
intel
MCS-4

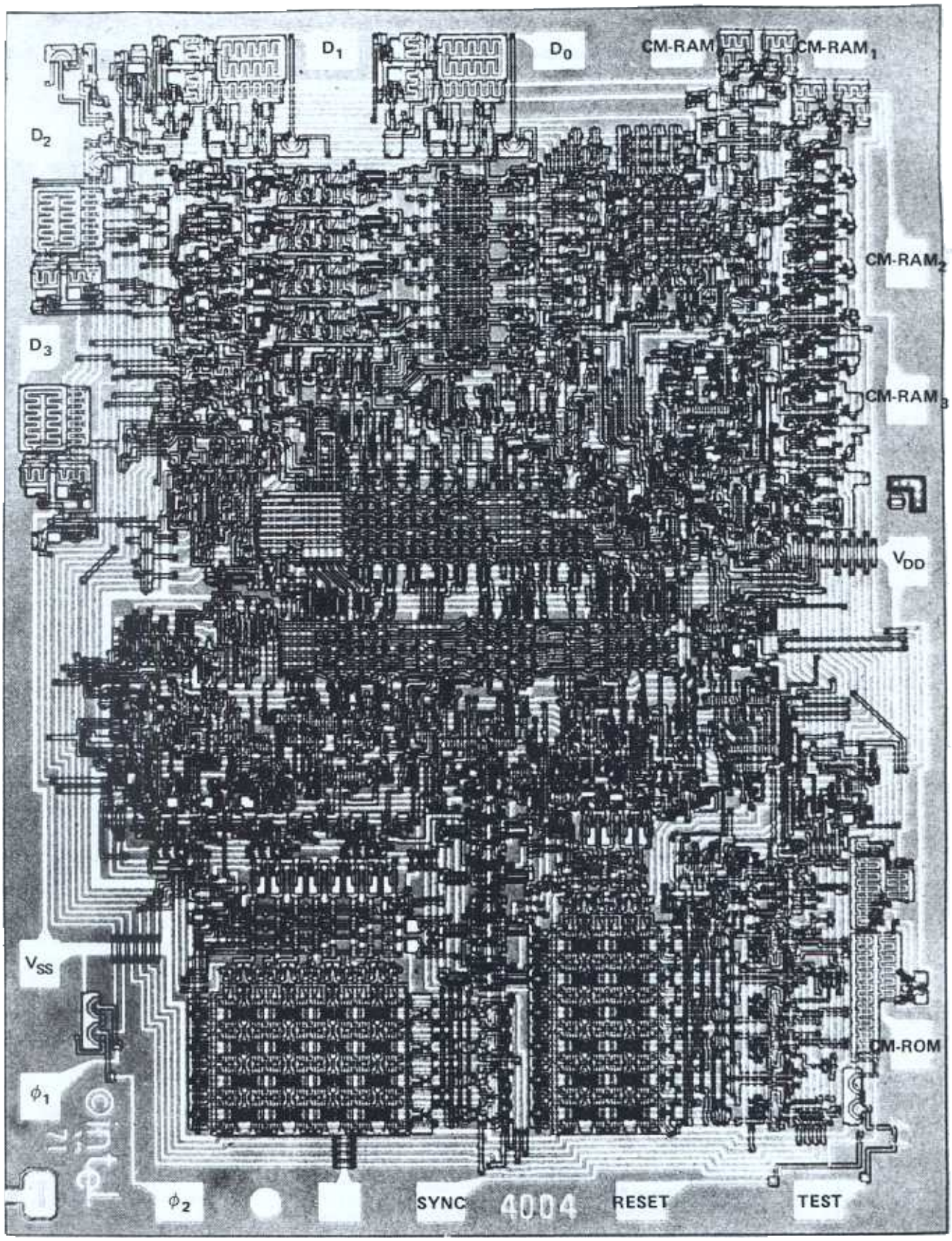
MCS™-4

FOUR-BIT PARALLEL MICROCOMPUTER SET

Features

- Microprogrammable General Purpose Computer Set
- 4-Bit Parallel CPU With 46 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
- 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
- I/O Ports and Control Lines are TTL Compatible
- Number of I/O Ports is Independent of the Size of the Program Memory
- New Instruction WPM (Write Program Memory) is Used for Loading Alterable Program Storage (RAM)

CONTENTS



4004 Photomicrograph With Pin Designations

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.

