assembly. Binary, octal or decimal may be chosen by typing B, O, or D respectively. Note that this option has no affect on the number system options within the assembly code.

Error Messages

If, in the course of an assembly, the error diagnostic routines discover errors within the user's source program; error messages will be typed, followed by the erroneous line of code. The error message will include the line number on which the error occurred and the address where the error occurred. In some cases, the line number or address may be off by one or two positions. After typing the error message and the offending line of code, the program prints a star. After the star appears, the user may type in a corrected line of code if he wishes. This corrected line will be used by the assembler when generating the output listing.

Error messages in most cases appear before the listing is generated.

The most common types of errors are invalid characters within the input record, invalid mnemonics or modifiers (such as pair designators, condition codes, etc.), the use of incorrect or undefined labels or illegal off-page references, or the use of numbers which are invalid in the number system chosen.

Once the assembler has completed its assembly and generated a listing, the assembler will type a message requesting whether or not a symbol or label table is required. The user should

type a Y or an N immediately following the message and then the carriage return. If a symbol table has been chosen, it will be output on the user teletype. The numeric value assigned to each label is also output with the symbol table. The number system to be used for this numeric output is requested prior to typing the symbol table.

After the symbol table has been typed, the assembler will type the message: 1601 OUTPUT? Y,N: This message requests whether the user wishes to generate tapes of a form compatible with programming 1601, 1602, 1701 or 1702 field programmable and field erasable read-only memories. If the user wishes to generate such tapes at this time, he types Y. The program will then request what output device he wishes the tapes to be generated on. The choices are: the paper tape punch located at the computer center, the user's teletypewriter, or the disk. If the disk is selected, the program will request an output file name.

Following the request for an output file name, if any, the program will type: PAGE NO., T OR C =. The user may then type the number of the page (ROM chip) for which he wishes to generate the programming tape. Page 0 corresponds to addresses Ø through 255, page 1 to addresses 256 through 511, etc., so the user must type a 1 or 2 digit decimal number in the range Ø-15. This must be followed by a comma or a space, followed in turn by the letter T or C, indicating whether the page is to be output in true (1=P, Ø=N) or complemented (1=N, Ø=P) form. For example, tapes for use with the SIM4-01 and SIM4-02 boards should be prepared in complemented form. After each tape has been generated, the program will request another page number. In the case where output is to disk, another file name will be requested prior to the page number.

If no more tapes are desired, typing any character other than a valid page number will cause the program to exit. Upon exiting, the program will print the amount of central processor time and the amount of terminal time used for the assembly.

Figure 3 shows an example of the TTY output for a very small sample program. Only the first 36 words of 1601 tape are shown.

```
SOURCE FILE NAME=TST4

LIST?(Y,N):Y

OUTPUT TO?(DSK,TTY,LPT):TTY

NUMBER SYSTEM?(B,O,D):B

ASSEMBLY OF TST4 .DAT ON 17-MAY-72 AT 13:17


        .D
        *16
000000010000 11110000   LABA,   CLB
000000010001 00101010            FIM P5 89 /NOTE COMMENT
             01011001
000000010011 00101011            SRC P5
000000010100 11101010   LAB2,    RDR
000000010101 00011001            JCN T1 LAB2
             00010100
000000010111 01010000            JMS TR3
             00011011
000000011001 01000000            JUN LABA
             00010000
000000011011 11011000   TR3,     LDM 8
000000011100 10001011            ADD 11
000000011101 10110101            XCH 5
000000011110 11000111            BBL 7
SYMBOL TABLE (Y,N)?Y

NUMBER SYSTEM?(B,O,D):D

 LAB2  000020          LABA  000016          TR3   000027
1601 OUTPUT?(Y,N):Y

OUTPUT TO?(PTP,TTY,DSK):TTY

PAGE NO.,T OR C=0,T


   0 BNNNNNNNNF BNNNNNNNNF BNNNNNNNNF BNNNNNNNNF
   4 BNNNNNNNNF BNNNNNNNNF BNNNNNNNNF BNNNNNNNNF
   8 BNNNNNNNNF BNNNNNNNNF BNNNNNNNNF BNNNNNNNNF
  12 BNNNNNNNNF BNNNNNNNNF BNNNNNNNNF BNNNNNNNNF
  16 BPPPPNNNNF BNNPNPNPNF BNPNPPNNPF BNNPNPNPPF
  20 BPPPNPNPNF BNNNPPNNPF BNNNPNPNNF BNPNPNNNNF
  24 BNNNPPNPPF BNPNNNNNNF BNNNPNNNNF BPPNPPNNNF
  28 BPNNNPNPPF BPNPPNPNPF BPPNNNPPPF BNNNNNNNNF
  32 BNNNNNNNNF BNNNNNNNNF BNNNNNNNNF BNNNNNNNNF
```

Figure 3.

Operation of SIM (MCS-4 Simulator)

The MCS-4 simulator program SIM allows the user to simulate execution of programs which have been assembled by the assembler ASF4. ASF4 generates a file ROMAR.DAT containing a packed image of the read-only memory contents. ROMAR.DAT is read by SIM as an input.

Other files in the same format may be read prior to ROMAR.DAT. Each successive file overlays the data of preceeding files, except where zeroes (NOP's) occur in the new file. See below.

To run the simulator, after having assembled a program, type RUN (*) SIM4↵. After typing a header message, the simulator types:

FOR INSTRUCTION LIST, TYPE Q:

If the user types the letter Q followed by a carriage return, the program responds by typing a list of recognized commands. The simulator indicates it is ready for input commands by typing an asterisk (*) at the left side of the teletype page.

A command to the simulator consists of a letter or a letter followed by a 1 to 5 digit number. The letter must be the first character typed. Each letter corresponds to a different command. Commands allow setting a starting address or a break point, initializing a trace mode, or interrogating registers. The following commands are recognized by SIM4:

An Set the program counter of the simulated MCS-4 to n and then return to command mode.

Bn Set the break point at n and return to command mode.

Cn Call a subroutine starting at ROM address n. Continue execution until the subroutine exits with a BBL or until the instruction limit counter overflows.

D Dump read/write memory.

En Examine RAM word n. (n must be between 0 and 63. Words 0-15 are in bank Ø, words 16-31 are in bank 2, etc.) When printed, main character #15 will be printed first, followed by the remaining main characters of the word in descending order. A space follwed by status characters Ø-31 in ascending order is printed after the main characters. Output is in hexadecimal with A-F corresponding to 1Ø-15.

Fn If n=1, read only memory input port data is read from a file rather than from the teletype. Data in the file must consist of a single numeric value per record, with the numeric value lying between Ø and 15 (decimal). If n≠1, the input port data will be requested from the teletype-writer.

H Prints a two-line leader record corresponding to the data printed during trace mode or printed upon reaching break points, BBL exits, or overflowing the instruction limit counter.

In Re-initializes the program counter to 0 and clears read/write memory. If n≠Ø, the instruction limit counter is

set to n. If n=Ø, the instruction limit counter is left unchanged. (The initial value for the instruction limit counter is 10,000.)

Jn Jump to and begin execution at ROM address n. Stop either when the break point is reached or the instruction limit counter overflows.

Ln Operate trace mode with output to the line printer rather than TTY (not available on time-shared versions).

Mn Input to RAM word n. (n must be between Ø and 63). The terminal will respond by requesting 20 characters. Exactly 20 must be typed, in the same sequence as used for the "E" command above. However, no space is allowed between main characters and status characters. The inputs allowed are Ø-9, A-F and -. Ø-F correspond to hexadecimal inputs of Ø through 15, while a typed - leaves the corresponding character position unchanged.

On If n=1, a shorter form of input and output will be used for ROM input port requests, etc. If n≠1, the long form will be used.

R Examine CPU registers.

S Examine stack and stack pointers.

Tn Operate trace mode for the next n instructions, starting at the current program counter value. Break points will terminate the sequence, but the instruction limit counter is disabled.

Un Execute n instructions, then stop and print the status information for the last instruction. Break points will terminate the sequence, but the instruction limit counter is disabled.

Wn Write into ROM word n. (n must fall between Ø and 4095). The terminal will request a numeric value (in decimal) between Ø and 255 which will be written into the selected ROM word.

X Exit from the program.

(All commands are followed by a typed carriage return. In each case, n represents a 1-5 digit decimal number.)

For example, to run trace mode for 50 instructions starting at address 100 (decimal), the user may type:

```
A100↵
T50↵
```

or he might type:

```
B100↵
J0↵
T50↵
```

in each case, waiting for a * before typing the next command line. In the second case, the instructions between Ø and 100 are executed where, in the first case, they are not.

At a break or in trace mode, the contents of CPU registers are displayed. The following sequence is used for long form output:

Program counter	4 digit decimal
Instruction and	3 character symbolic OP code
Modifiers	2 characacter modifier
	4 character address or data
CY FF	1 digit (0,1)
Accumulator	2 digit decimal (Ø-15)
Registers	16 hexadecimal characters in order RØ to R15
RAM Bank No.	1 digit decimal
RAM Register No.	2 digit decimal
RAM Main Char. No.	2 digit decimal
Cycle counter	5 digit decimal

Short form output omits the modifier and RAM data.

When an RDR instruction (read ROM port) is executed, the simulator types:

```
INPUT REQUESTED FROM ROM PORT #N (IN DEC)
*
```

The user types a one or two digit decimal value corresponding to the 4 lines of the input port. When short form messages are used, the request message is:

```
ROMPT INPUT #N?
*
```

Note on use of input files

The simulator program will request an input file name with the message:

```
INPUT FILE NAME=.
```

To input the standard data file ROMAR.DAT, merely type a carriage return.

However, it is possible to break a program into segments and load several of the segments into the simulator one at a time. As each segment is assembled, the assembler will produce the assembled code in the file ROMAR.DAT. To save the segment, so that it will not be destroyed by the next assembly, the user must rename the ROMAR.DAT file with some other name with the extension .DAT.

The last segment assembled is left in the ROMAR.DAT file.

When the simulator requests the input file name, each segment can be loaded by typing the file name. For example, a file SEG1.DAT is loaded by typing SEG1 when the file name is requested. After each such named data file is loaded, the program will request another file. The last file, ROMAR.DAT, is loaded by typing just the carriage return.

Each file so loaded overlays all previously loaded data, except where the newly loaded file contains zeroes (NOP's). If any previously non zero locations are changed by the loading of the file, the addresses of the first and last changed locations are typed.

When assembling a program in segments, care must be used to provide sufficient label information for each segment. Because NOP's do not cancel previously loaded data, dummy labels may be used as in the following example:

Suppose a program must call a subroutine LB1 at address 1100 decimal, which has been assembled in a previous segment. The proper linkage can be established in the current segment by including the pair of statements:

```
      *1100D
LB1,  NOP
```

or the pair

```
      *1100D
LB1,  Ø
```

APPENDIX I. MCS-4 PROGRAMMING EXAMPLE — CONTROL PROGRAM FOR PROM PROGRAMMING

Intel tapes A0540, A0541, A0543 — Listings are provided for reference only and may be used as examples when developing your own programs.

```
0000 00209      START,  LDM 1   / SUPPRESS TTY
0001 00032              FIM P0 0
     00000
0003 00033              SRC P0
0004 00225              WMP     / SEND TO RAM PORT
0005 00080      STA1,   JMS CRLF        /POSITION POINTER
     00226
0007 00080              JMS ST  / RECIELVE INPUT DATA
     00240
0009 00038              FIM P3 80       /IR(6)=5, IR(7)=0, (P)
     00080
0011 00081              JMS COMPR
     00033
0013 00028              JCN AN LISTN
     00032
0015 00032              FIM P0 32       /IR(0)=2, IR(1)=0
     00032
0017 00209              LDM 1
0018 00033              SRC P0
0019 00230              WR2
0020 00064              JUN STA2
     00045
0022 00038      NEXT,   FIM P3 83       /IR(6)=5, IR(7)=3, (S)
     00083
0024 00081              JMS COMPR
     00033
0026 00020              JCN AZ STA2
     00045
0028 00080              JMS ERROR
     00135
0030 00064              JUN STA1
     00005
0032 00038      LISTN,  FIM P3 76       /IR(6)=4, IR(7)=12, (L)
     00076
0034 00081              JMS COMPR
     00033
0036 00028              JCN AN NEXT
     00022
0038 00220              LDM 12
0039 00191              XCH 15  / IR(15)=15
0040 00032              FIM P0 32       /IR(0)=2, IR(1)=0
     00032
0042 00209              LDM 1
0043 00033              SRC P0
0044 00229              WR1
0045 00080      STA2,   JMS CRLF
     00226
0047 00222              LDM 14  / ENTER ADDRESS
0048 00196              XCH 10  / IR(10)=14
0049 00221      ADRS2,  LDM 13
0050 00185              XCH 9   / IR(9)=13
0051 00080      ADRS1,  JMS ST
     00240
0053 00080              JMS STORE
     00140
0055 00121              ISZ 9 ADRS1
     00051
0057 00080              JMS CRLF
     00226
0059 00122              ISZ 10 ADRS2
     00249
0061 00080              JMS LF
     00230
0063 00080              JMS LF
     00230
0065 00081              JMS ADDTN
     00046
0067 00032              FIM P0 0
     00000
0069 00080              JMS OBIN        /FINAL ADDRESS
     00105
0071 00163              LD 3
0072 00230              WR2     / C(0),R(0),CH(2),LORD 4-BITS
0073 00162              LD 2
0074 00231              WR3     / C(0),R(0),CH(3),HIORD 4-BITS
0075 00032              FIM P0 3        /IR(0)=0, IR(1)=3
     00003
0077 00080              JMS OBIN        /INITIAL ADDRESS
     00105
0079 00163              LD 3
0080 00228              WR0     / C(0),R(0),CH(0),LORD 4-BITS
0081 00162              LD 2
0082 00229              WR1     / C(0),R(0),CH(1),HIORD 4-BITS
0083 00036      PRGM1,  FIM P2 0        / IR(4-5)=0
     00000
0085 00032              FIM P0 0        / ADDRESS 1701 READ TO WRITE
     00000
0087 00033              SRC P0
0088 00236              RD0
0089 00226              WRR
0090 00100              INC 4
0091 00237              RD1
0092 00037              SRC P2
0093 00226              WRR
0094 00032              FIM P0 32       /IR(0)=2, IR(1)=0
     00032
0096 00033              SRC P0
0097 00237              RD1
0098 00246              RAR
0099 00018              JCN CN NEXT1
     00103
0101 00065              JUN REPT        / CALL FOR INPUT DATA
     00073
0103 00066      NEXT1,  JUN LISTR       /CALL LISTING ROUTINE
     00187
/ BCD TO BINARY CONVERSION ROUTINE
0105 00034      OBIN,   FIM P1 0        /IR(2)=0, IR(3)=0
     00000
0107 00036              FIM P2 10       /IR(4)=0, IR(5)=10
     00010
0109 00222              LDM 14
0110 00182              XCH 6
0111 00033              SRC P0
0112 00233              ROM
0113 00179              XCH 3
0114 00097      BDBN,   INC 1
0115 00033              SRC P0
0116 00233      BB1,    ROM
0117 00020              JCN AZ BB2
     00130
0119 00248              DAC
0120 00224              WRM
0121 00241              CLC
0122 00163              LD 3
0123 00133              ADD 5
0124 00179              XCH 3
0125 00162              LD 2
0126 00132              ADD 4
0127 00178              XCH 2
0128 00064              JUN BB1
     00116
0130 00036      BB2,    FIM P2 100      /IR(4)=6, IR(5)=4
     00100
0132 00118              ISZ 6 BDBN
     00114
0134 00192              BBL 0
/
/
0135 00034      ERROR,  FIM P1 191      /IR(2)=11, IR(3)=15, (?)
     00191
0137 00080              JMS PRINT
     00178
0139 00192              BBL 0
/
/
/
0140 00032      STORE,  FIM P0 11       /IR(0)=0, IR(1)=11
     00011
0142 00036              FIM P2 53       /IR(4)=3, IR(5)=5
     00053
0144 00038              FIM P3 52       /IE(6)=3, IR(7)=4
     00052
0146 00039      REP1,   SRC P3
0147 00233              ROM
0148 00037              SRC P2
0149 00224              WRM
0150 00165              LD 5
0151 00248              DAC
0152 00181              XCH 5
0153 00241              CLC
0154 00167              LD 7
0155 00248              DAC
0156 00183              XCH 7
0157 00241              CLC
0158 00113              ISZ 1 REP1
     00146
0160 00163              LD 3
0161 00037              SRC P2
0162 00224              WRM
0163 00192              BBL 0
/
/
/
/
/       TIMING SUBROUTINES
/
0164 00032      SBR1,   FIM P0 0        / IR(0-1)=0
     00000
0166 00112      L1,     ISZ 0 L1        /547 MEMORY CYCLES
     00166
0168 00113              ISZ 1 L1
     00166
0170 00192              BBL 0
/
/
0171 00032      SBR2,   FIM P0 8        / IR(0)=0,IR(1)=8
     00008
0173 00112      L2,     ISZ 0 L2        / 275
     00173
0175 00113              ISZ 1 L2
     00173
0177 00192              BBL 0
/
/
/       PRINT ROUTINE
/
0178 00032      PRINT,  FIM P0 16       /IR(0)=1, IR(1)=0
     00016
0180 00215              LDM 7
0181 00033              SRC P0
0182 00224              WRM
0183 00097              INC 1
0184 00179              XCH 3
0185 00033              SRC P0
0186 00224              WRM
0187 00097              INC 1
0188 00178              XCH 2
0189 00033              SRC P0
0190 00224              WRM
0191 00040              FIM P4 16       /IR(8)=1, IR(9)=0
     00016
0193 00034              FIM P1 208      / IR(2)=13,IR(3)=0
     00208
0195 00036      ST7,    FIM P2 12       / IR(4)=0,IR(5)=12
     00012
0197 00041              SRC P4
0198 00233              ROM
0199 00225      ST8,    WMP
```

```
0200 00180              XCH 4   / SAVE C(AC) IN INDEX REG 4
0201 00080              JMS SBR1        / DELAY ROUTINE #1
     00164
0203 00080              JMS SBR2        / DELAY ROUTINE #2
     00171
0205 00117              ISZ 5 ST9       / NUMBER OF ROTATIONS
     00218
0207 00105              INC 9   / NUMBER OF DIGITS
0208 00000              NOP
0209 00000              NOP
0210 00000              NOP
0211 00000              NOP
0212 00000              NOP
0213 00114              ISZ 2 ST7       / NUMBER OF 4-BIT WORDS
     00195
0215 00209              LDM 1
0216 00225              WMP
0217 00192              BBL 0
0218 00220      ST9,    LDM 12  / C(AC)=12
0219 00183              XCH 7   / C(7)=12
0220 00180              XCH 4   / RESTORE SAVED C(AC)
0221 00119      ST12,   ISZ 7 ST12
     00221
0223 00246              RAR
0224 00064              JUN ST8
     00199
/
/
/       CR/LF ROUTINE
/
0226 00034      CRLF,   FIM P1 141      /IR(2)=8, IR(3)=13, (CR)
     00141
0228 00080              JMS PRINT
     00178
0230 00034      LF,     FIM P1 138      /IR(2)=8, IR(3)=10, (LF)
     00138
0232 00080              JMS PRINT
     00178
0234 00192              BBL 0
/
/
/
/
/ ROUTINES FOR COMPARE PROGRAM
/
/
/ TELETYPE INPUT HANDLER
/
0235 00209      TTY,    LDM 1
0236 00032              FIM P0 64
     00064
0238 00033              SRC P0
0239 00225              WMP
0240 00017      ST,     JCN TZ ST       /WAIT FOR START BIT
     00240
0242 00032              FIM P0 64       / SET RAM 1 ADDRESS
     00064
0244 00033              SRC P0  / SEND RAM ADDRESS
0245 00208              LDM 0
0246 00225              WMP     / STOP READER
0247 00080              JMS SBR2
     00171
0249 00032              FIM P0 0
     00000
0251 00033              SRC P0
0252 00234              RDR
0253 00244              CMA
0254 00225              WMP
0255 00080              JMS SBR1        /TIME OUT START BIT
     00164
0257 00034              FIM P1 0        / RESET DATA LOCATION
     00000
0259 00216              LDM 8
0260 00180              XCH 4
0261 00080      ST1,    JMS SBR2        /5.6 MSEC. DELAY
     00171
0263 00241              CLC     / IT TREATS REG. PAIR 1 (2-3)
0264 00033              SRC P0  / AS IF IT WERE 1 8 BIT SHIFT
0265 00234              RDR     / REGISTER, TO INPUT DATA
0266 00244              CMA
0267 00225              WMP
0268 00246              RAR     / COMPLEMENT DATA TO TRUE STATE
0269 00162              LD 2    / AND LINK INTO REG 2 HIORD WORD
0270 00246              RAR
0271 00178              XCH 2
0272 00163              LD 3    / THEN LINK REG 2 OVERFLOW TO
0273 00246              RAR
0274 00179              XCH 3   / LORD WORD TO COMPLETE SHIFT
0275 00080              JMS SBR1
     00164
0277 00116              ISZ 4 ST1       / 8 DATA BITS ACCEPTED?
     00005
0279 00080              JMS SBR1
     00164
0281 00209              LDM 1
0282 00225              WMP     / RE-SUPPRESS TELETYPE
0283 00162              LD 2    / INPUT COMPLETE NOW
0284 00245              RAL
0285 00241              CLC     / ELIMINATE PARITY BIT
0286 00246              RAR
0287 00178              XCH 2
0288 00192              BBL 0   / EXIT WITH DATA INT PAIR 1 (2-3)
/
/
/
/
/ COMPARE ROUTINE, ACCEPTS INPUT IN PAIR 3
/ AND PAIR 1 WILL OUTPUT A (1) IF COMPARE
/ FAILS.
/
0289 00241      COMPR,  CLC             /DATA COMPARE ROUTINE
0290 00183              XCH 7
0291 00147              SUB 3
0292 00028              JCN AN NEG      /TEST FOR AC IS NOT 0
     00045
0294 00241              CLC
0295 00182              XCH 6
0296 00146              SUB 2
0297 00241              CLC
0298 00028              JCN AN NEG
     00045
0300 00192              BBL 0
0301 00193      NEG,    BBL 1
/
0302 00032      ADDTN,  FIM P0 0        /IR(0)=0, IR(1)=0
     00000
0304 00036              FIM P2 48       /IR(4)=3, IR(5)=0
     00048
0306 00218              LDM 10
0307 00182              XCH 6
0308 00241              CLC
0309 00037      AD1,    SRC P2
0310 00233              RDM
0311 00033              SRC P0
0312 00235              ADM
0313 00251              DAA
0314 00224              WRM
0315 00097              INC 1
0316 00101              INC 5
0317 00118              ISZ 6 AD1
     00053
0319 00192              BBL 0
/
0320 00208      CLRAM,  LDM 0
0321 00177              XCH 1
0322 00208      CLEAR,  LDM 0
0323 00037              SRC P2
0324 00224              WRM
0325 00101              INC 5
0326 00113              ISZ 1 CLEAR
     00066
0328 00192              BBL 0
/
/
/
/
/
/
/
0329 00038      REPT,   FIM P3 66       /IR(6)=4, IR(7)=2, (B)
     00066
0331 00080              JMS TTY / CALL INPUT ROUTINE
     00235
0333 00081              JMS COMPR       /CALL FOR COMPARE ROUTINE
     00033
0335 00028              JCN AN REPT     /TEST FOR NONZERO AC
     00073
0337 00216      REPTB,  LDM 8           /ACCEPT 8 DATA BITS
0338 00185              XCH 9
0339 00080      DATA1,  JMS TTY / LOOK FOR A DATA WORD
     00235
0341 00038              FIM P3 80       /IR(6)=5, IR(7)=0, (P)
     00080
0343 00081              JMS COMPR
     00033
0345 00020              JCN AZ CONT     /TEST FOR A MATCH
     00111
0347 00038              FIM P3 78       /IR(6)=4, IR(7)=14, (N)
     00078
0349 00081              JMS COMPR
     00033
0351 00020              JCN AZ CONT     /IF AC=0 PRINT
     00111