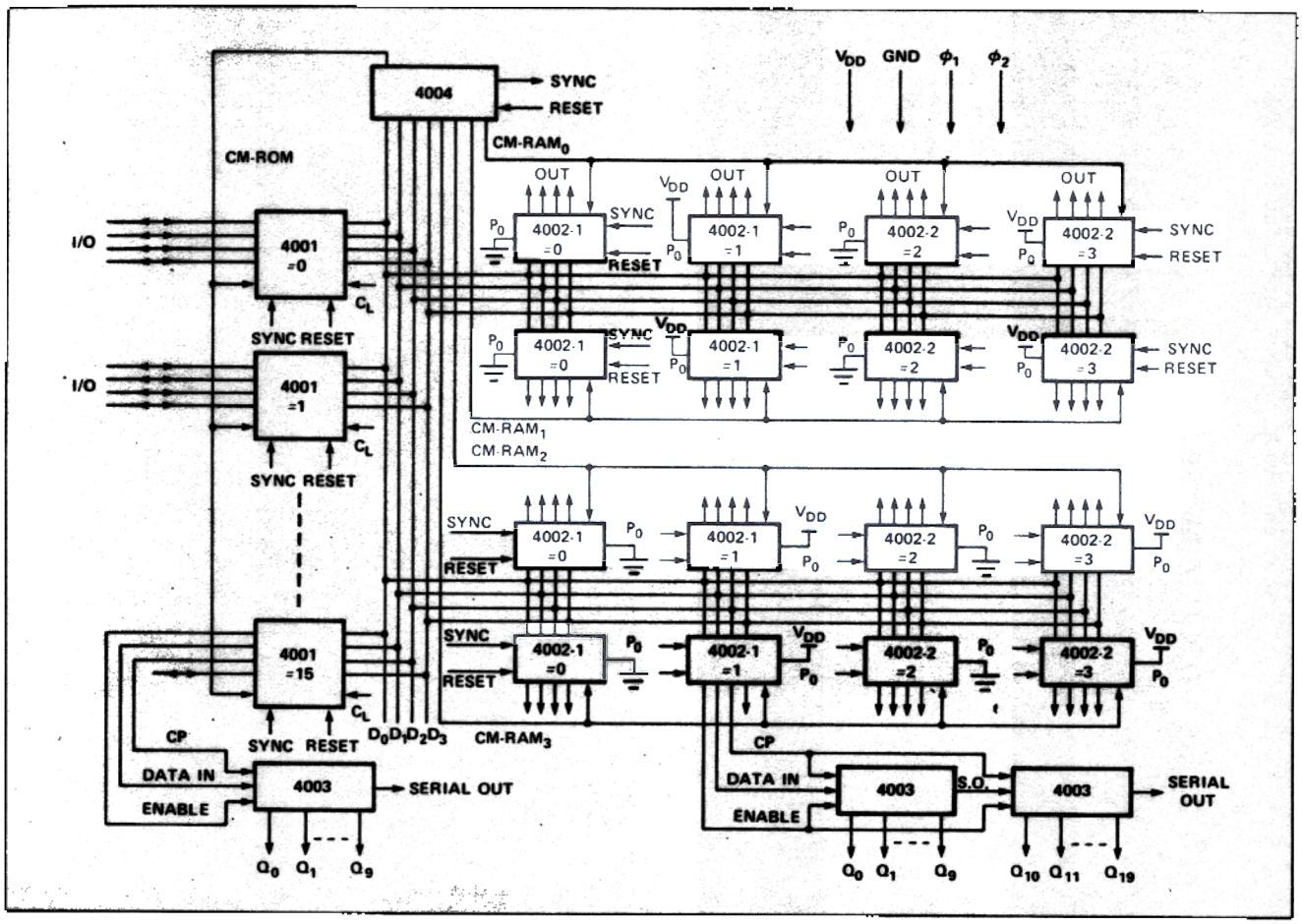


Figure 1. MCS-4 System Interconnection

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 - ϕ_1, ϕ_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 ϕ_1, ϕ_2.