0353 00038              FIM P3 66       /IR(6)=4, IR(7)=2, (B)
     00066
0355 00081              JMS COMPR
     00033
0357 00020              JCN AZ REPTB
     00081
0359 00038              FIM P3 127      /IR(6)=7, IR(7)=5, (R0)
     00127
0361 00081              JMS COMPR
     00033
0363 00028              JCN AN FORMT
     00158
0365 00065              JUN RBOUT
     00144
0367 00032      CONT,   FIM P0 16       / DATA STORAGE
     00016
0369 00033              SRC P0
0370 00163              LD 3
0371 00245              RAL
0372 00237              RD1     / CHIP 0, REG. 1, HAR 1
0373 00245              RAL
0374 00229              WR1     / LEAST SIGNIFICANT BIT, DATA
0375 00236              RD0     / CHIP 0,REG. 1, CHAR 0
0376 00245              RAL
0377 00228              WR0     / MOST SIGNIFICANT BIT,DATA
0378 00121              ISZ 9 DATA1
     00083
0380 00080      FINAL,  JMS TTY / GET NEXT CHAR
     00235
0382 00038              FIM P3 70       /IR(6)=4, IR(7)=6, (F)
     00070
0384 00081              JMS COMPR
     00033
0386 00020              JCN AZ CONA     /AC=0, IF NOT RESTART INPUT
     00189
0388 00038              FIM P3 66       /IR(6)=4, IR(7)=2, (B)
     00066
```

```
0390 00081              JMS COMPR
     00033
0392 00020              JCN AZ REPTB
     00081
0394 00038              FIM P3 127      /IR(6)=7, IR(7)=15, (RO)
     00127
0396 00081              JMS COMPR
     00033
0398 00028              JCN AN FORMT    /CHECK FOR FORMAT ERROR
     00158
0400 00032      RBOUT,  FIM P0 16
     00016
0402 00033              SRC P0
0403 00236              RD0
0404 00246              RAR
0405 00228              WR0
0406 00237              RD1
0407 00246              RAR
0408 00229              WR1
0409 00169              LD 9
0410 00248              DAC
0411 00185              XCH 9
0412 00065              JUN DATA1
     00083
0414 00080      FORMT,  JMS CRLF        /POSITION CARRIAGE
     00226
0416 00082              JMS FMER
     00174
0418 00081      PRINL,  JMS PRINA
     00171
0420 00042              FIM P5 32       /IR(10)=2, IR(11)=0
     00032
0422 00209              LDM 1
0423 00043              SRC P5
0424 00228              WR0     / CHIP 0,REG. 2,CHAR 0
0425 00066              JUN STB
     00105
/
0427 00221      PRINA,  LDM 13          /PRINT FMERROR ADDRESS
0428 00188              XCH 12          /IR(12)=13
0429 00042              FIM P5 53       /IR(10)=3, IR(11)=5
     00053
0431 00043      FORM1,  SRC P5  / PRINT OUT ADDRESS
0432 00233              RDM
0433 00179              XCH 3
0434 00219              LDM 11
0435 00178              XCH 2
0436 00080              JMS PRINT
     00178
0438 00171              LD 11
0439 00248              DAC
0440 00187              XCH 11
0441 00241              CLC
0442 00124              ISZ 12 FORM1
     00175
0444 00192              BBL 0
/
0445 00032      CONA,   FIM P0 16
     00016
0447 00033              SRC P0
0448 00237              RD1
0449 00244              CMA
0450 00229              WR1
0451 00236              RD0
0452 00244              CMA
0453 00228              WR0
0454 00032              FIM P0 32       /IR(0)2, IR(1)=0, INIT ADR
     00032
0456 00033              SRC P0
0457 00238              RD2
0458 00246              RAR
0459 00018              JCN CN CONT2
     00247
0461 00032              FIM P0 0        /IR(0)=0, IR(1)=0, ADR COMP
     00000
0463 00033              SRC P0
0464 00236              RD0
0465 00183              XCH 7
0466 00237              RD1
0467 00182              XCH 6
0468 00096              INC 0
0469 00033              SRC P0
0470 00238              RD2
0471 00179              XCH 3
0472 00239              RD3
0473 00178              XCH 2
0474 00081              JMS COMPR
     00033
0476 00028              JCN AN SRCH
     00249
0478 00034              FIM P1 170      /IR(2)=12,IR(3)=10,  (*)
     00170
0480 00080              JMS PRINT
     00178
0482 00032              FIM P0 32       /IR(0)=2, IR(1)=0
     00032
0484 00209              LDM 1   / SET INIT ADR FLAG
0485 00033              SRC P0
0486 00230              WR2
0487 00032              FIM P0 0        / IR(0-1)=0
     00000
0489 00036              FIM P2 16       /IR(4)=1, IR(5)=0
     00016
0491 00033              SRC P0
0492 00238              RD2
0493 00242              IAC
0494 00037              SRC P2
0495 00230              WR2
0496 00208              LDM 0
0497 00178              XCH 2
0498 00033              SRC P0
0499 00239              RD3
0500 00130              ADD 2
0501 00037              SRC P2
0502 00231              WR3
0503 00066      CONT2,  JUN CONT1
     00006
0505 00032      SRCH,   FIM P0 16       /IR(0)=1, IR(1)=0
     00016
0507 00033              SRC P0  / INC INIT ADDRESS BY ONE (1)
0508 00238              RD2
0509 00242              IAC
0510 00230              WR2
0511 00208              LDM 0
0512 00178              XCH 2
0513 00239              RD3
0514 00130              ADD 2
0515 00231              WR3
0516 00065              JUN REPT
     00073
0518 00032      CONT1,  FIM P0 32       /IR(0)=2, IR(1)=0
     00032
0520 00033              SRC P0  / CHECK FORMAT ERROR EXIST
0521 00236              RD0
0522 00246              RAR
0523 00026              JCN CZ NOFE
     00015
0525 00066              JUN ADCHK
     00090
0527 00032      NOFE,   FIM P0 16       /IR(0)=1, IR(1)=0
     00016
0529 00036              FIM P2 32       /IR(4)=2, IR(5)=0
     00032
0531 00033              SRC P0  / WRITE DATA TO 1701
0532 00237              RD1
0533 00037              SRC P2
0534 00226              WRR     / LS 4-BITS, DATA
0535 00033              SRC P0
0536 00236              RD0
0537 00100              INC 4
0538 00037              SRC P2
0539 00226              WRR     / MS 4-BITS, DATA
0540 00220              LDM 12
0541 00186              XCH 10
0542 00038      PRGRM,  FIM P3 64       /IR(6)=4, IR(7)=0
     00064
0544 00039              SRC P3
0545 00000              NOP
0546 00000              NOP
/
/
0547 00210              LDM 2           /PULSING FOR 517 MSEC.
0548 00225              WMP
0549 00036              FIM P2 0        /IR(4)=0, IR(5)=0
     00000
0551 00040              FIM P4 11       /IR(8)=0, IR(9)=11
     00011
0553 00116      DELY1,  ISZ 4 DELY1
     00041
0555 00117              ISZ 5 DELY1
     00041
0557 00120              ISZ 8 DELY1
     00041
0559 00080              JMS SBR1
     00164
0561 00080              JMS SBR2
     00171
0563 00121              ISZ 9 DELY1
     00041
0565 00208              LDM 0
0566 00225              WMP
0567 00036              FIM P2 128      /IR(4)=8, IR(5)=0
     00128
0569 00080              JMS SBR1        / READ DELAY (5.6 MS)
     00164
0571 00080      READ,   JMS SBR1        / READ DELAY
     00164
0573 00096              INC 0
0574 00096              INC 0
0575 00033              SRC P0
0576 00234              RDR
0577 00037              SRC P2
0578 00225              WMP
0579 00183              XCH 7   / LS 4-BITS,1701 OUTPUT DATA
0580 00212              LDM 4
0581 00132              ADD 4
0582 00180              XCH 4
0583 00096              INC 0
0584 00033              SRC P0
0585 00234              RDR
0586 00037              SRC P2
0587 00225              WMP
0588 00182              XCH 6   / MS 4S,1701 OUTPUT DATA
0589 00209              LDM 1
0590 00176              XCH 0
0591 00033              SRC P0
0592 00236              RD0
0593 00178              XCH 2   / MS 4-BITS, INPUT DATA
0594 00237              RD1
0595 00179              XCH 3   / LS 4-BITS, INPUT DATA
0596 00081              JMS COMPR
     00033
0598 00020              JCN AZ ADCHK
     00090
0600 00066              JUN RPRGM
     00162
0602 00032      ADCHK,  FIM P0 0
     00000
0604 00033              SRC P0
0605 00236              RD0
```

```
0606 00179              XCH 3
0607 00237              RD1
0608 00178              XCH 2
0609 00238              RD2
0610 00183              XCH 7
0611 00239              RD3
0612 00182              XCH 6
0613 00081              JMS COMPR
     00033
0615 00028              JCN AN INADR
     00124
0617 00080      STB,    JMS CRLF
     00226
0619 00034              FIM P1 70       /IR(2)=4, IR(3)=5, (F)
     00070
0621 00080              JMS PRINT
     00178
0623 00032              FIM P0 32
     00032
0625 00033              SRC P0
0626 00208              LDM 0
0627 00228              WR0
0628 00229              WR1
0629 00230              WR2
0630 00036              FIM P2 0
     00000
0632 00081              JMS CLRAM
     00064
0634 00064              JUN STA1
     00005
0636 00036      INADR,  FIM P2 0         /ADDRESS INCREMENT
     00000
0638 00081              JMS CLRAM
     00064
0640 00032              FIM P0 3        /IR(0)=0, IR(1)=3
     00003
0642 00033              SRC P0
0643 00209              LDM 1
0644 00224              WRM
0645 00081              JMS ADDTN
     00046
0647 00032              FIM P0 0
     00000
0649 00036              FIM P2 48       /IR(4)=3, IR(5)=0
     00048
0651 00218              LDM 10
0652 00182              XCH 6
0653 00033      STOR1,  SRC P0
0654 00233              RDM
0655 00037              SRC P2
0656 00224              WRM
0657 00097              INC 1
0658 00101              INC 5
0659 00118              ISZ 6 STOR1
     00141
0661 00032              FIM P0 0
     00000
0663 00033              SRC P0
0664 00236              RD0
0665 00242              IAC
0666 00228              WR0
0667 00208              LDM 0
0668 00178              XCH 2
0669 00237              RD1
0670 00130              ADD 2
0671 00229              WR1
0672 00064              JUN PRGM1
     00083
0674 00122      RPRGM,  ISZ 10 PRGMA
     00168
0676 00080              JMS ERROR
     00135
0678 00065              JUN PRINL
     00162
0680 00034      PRGMA,  FIM P1 164       /IR(2)=10, IR(3)=4, (DS)
     00164
0682 00080              JMS PRINT
     00178
0684 00066              JUN PRGRM
     00030
/
/
```

```
0686 00034      FMER,   FIM P1 198       /IR(2)=12, IR(3)=6, (F)
     00198
0688 00080              JMS PRINT
     00178
0690 00034              FIM P1 197       /IR(2)=12, IR(3)=5, (E)
     00197
0692 00080              JMS PRINT
     00178
0694 00034              FIM P1 160       /IR(2)=10, IR(3)=0, (SP)
     00160
0696 00080              JMS PRINT
     00178
0698 00192              BBL 0
/
/
0699 00081      LISTR,  JMS PRINA
     00171
0701 00034              FIM P1 160       /IR(2)=10, IR(3)=0, (SP)
     00160
0703 00080              JMS PRINT
     00178
0705 00034              FIM P1 194       /IR(2)=12, IR(3)=2, (B)
     00194
0707 00080              JMS PRINT
     00178
0709 00222              LDM 14
0710 00186              XCH 10           /SET BIT COUNTER
0711 00044              FIM P6 48        /IR(12)=3, IR(13)=0
     00048
0713 00220      SECON,  LDM 12
0714 00182              XCH 6
0715 00245              SRC P6
0716 00234              RDR
0717 00245      ROTAT,  RAL
0718 00190              XCH 14
0719 00026              JCN CZ PRNTN
     00215
0721 00034              FIM P1 208       /IR(2)=13, IR(3)=0, (P)
     00208
0723 00080              JMS PRINT
     00178
0725 00066              JUN NEXT2
     00219
0727 00034      PRNTN,  FIM P1 206       /IR(2)=12, IR(3)=14, (N)
     00206
0729 00080              JMS PRINT
     00178
0731 00190      NEXT2,  XCH 14
0732 00118              ISZ 6 ROTAT
     00205
0734 00172              LD 12
0735 00248              DAC
0736 00188              XCH 12
0737 00122              ISZ 10 SECON
     00201
0739 00034              FIM P1 198       /IR(2)=12, IR(3)=6, (F)
     00198
0741 00080              JMS PRINT
     00178
0743 00034              FIM P1 187       /IR(2)=11, IR(3)=11, (SC)
     00187
0745 00080              JMS PRINT
     00178
0747 00127              ISZ 15 AINC
     00245
0749 00220              LDM 12
0750 00191              XCH 15
0751 00080              JMS CRLF
     00226
0753 00080              JMS LF
     00230
0755 00066              JUN ADCHK
     00090
0757 00034      AINC,   FIM P1 160       / SPACE
     00160
0759 00080              JMS PRINT
     00178
0761 00034              FIM P1 160               / SPACE
     00160
0763 00080              JMS PRINT
     00178
0765 00066              JUN ADCHK
     00090
```

PROM and ROM Duplication and Verification Program (A0544)

Refer to the MCB4-20 description (Appendix E) for a description of this program's use.

```
0000 00000              NOP
0001 00000              NOP
0002 00017      START,  JCN TZ START     / WAIT FOR TEST
     00002
0004 00046              FIM P7 192       /SET PASS CTR = 4
     00192
0006 00032              FIM P0 0         / SET SRC - RAM 0
     00000
0008 00033              SRC P0           / SEND RAM 0
0009 00210              LDM 2            / LOAD STARTLAMP BIT
0010 00225              WMP              / START LAMP ON (001)
0011 00096              INC 0            / SET SRC - ROM 1
0012 00033              SRC P0           / SEND ROM (IP1)
0013 00234              RDR              / READ MODE SELECT (IP1)
0014 00191              XCH 15           /STORE MODE SELECT (R15)
0015 00044              FIM P6 0         /CLEAR ROM ADR REG
     00000
0017 00032      ROMAD,  FIM P0 0         / SET SRC - ROM 0 (ROM ADR L)
     00000
0019 00033              SRC P0           / SEND ROM 0 (OP 0)
```

```
0020 00173              LD 13            / LOAD ROM ADR L - AC
0021 00226              WRR              / WRITE ROM ADR L (OP 0)
0022 00096              INC 0            / SET SRC - ROM 1 (ROM ADR H)
0023 00033              SRC P0           / SEND ROM 1 (OP 1)
0024 00172              LD 12            / LOAD ROM ADR H - AC
0025 00226              WRR              / WRITE ROM ADR H (OP 1)
0026 00032              FIM P0 64        / SET SRC - RAM 1 (ROM CS)
     00064
/
0028 00033              SRC P0           / SEND RAM 1
0029 00216              LDM 8            / LOAD ROM CSBIT
/
0030 00225              WMP              / ENABLE ROM CS (013)
0031 00080              JMS DLYTO        /ROM READ DELAY (10 MS)
     00190
0033 00032              FIM P0 32        / SET SRC - ROM 2 (ROM DATA L)
     00032
0035 00033              SRC P0           / SEND ROM 2 (IP 2)
0036 00234              RDR              / READ ROM DATA L (IP 2)
0037 00244              CMA              / COMPLEMENT DATA
```

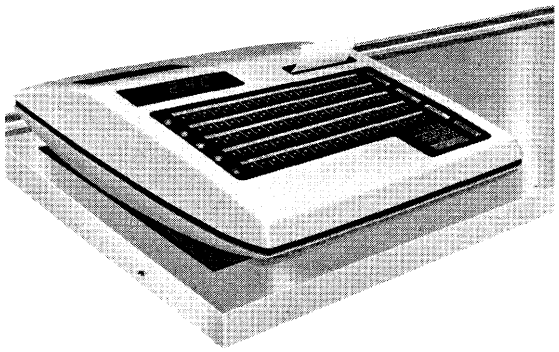
```
0038 00187          XCH 11          / STORE ROM DATA L (R11)
0039 00096          INC 0           / SET SRC - ROM 3 (ROM DATA H)
0040 00033          SRC P0          / SEND ROM 3 (IP 3)
0041 00234          RDR             / READ ROM DATA H (IP 3)
0042 00244          CMA             / COMPLEMENT DATA
0043 00186          XCH 10          / STORE ROM DATA H (R LØ)
0044 00032          FIM P0 64       / SET SRC - RAM 1 (ROM CS)
     00064
0046 00033          SRC P0          / SEND RAM 1
0047 00240          CLB             / CLEAR ROM CS BIT
0048 00225          WMP             / DISABLE ROM CS (Ø13)
0049 00240 MODE,    CLB             / CLEAR AC, CY
0050 00175          LD 15           /LOAD MODE SELECT - AC
0051 00246          RAR             / CHECK 1702A MODE
0052 00018          JCN C1 DUP      / JUMP IF CY=1 - DUP
     00078
0054 00246          RAR             / CHECK 1702 MODE
0055 00018          JCN C1 DUP      / JUMP IF CY=1 - DUP
     00078
0057 00246          RAR             /CHECK COMPARE MODE
0058 00026          JCN C0 STOP     / JUMP IF CY = Ø - STOP
     00071
0060 00032          FIM P0 32       / SET SRC - ROM 2 (ROM DATA L)
     00032
0062 00033          SRC P0          / SEND ROM 2 (IP 2)
0063 00171          LD 11
0064 00226          WRR             / WRITE ROM DATA L
0065 00096          INC 0           / SET SRC - ROM 3 (ROM DATA H)
0066 00033          SRC P0          / SEND ROM 3 (IP 3)
0067 00170          LD 10
0068 00226          WRR             / WRITE ROM DATA L
0069 00064          JUN COMP
     00122
0071 00032 STOP,    FIM P0 0        / SET SRC - RAM Ø
     00000
0073 00033          SRC P0          / SEND RAM Ø
0074 00240          CLB             / CLEAR AC, CY
0075 00225          WMP             / START LAMP OFF (ØØ1)
0076 00064          JUN START       / JUMP TO START
     00002
0078 00032 DUP,     FIM P0 32       / SET SRC - ROM 2 (PROM DATA IN L)
     00032
0080 00033          SRC P0          / SEND ROM 2 (OP 2)
0081 00171          LD 11           /  LOAD ROM DATA L - AC
0082 00226          WRR             /  WRITE PROM DATA IN L (OP 2)
0083 00096          INC 0           /SET SRC - ROM 3 (PROM DATA IN H)
0084 00033          SRC P0          / SEND ROM 3 (OP 3)
0085 00170          LD 10           / LOAD ROM DATA H - AC
0086 00226          WRR             / WRITE PROM DATA IN H (OP 3)
0087 00080          JMS DLYTO       / DATA SETTLE DELAY (10 MS)
     00190
0089 00240          CLB             /  CLEAR AC, CY
0090 00175          LD 15           / LOAD MODE SELECT - AC
0091 00246          RAR             / CHECK 1702A MODE
0092 00018          JCN C1 PGMA     / JUMP IF CY=1 - PGMA
     00099
0094 00246          RAR             / CHECK 1702 MODE
0095 00018          JCN C1 PGM02    / JUMP IF CY=1 - PROGRAM 1702
     00110
0097 00064          JUN STOP        / JUMP - STOP IFNOT PGM MODE
     00071
0099 00032 PGMA,    FIM P0 64       / SET SRC - RAM 1
     00064
0101 00033          SRC P0          / SEND RAM 1
0102 00210          LDM 2           / LOAD R/WA BIT
0103 00225          WMP             / ENABLE 1702A PGM (Ø11)
0104 00080          JMS DLYA        / PGM TIME (517 MS)
     00200
0106 00208          LDM Ø           / CLEAR R/WA BIT
0107 00225          WMP             / DISABLE 1702A PGM (Ø11)
0108 00064          JUN COMP        / JUMP - COMP
     00122
0110 00032 PGM02,   FIM P0 64       / SET SRC - RAM 1
     00064
0112 00033          SRC P0          / SEND RAM 1
0113 00212          LDM 4           / LOAD R/W 02 BIT
0114 00225          WMP             / ENABLE 1702 PGM (Ø12)
0115 00080          JMS DLY02       / PROGRAM TIME (5.6 SEC)
     00215
0117 00208          LDM Ø           / CLEAR R/W 02 BIT
0118 00032          FIM P0 64
     00064
0120 00033          SRC P0
0121 00225          WMP             / DISABLE 1702 PGM (Ø12)
0122 00080 COMP,    JMS DLYTO       / DATA SETTLE TIME (10MS)
     00190
0124 00032          FIM P0 32       / SET SRC - ROM 2 (PROM DATA OUT L)
     00032
0126 00034          FIM P1 128      / SET SRC - RAM 2 (DATA DSPLY)
     00128
0128 00033          SRC P0          / SEND ROM 2 (IP2)
0129 00234          RDR             / READ PROM DATA OUT L (IP2)
0130 00035          SRC P1          / SEND RAM 2 (PROM DATA OUT DSPLY L
0131 00225          WMP             / WRITE PROM DATA OUT DSPLY L (Ø2)
0132 00185          XCH 9           / STORE PROM DATA OUT L (R9)
0133 00096          INC Ø           / SET SRC - ROM 3 (PROM DATA OUT H)
0134 00212          LDM 4           / LOAD '64' BIT FOR RAM 3 SRC
0135 00130          ADD 2           /  SET SRC - RAM 3
0136 00178          XCH 2           / LOAD SRC H
0137 00033          SRC P0          / SEND ROM 3 (IP3)
0138 00234          RDR             / READ PROM DATA OUT H (IP3)
0139 00035          SRC P1          / SEND RAM 3 (PROM DATA OUT OSPLY H)
0140 00225          WMP             / WRITE PROM DATA OUT OSPLY H (Ø3)
0141 00184          XCH 8           / STORE PROM DATA OUT H (R8)
0142 00241          CLC             / CLEAR CY
0143 00171          LD 11           / LOAD ROM DATA L - AC
0144 00153          SUB 9           / SUB PROM DATA OUT L
0145 00028          JCN AN NOCMP    / JUMP IF DATA L NO COMP
     00155
0147 00241          CLC             / CLEAR CY
0148 00170          LD 10           / LOAD ROM DATA H - AC
0149 00152          SUB 8           / SUB PROM DATA OUT L
0150 00241          CLC             / CLEAR CY
0151 00028          JCN AN NOCMP    / JUMP IF DATA H NO COMP
     00155
0153 00064          JUN INADR       / JUMP - INADR IF DATA COMP
     00168
0155 00126 NOCMP,   ISZ 14 MODE     / INC PASS CTR, JUMP-MODE IF NOT
 4TH
     00049
0157 00034 ERROR,   FIM P1 Ø        / SET SRC - RAM Ø (ERROR LAMP)
     00000
0159 00035          SRC P1          / SEND RAM Ø
0160 00214          LDM 6           / LOAD ERROR LAMP ON BIT
0161 00225          WMP             / ERROR LAMP ON (ØØ2)
0162 00017 WAIT,    JCN TZ WAIT     / CONTINUE IF TEST PRESSED
     00162
0164 00210          LDM 2           / LOAD ERROR LAMP OFF BIT
0165 00225          WMP             / ERROR LAMP OFF (ØØ2)
0166 00220          LDM 12          / RESET PASS CTR = 4
0167 00190          XCH 14          / LOAD PASS CTR
/
/
/
/
0168 00173 INADR,   LD 13           / LOAD ROM ADR L - AC
0169 00241          CLC             / CLEAR CARRY
0170 00242          IAC             / INC ROM ADR L
0171 00189          XCH 13          / STORE NEW ROM ADR L
0172 00018          JCN C1 IRAH     / JUMP - INC ROM ADR H IF OVFL
     00176
0174 00064          JUN ROMAO       / JUMP - ROM ADR DATA
     00017
0176 00172 IRAH,    LD 12           / LOAD ROM ADR H - AC
0177 00242          IAC             / INC ROM ADR H
0178 00188          XCH 12          / STORE NEW ROM ADR L
0179 00018          JCN C1 FIN
     00183
0181 00064          JUN ROMAD
     00017
0183 00034 FIN,     FIM P1 Ø
     00000
0185 00035          SRC P1
0186 00216          LDM 8           / LOAD FINISH LAMP BIT
0187 00225          WMP             / FINISH LAMP ON
0188 00064          JUN START
     00002
/
/
/SUBROUTINES
/
/
0190 00038 DLYTO,   FIM P3 Ø        / DELAY TIME OUT = 9.08MS
     00000
0192 00000 TLT,     NOP
0193 00119          ISZ 7 TLT
     00192
0195 00000          NOP
0196 00000          NOP
0197 00118          ISZ 6 TLT
     00192
0199 00192          BBL Ø
0200 00034 DLYA,    FIM P1 Ø        / DELAY A = 517MS
     00000
0202 00036          FIM P2 11
     00011
0204 00114 TLA,     ISZ 2 TLA
     00204
0206 00115          ISZ 3 TLA
     00204
0208 00116          ISZ 4 TLA
     00204
0210 00080          JMS DLYTO
     00190
0212 00117          ISZ 5 TLA
     00204
0214 00192          BBL Ø
0215 00032 DLY02,   FIM P0 4        / DELAY Ø2 = 5.6SEC
     00004
0217 00113 TL2,     ISZ 1 A         / (LOOP SET = 11)
     00220
0219 00192          BBL Ø
0220 00080 A,       JMS DLYA
     00200
0222 00064          JUN TL2
     00217
/
/
/


SYMBOL TABLE:

 A      00220     FIN    00183     ROMAD 00017
 COMP   00122     INADR  00168     START 00002
 DLY02  00215     IRAH   00176     STOP  00071
 DLYA   00200     MODE   00049     TL2   00217
 DLYTO  00190     NOCMP  00155     TLA   00204
 DUP    00078     PGM02  00110     TLT   00192
 ERROR  00157     PGMA   00099     WAIT  00162
```

INTEL MICRO COMPUTERS

POINT-OF-SALE TERMINALS

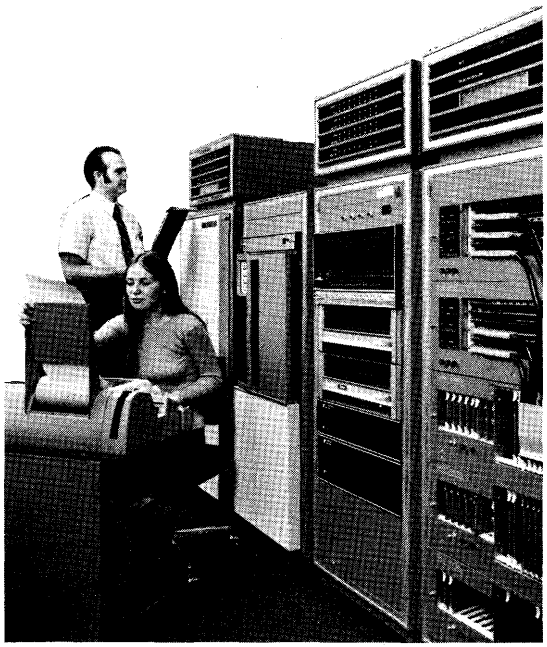


Staid, Inc. of Casselberry, Florida is using Intel micro computers to build advanced point-of-sale terminals for a large chain of cafeterias in the Southeast. Operated by the cashier, the terminal automatically enters item prices, totals items, adds taxes, prints a sales slip, dispenses change, adjusts the inventory of each item as it is sold, and transmits all this information to corporate headquarters. It can handle 100 separate items and is expandable to accommodate 200.

Staid says the Intel micro computer on only two PC cards does the work of about a dozen cards of random logic, and increases estimated reliability by an order of magnitude. Cost reduction, compared to random logic, is estimated to range from 20% to 30%.

Since the micro computers are programmed by Intel PROMs, Staid can produce point-of-sale terminals for the other types of businesses that have different requirements without redesign. They simply change the PROMs to make the terminal perform according to the new customer's requirements. Obviously, this saves a lot of money and enables them to deliver systems soon after receipt of order.

DATA COMMUNICATIONS PROCESSING



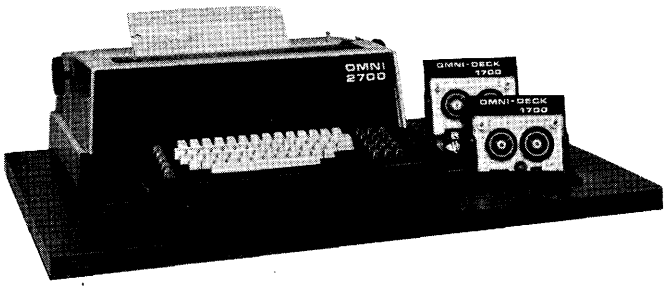
Action Communication Systems of Dallas used Intel micro computers as front-end processors in this high-speed dial-up communications controller built for The Bekins Company.

Action adopted Intel micro computers in order to save both development time and system cost. The Bekins system was fully developed and delivered *only 90 days* after Action decided to use Intel micro computers. And Action estimates they saved about $10,000 in over-all cost.

The Bekins controller, located in Glendale, California, is the heart of a nationwide multi-terminal system that carries administrative messages, financial data, shipping notices and customer inquiries. A micro computer on each of fives lines puts messages in a binary synchronous format, checks for errors, and signals for re-transmission when an error is detected.

Action used Intel's standard SIM4-02 micro computer boards in the system, and did the final programming with Intel's electrically-programmed PROMs. Intel's Micro Computer Systems Group worked very closely with Action in both the design and debugging phases of the project.

COMPACT BUSINESS MACHINES



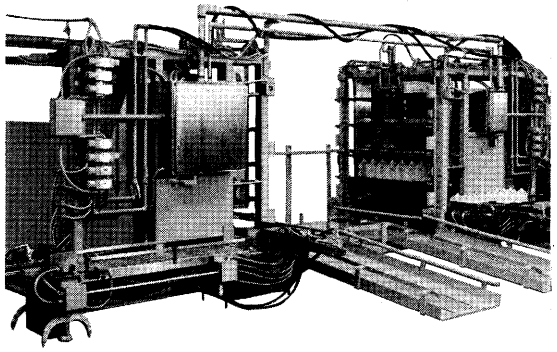
This general-purpose data processing machine for small businesses is built by Omni Electronics using an Intel micro computer as the heart of the system. Suitably programmed, this machine will tabulate accounts, type invoices, write checks, and even produce personalized form letters.

Omni says they saved about $3,000 by using an Intel micro computer in place of a mini. Moreover, the micro computer enabled them to reduce the whole system to typewriter size.

They say the micro computer has even more speed than they need, and offers the extraordinary reliability they require in this application.

In addition to the Intel integrated CPU, which does all central processing, the machine uses Intel's electrically-programmed PROMs for bootstrap programming and Intel's 2102 N-channel 1024-bit MOS RAMs as the central memory, a memory which stores up to 16K 8-bit bytes. Peripheral memory is supplied by one to eight Omni tape decks, which store 15,000,000 bits per cartridge.

PROCESS CONTROL



An Intel micro computer does all the thinking for this automatic bottle-loading machine. The micro computer, built by Comstar Corporation of Edina, Minnesota, for Conveyor Specialities, tells the machine how to load bottles of different sizes and when to perform each step in the loading process.

The little computer in a 6" x 6" x 1½" space replaces several racks of counters, timers and relays that would otherwise be required. According to Comstar, the computer's flexible programming is a major advantage. Programs on PROMs can be changed in half an hour.

Comstar estimates that the micro computer halved the cost of the control portion of this system, and reduced the time required to build it by a factor of two or three. The company is now building other types of systems with Intel micro computers, including an automatic meat weighing and packaging machine.

SALES OFFICES

NATIONAL SALES MANAGER

CALIFORNIA

Hank O'Hara
3065 Bowers Avenue
408/246-7501 TWX: 910-338-0026
Telex: 34-6372
Santa Clara 95051

U.S. REGIONAL SALES OFFICES

CALIFORNIA

William T. O'Brien
17401 Irvine Blvd., Suite K
714/838-1126, TWX: 910-595-1114
*Tustin 92680

MASSACHUSETTS

Myles Franklin
594 Marrett Road, Suite 27
617/861-1136, Telex: 92-3493
*Lexington 02173

MINNESOTA

Mick Carrier
800 Southgate Office Plaza
5001 West 78th Street
612/835-6722
*Bloomington 55437

U. S. SALES OFFICES

ARIZONA

Engineering Sales
7155 E. Thomas Road, No. 6
602/945-5781
Scottsdale 85252

CALIFORNIA

Jess Huffman
3065 Bowers Avenue
408/246-7501
*Santa Clara 95051

Jerry Plymire
3065 Bowers Avenue
408/246-7501
*Santa Clara 95051

John Alfoldy
17401 Irvine Blvd., Suit K
714/838-1126, TWX: 910-595-1114
*Tustin 92680

Jim Saxton
17401 Irvine Blvd., Suite K
714/838-1126, TWX: 910-595-1114
*Tustin 92680

Dave Neubauer
17401 Irvine Blvd., Suite K
714/838-1126, TWX: 910-595-1114
*Tustin 92680

Earle Associates, Inc.
4433 Convoy Street, Suite A
714/278-5441
San Diego 92111

COLORADO

Waugaman Associates, Inc.
4643 Wadsworth, Suite C
303/423-1020, TWX: 910-938-0750
Wheatridge 80033

*Direct Intel Sales Office

FLORIDA

Semtronic Associates
5100 DuPont Blvd., Suite 8E
DuPont Towers
305/782-1596
Ft. Lauderdale 33310

Semtronic Associates
100 Maitland Avenue, Suite 216
305/831-6851
Altamonte Springs 32701

ILLINOIS

Mar-Con Associates, Inc.
4836 Main Street
312/675-6450
Skokie 60076

MARYLAND

Barnhill and Associates
1931 Greenspring Drive
301/252-5610
Timonium 21093

Barnhill and Associates
P.O. Box 251
301/252-5610
Glen Arm 21057

MASSACHUSETTS

Bill D'Eramo
594 Marrett Road, Suite 27
617/861-1136, Telex: 92-3493
*Lexington 02173

Datcom
7A Cypress Drive
617/273-2990
Burlington 01803

MICHIGAN

Sheridan Assoc., Inc.
33708 Grand River Avenue
313/477-3800
Farmington 48024

MINNESOTA

Carl Branger
800 Southgate Office Plaza
5001 West 78th Street
612/835-6722
*Bloomington 55437

E.C.R., Inc.
4004 W. 78th Street
612/927-4547, TWX: 910-576-3153
Minneapolis 55435

MISSOURI

Sheridan Assoc., Inc.
110 S. Highway 140, Suite 10
314/837-5200
Florissant 63033

NEW JERSEY

Addem
Post Office Box 231
516/567-5900
Keasbey 08832

NEW YORK

Ossmann Components Sales Corp.
395 Cleveland Drive
716/832-4271
Buffalo 14215

Addem
37 Pioneer Blvd.
516/567-5900
Huntington Station, L.I. 11746

NEW YORK (Continued)

Ossmann Components Sales Corp.
280 Metro Park
716/442-3290
Rochester 14623

Ossmann Components Sales Corp.
1911 Vestal Parkway E.
607/785-9949
Vestal 13850

Ossmann Components Sales Corp.
132 Pickard Building
315/454-4477
Syracuse 13211

Ossmann Components Sales Corp.
411 Washington Avenue
914/338-5505
Kingston 12401

NORTH CAROLINA

Barnhill and Associates
6030 Bellow Street
703/846-4624
Raleigh 27602

OHIO

Sheridan Assoc., Inc.
10 Knollcrest Drive
513/761-5432, TWX: 810-461-2670
Cincinnati 15237

Sheridan Assoc., Inc.
7800 Wall Street
216/524-8120
Cleveland 44125

Sheridan Assoc., Inc.
Shiloh Bldg., Suite 250
5045 North Main Street
513/277-8911
Dayton 45405

PENNSYLVANIA

Vantage Sales Company
21 Bala Avenue
215/667-0990
Bala Cynwyd 19004

John Kitzrow
21 Bala Avenue
215/664-6636
Bala Cynwyd 19004

Sheridan Assoc., Inc.
4268 North Pike,
North Pike Pavilion
412/373-1070
Monroeville 15146

TENNESSEE

Barnhill and Associates
206 Chicashaw Drive
703/846-4624
Johnson City 37601

TEXAS

Evans and McDowell Assoc.
13333 N. Central Expressway
Room 180 214/238-7157
Dallas 75222

Evans and McDowell Assoc.
8814 Triola Lane 713/777-1282
Houston 77036

VIRGINIA

Barnhill and Associates
P.O. Box 1104
703/846-4624
Lynchburg 24505

WASHINGTON

SD.R^2 Products and Sales
14040 N.E. 8th Street
206/747-9424
Bellevue 98007

EUROPEAN MARKETING HEADQUARTERS

BELGIUM

Jens Paulsen
Intel Office
216 Avenue Louise
492003, Telex: 846-21060
*Bruxelles 1050

EUROPEAN MARKETING OFFICES

FRANCE

Intel Office
Cidex R-141
94-534 Rungis
(1) 677-60-75
*France

ENGLAND

Intel Corporation
Broadfield House
4 Between Towns Road
72992
Cowley, Oxford

EUROPEAN DISTRIBUTORS

AUSTRIA

Bacher Electronische Gerate GmbH
0222-93 01 43, Telex: (01) 1532
Vienna

BELGIUM

S.A. Inelco N.V.
02/60.00.12, Telex: 25441
Bruxelles

DENMARK

Scandinavian Semiconductor Supply
Aegir 5090, Telex: 19037
Copenhagen

FINLAND

Havulinna Oy
90-61451, Telex: 12426
Helsinki

FRANCE

Tekelec Airtronics
626-02-35, Telex: 25997
Paris

GERMANY

Ing. Erich Sommer
Elektronik GmbH
0611-55-02-89, Telex: 414069
Frankfurt

ISRAEL

Telsys Ltd. Engineering Co.
25 28 39, Telex: TSEE-IL 333192
Tel-Aviv

ITALY

Eledra 3 S,
(02) 86-03-07
Milano

NETHERLANDS

INELCO NV
020 44 16 66, Telex: 12534
Amsterdam

NORWAY

Nordisk Elektronik (Norge)
602590, Telex: 16963
Oslo

SOUTH AFRICA

Electronic Building Elements (PTY) Ltd.
P.O. Box 4609
Pretoria

SWEDEN

Nordisk Elektronik AB
08-24-83-40, Telex: 10547
Stockholm

SWITZERLAND

Industrade AG
01-60-22-30, Telex: 56788
Zurich

UNITED KINGDOM

Walmore Electronics Ltd.,
01-836-0201, Telex: 28752
London

ORIENT MARKETING HEADQUARTERS

JAPAN

Y. Magami
Intel, Japan
Han-Ei 2nd Bldg.
1-1, Shinjuku, Shinjuku-Ku
03-403-4747, Telex: 781-28426
*Tokyo 160

ORIENT DISTRIBUTORS

JAPAN

Pan Electron Inc.,
045-471-8321, Telex: 781-4773
Yokohama

DISTRIBUTORS

U. S. DISTRIBUTORS

WEST

ARIZONA

Hamilton/Avnet Electronics
1739 N. 28th Avenue
602/269-1391
Phoenix 85009

Cramer/Arizona
2816 N. 16th Street
602/263-1112
Phoenix 85006

CALIFORNIA

Hamilton/Avnet Electronics
340 E. Middlefield Road
415/961-7000
Mountain View 94041

Cramer/San Francisco
695 Veterans Blvd.
415/365-4000
Redwood City 94063

Hamilton Electro Sales
10912 W. Washington Blvd.
213/870-7171
Culver City 90230

Cramer/Los Angeles
17201 Daimler Street
714/979-3000
Irvine 92705

Hamilton/Avnet Electronics
8817 Complex Drive
714/279-2421
San Diego 92123

Cramer/San Diego
7719 Convoy Court
714/279-6300
San Diego 92111

COLORADO

Cramer/Denver
5465 E. Evans Place at Hudson
303/758-2100
Denver 80222

Hamilton/Avnet Electronics
5921 N. Broadway
303/534-1212
Denver 80216

NEW MEXICO

Cramer/New Mexico
137 Vermont, N.E.
505/265-5767
Albuquerque 87108

OREGON

Almac/Stroum Electronics
8888 S.W. Canyon Road
503/292-3534
Portland 97225

UTAH

Cramer/Utah
391 W. 2500 South
801/487-3681
Salt Lake City 84115

Hamilton/Avnet Electronics
647 W. Billinis Road
801/262-8451
Salt Lake City 84115

WASHINGTON

Almac/Stroum Electronics
5811 Sixth Avenue South
206/763-2300
Seattle 98108

Cramer/Seattle
5602 6th Avenue South
206/762-5755
Seattle 98108

Hamilton/Avnet Electronics
2320 Sixth Avenue
206/624-5930
Seattle 98121

MID-AMERICA

ILLINOIS

Cramer/Chicago
1911 South Busse Road
312/593-0230
Mt. Prospect 60056

Hamilton/Avnet Electronics
3901 North 25th Avenue
312/678-6310
Schiller Park 60176

INDIANA

Sheridan Assoc., Inc.
4165 Millersville Road, Suite DI-C7
317/542-0661
Indianapolis 46205

KANSAS

Hamilton/Avnet Electronics
3500 W. 75th Street, Suite 106
913/362-3250
Prairie Village 66208

MICHIGAN

Sheridan Assoc., Inc.
33708 Grand River Avenue
313/477-3800
Farmington 48024

Cramer/Detroit
13193 Wayne Road
313/425-7000
Livonia 48150

Hamilton/Avnet Electronics
13150 Wayne Road
313/522-4700
Livonia 48150

MINNESOTA

Cramer/Minneapolis
8053 Bloomington Freeway, Suite 105
612/881-8678
Bloomington 55420

Cramer/Bonn
7275 Bush Lake Road
612/941-4860
Edina 55435

Hamilton/Avnet Electronics
4940 Viking Drive
612/854-4800
Minneapolis 55431

Industrial Components, Inc.
4004 West 78th Street
612/927-9991
Minneapolis 55431

MISSOURI

Sheridan Assoc., Inc.
110 South Highway 140, Suite 10
314/837-5200
Florissant 63033

Hamilton Electro Sales
392 Brookes Lane
314/731-1144
Hazelwood 63042

OKLAHOMA

Cramer/Tulsa
6336 East 13th Street
918/836-3371
Tulsa 74112

OHIO

Cramer/Tri-States Inc.
666 Redna Terrace
513/771-6441
Cincinnati 45215

Sheridan Assoc., Inc.
10 Knollcrest Drive
513/761-5432
Cincinnati 45237

Cramer/Cleveland
5835 Harper Road
216/248-7740
Cleveland 44139

Sheridan Assoc., Inc.
7800 Wall Street
216/524-8120
Cleveland 44125

Sheridan Assoc., Inc.
Shiloh Bldg., Suite 250
5045 North Main St.
513/277-8911
Dayton 45405

TEXAS

Cramer Electronics
2970 Blystone
214/350-1355
Dallas 75220

Hamilton/Avnet Electronics
2403 Farrington
214/638-2850
Dallas 75207

Hamilton/Avnet Electronics
1216 West Clay
713/526-4661
Houston 77019

WISCONSIN

Cramer/Wisconsin
5626 N. 91st Street
414/462-8300
Milwaukee 53225

NORTHEAST

CONNECTICUT

Cramer/Connecticut
35 Dodge Avenue, North Haven
203/239-5641
New Haven 06473

MARYLAND

Cramer/EW Baltimore
922-24 Patapsco Avenue
301/354-0100
Baltimore 21230

Cramer/EW Washington
16021 Industrial Drive
301/948-0110
Gaithersburg 20760

Hamilton/Avnet Electronics
7255 Standard Drive
301/796-5000
Hanover 20176

MASSACHUSETTS

Cramer Electronics, Inc.
85 Wells Avenue
617/969-7700
Newton 02159

Hamilton/Avnet Electronics
185 Cambridge Street
617/273-2120
Burlington 01803

NEW JERSEY

Hamilton Electro Sales
220 Little Falls Road
201/239-0800
Cedar Grove 07009

Cramer/New Jersey
No. 1 Barett Avenue
201/935-5600
Moonachie 07074

Hamilton/Avnet Electronics
1608 Marlton Pike
609/662-9337
Cherry Hill 08034

Cramer/Pennsylvania, Inc.
7300 Route 130 North
609/662-5061
Pennsauken 08110

NEW YORK

Cramer/Binghamton
3220 Watson Boulevard
607/754-6661
Endwell 13760

Cramer/Rochester
3259 Winton Road South
716/275-0300
Rochester 14623

Cramer/Syracuse
6716 Joy Road
315/437-6671
East Syracuse 13057

Hamilton/Avnet Electronics
6400 Joy Road
315/437-2642
Syracuse 13211

Cramer/Long Island
29 Oser Avenue
516/231-5600
Hauppauge, L.I. 11787

Hamilton/Avnet Electronics
70 State Street
516/333-5800
Westbury, L.I. 11590

PENNSYLVANIA

Sheridan Assoc., Inc.
4268 North Pike
North Pike Pavilion
412/373-1070
Monroeville 15146

SOUTHEAST

ALABAMA

Cramer/EW Huntsville, Inc.
2310 Bob Wallace Avenue
205/539-5722
Huntsville 35805

FLORIDA

Cramer/EW Hollywood
4035 North 29th Avenue
305/923-8181
Hollywood 33020

Hamilton/Avnet Electronics
4020 North 29th Avenue
305/925-5401
Hollywood 33020

Cramer/EW Orlando
345 North Graham Ave.
305/841-1550
Orlando 32814

GEORGIA

Cramer/EW Atlanta
3130 Marjan Drive
404/448-9050
Atlanta 30340

Hamilton/Avnet Electronics
6700 Interstate 85 Access Road
404/448-0800
Atlanta 30071

NORTH CAROLINA

Cramer/EW Raleigh
3901 Winton Road
919 876-2371
Raleigh 27604

Cramer/EW Winston-Salem
938 Burke Street
919 725-8711
Winston-Salem 27102

PUERTO RICO

Cramer Electronics de
Puerto Rico
Subdivision Industrial,
Bo Retiro
San German 00753

CANADA

ONTARIO

Cramer/Canada
920 Alness Avenue, Unit No. 9
Downsview
416/661-9222
Toronto 392

Hamilton/Avnet Electronics
6291 Dormain Rd. No. 19
416/677-7432
Mississauga

Hamilton/Avnet Electronics
880 Lady Ellen Place
613/725-3071
Ottawa

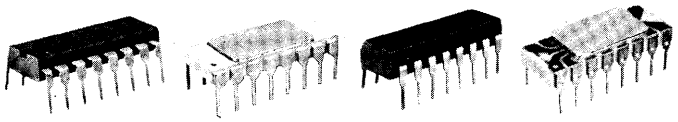
QUEBEC

Hamilton/Avnet Electronics
935 Monte De Liesse
514/735-6393
St. Laurent, Montreal 377

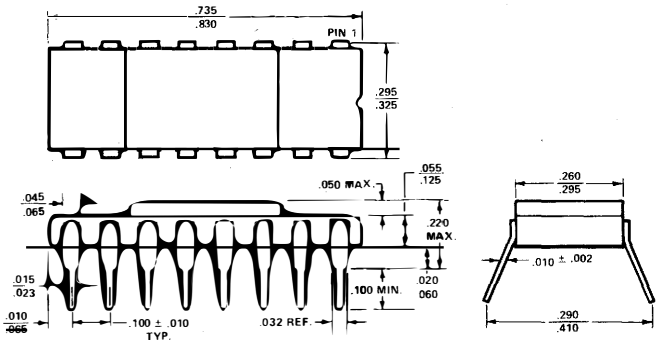
Ordering Information

1. The 4004 (CPU) is available in ceramic only and should be ordered as C4004.
2. The 4001 (ROM), 4002 (RAM) and 4003 (SR) are presently available off the shelf in plastic only. Standard devices should be ordered as follows:
   - P4001 Plastic Package
   - P4002-1 (Metal Option #1) - Plastic Package
   - P4002-2 (Metal Option #2) - Plastic Package
   - P4003 Plastic Package
3. The 4008 and 4009 standard memory and I/O interface set are available in plastic only (24 pin DIP). They should be used as a set and ordered as P4008 and P4009.
4. **Mask Programming of the 4001**
   The custom patterns, chip numbers and I/O options (including inverting and non-inverting inputs or outputs and on-chip resistor connected to either V_{DD} or V_{SS}) must be specified on a truth table for each 4001 ordered. Blank custom truth tables are available upon request from Intel.
5. **SIM4-01 Prototyping System**
   An interface board in which 1702A electrically programmable and erasable ROMs simulate the 4001 mask programmable ROMs provides a design tool for developing a system. This board contains one 4004, four 4002s and has provision for up to four 1702As. The board should be ordered as SIM4-01. The PROMs should be ordered separately.
6. **SIM4-02 Prototyping System**
   This board is an expanded version of the SIM4-01. It contains one 4004, four 4002s, and has provision for an additional twelve 4002s and sixteen 1702As. The board should be ordered as SIM4-02 (the type and number of PROMs and RAMs should also be indicated).
7. **MP7-03 PROM Programmer**
   This is the programmer board for 1602A/1702A. The three 1702A control ROMs used with the SIM4 cards for an automatic programming system are specified by pattern numbers A0540, A0541, A0543. All items should be ordered individually.
8. **SIM4 Hardware Assembler**
   Four PROMs plug into either prototyping board enabling the micro computer prototype to help program itself. To order this set of PROMs, specify pattern numbers A0740, A0741, A0742, and A0743.
9. **SIM4 Hardware Simulator**
   Nine PROMs plug into the SIM4-02 to aid in debugging of programs. To order this set of PROMs, specify pattern numbers A0750 through A0758.
10. **MCS-4 Assembler and Simulator Software Package**
    This software package converts a list of instruction mnemonics into machine instructions and then simulates the operation of the MCS-4 program. These programs are written in Fortran IV and are available via time-sharing service or directly from Intel.
11. **MCB4-10 System Interconnect and Control Module**
    This module provides control, display and I/O interconnect capability for the SIM4-01. In addition, it provides complete interconnection between the SIM4-01 and the MP7-03. To order the interconnect module only, specify MCB4-10.
12. **MCB4-20 System Interconnect and Control Module**
    This module provides control, display and I/O interconnect capability for the SIM4-02. In addition, it provides complete interconnection between the SIM4-02 and the MP7-03. To order the interconnect module only, specify MCB4-20.

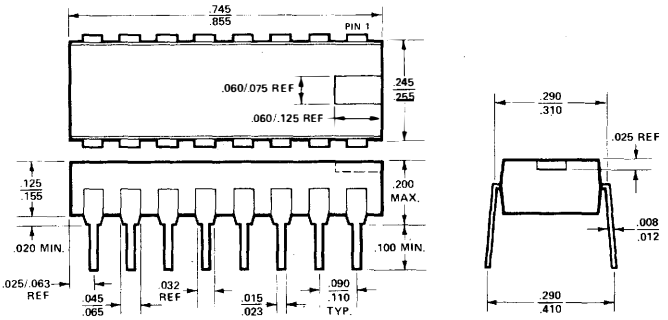
Packaging Information



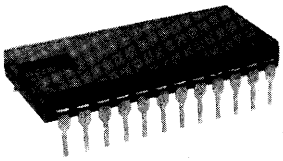
16-LEAD CERAMIC DUAL IN-LINE PACKAGE OUTLINE

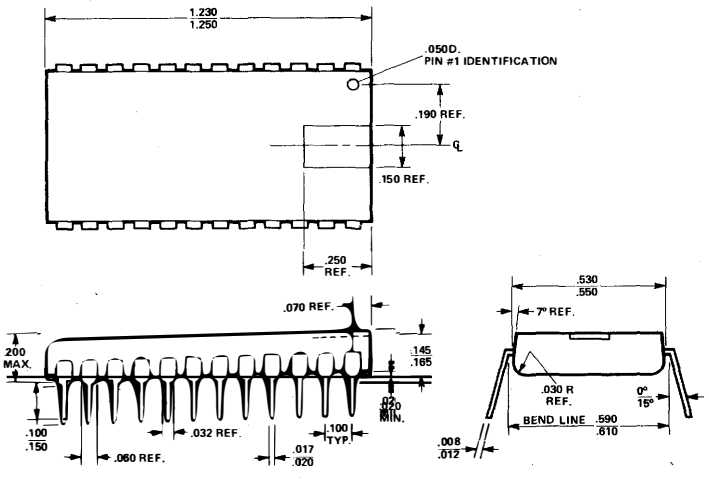


16-LEAD PLASTIC DUAL IN-LINE PACKAGE OUTLINE



24-LEAD PLASTIC DUAL IN-LINE PACKAGE OUTLINE





MCS-4 Instruction Set

[Those instructions preceded by an asterisk (*) are 2 word instructions that occupy 2 successive locations in ROM]

MACHINE INSTRUCTIONS

MNEMONIC	OPR $D_3 D_2 D_1 D_0$	OPA $D_3 D_2 D_1 D_0$	DESCRIPTION OF OPERATION
NOP	0 0 0 0	0 0 0 0	No operation.
*JCN	0 0 0 1 $A_2 A_2 A_2 A_2$	$C_1 C_2 C_3 C_4$ $A_1 A_1 A_1 A_1$	Jump to ROM address $A_2 A_2 A_2 A_2, A_1 A_1 A_1 A_1$ (within the same ROM that contains this JCN instruction) if condition $C_1 C_2 C_3 C_4$[1] is true, otherwise skip (go to the next instruction in sequence).
*FIM	0 0 1 0 $D_2 D_2 D_2 D_2$	R R R 0 $D_1 D_1 D_1 D_1$	Fetch immediate (direct) from ROM Data D_2, D_1 to index register pair location RRR.[2]
SRC	0 0 1 0	R R R 1	Send register control. Send the address (contents of index register pair RRR) to ROM and RAM at X_2 and X_3 time in the Instruction Cycle.
FIN	0 0 1 1	R R R 0	Fetch indirect from ROM. Send contents of index register pair location 0 out as an address. Data fetched is placed into register pair location RRR
JIN	0 0 1 1	R R R 1	Jump indirect. Send contents of register pair RRR out as an address at A_1 and A_2 time in the Instruction Cycle.
*JUN	0 1 0 0 $A_2 A_2 A_2 A_2$	$A_3 A_3 A_3 A_3$ $A_1 A_1 A_1 A_1$	Jump unconditional to ROM address A_3, A_2, A_1.
*JMS	0 1 0 1 $A_2 A_2 A_2 A_2$	$A_3 A_3 A_3 A_3$ $A_1 A_1 A_1 A_1$	Jump to subroutine ROM address A_3, A_2, A_1, save old address. (Up 1 level in stack.)
INC	0 1 1 0	R R R R	Increment contents of register RRRR.[3]
*ISZ	0 1 1 1 $A_2 A_2 A_2 A_2$	R R R R $A_1 A_1 A_1 A_1$	Increment contents of register RRRR. Go to ROM address A_2, A_1 (within the same ROM that contains this ISZ instruction) if result $\neq 0$, otherwise skip (go to the next instruction in sequence).
ADD	1 0 0 0	R R R R	Add contents of register RRRR to accumulator with carry.
SUB	1 0 0 1	R R R R	Subtract contents of register RRRR to accumulator with borrow.
LD	1 0 1 0	R R R R	Load contents of register RRRR to accumulator.
XCH	1 0 1 1	R R R R	Exchange contents of index register RRRR and accumulator.
BBL	1 1 0 0	D D D D	Branch back (down 1 level in stack) and load data DDDO to accumulator.
LDM	1 1 0 1	D D D D	Load data DDDD to accumulator.

INPUT/OUTPUT AND RAM INSTRUCTIONS

(The RAM's and ROM's operated on in the I/O and RAM instructions have been previously selected by the last SRC instruction executed.)

MNEMONIC	OPR $D_3 D_2 D_1 D_0$	OPA $D_3 D_2 D_1 D_0$	DESCRIPTION OF OPERATION
WRM	1 1 1 0	0 0 0 0	Write the contents of the accumulator into the previously selected RAM main memory character.
WMP	1 1 1 0	0 0 0 1	Write the contents of the accumulator into the previously selected RAM output port. (Output Lines)
WRR	1 1 1 0	0 0 1 0	Write the contents of the accumulator into the previously selected ROM output port. (I/O Lines)
WPM	1 1 1 0	0 0 1 1	Write the contents of the accumulator into the previously selected half byte of read/write program memory (for use with 4008/4009 only)
WRϕ[4]	1 1 1 0	0 1 0 0	Write the contents of the accumulator into the previously selected RAM status character 0.
WR1[4]	1 1 1 0	0 1 0 1	Write the contents of the accumulator into the previously selected RAM status character 1.
WR2[4]	1 1 1 0	0 1 1 0	Write the contents of the accumulator into the previously selected RAM status character 2.
WR3[4]	1 1 1 0	0 1 1 1	Write the contents of the accumulator into the previously selected RAM status character 3.
SBM	1 1 1 0	1 0 0 0	Subtract the previously selected RAM main memory character from accumulator with borrow.
RDM	1 1 1 0	1 0 0 1	Read the previously selected RAM main memory character into the accumulator.
RDR	1 1 1 0	1 0 1 0	Read the contents of the previously selected ROM input port into the accumulator. (I/O Lines)
ADM	1 1 1 0	1 0 1 1	Add the previously selected RAM main memory character to accumulator with carry.
RDϕ[4]	1 1 1 0	1 1 0 0	Read the previously selected RAM status character 0 into accumulator.
RD1[4]	1 1 1 0	1 1 0 1	Read the previously selected RAM status character 1 into accumulator.
RD2[4]	1 1 1 0	1 1 1 0	Read the previously selected RAM status character 2 into accumulator.
RD3[4]	1 1 1 0	1 1 1 1	Read the previously selected RAM status character 3 into accumulator.

ACCUMULATOR GROUP INSTRUCTIONS

CLB	1 1 1 1	0 0 0 0	Clear both. (Accumulator and carry)
CLC	1 1 1 1	0 0 0 1	Clear carry.
IAC	1 1 1 1	0 0 1 0	Increment accumulator.
CMC	1 1 1 1	0 0 1 1	Complement carry.
CMA	1 1 1 1	0 1 0 0	Complement accumulator.
RAL	1 1 1 1	0 1 0 1	Rotate left. (Accumulator and carry)
RAR	1 1 1 1	0 1 1 0	Rotate right. (Accumulator and carry)
TCC	1 1 1 1	0 1 1 1	Transmit carry to accumulator and clear carry.
DAC	1 1 1 1	1 0 0 0	Decrement accumulator.
TCS	1 1 1 1	1 0 0 1	Transfer carry subtract and clear carry.
STC	1 1 1 1	1 0 1 0	Set carry.
DAA	1 1 1 1	1 0 1 1	Decimal adjust accumulator.
KBP	1 1 1 1	1 1 0 0	Keyboard process. Converts the contents of the accumulator from a one out of four code to a binary code.
DCL	1 1 1 1	1 1 0 1	Designate command line.

NOTES: [1]The condition code is assigned as follows:

$C_1 = 1$ Invert jump condition $C_2 = 1$ Jump if accumulator is zero $C_4 = 1$ Jump if test signal is a 0

$C_1 = 0$ Not invert jump condition $C_3 = 1$ Jump if carry/link is a 1

[2]RRR is the address of 1 of 8 index register pairs in the CPU.

[3]RRRR is the address of 1 of 16 index registers in the CPU.

[4]Each RAM chip has 4 registers, each with twenty 4-bit characters subdivided into 16 main memory characters and 4 status characters. Chip number, RAM register and main memory character are addressed by an SRC instruction. For the selected chip and register, however, status character locations are selected by the instruction code (OPA).

intel

MCS-4™ MICRO COMPUTER SET • USERS MANUAL

FEBRUARY 1973

intel® INTEL CORPORATION, 3065 Bowers Avenue, Santa Clara, California 95051 • (408) 246-7501