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

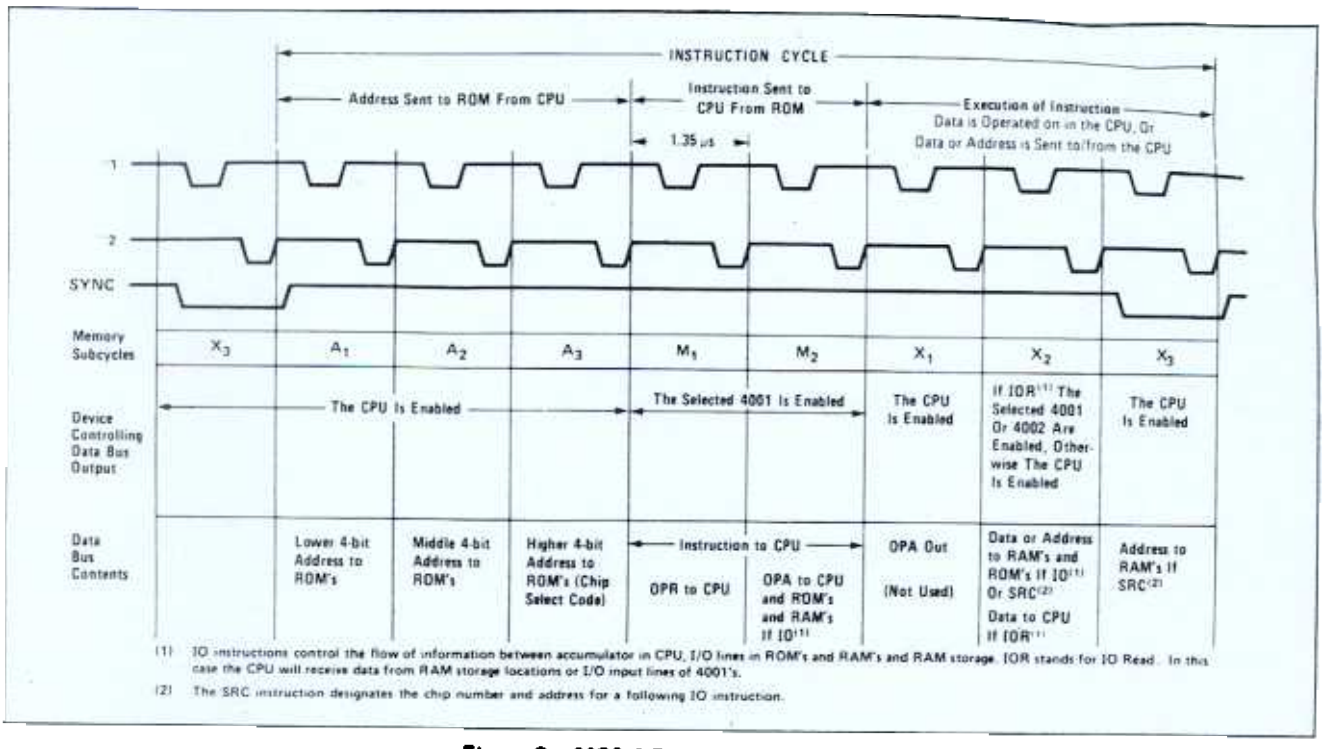


Figure 2. MCS-4 Basic Instruction Cycle

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.

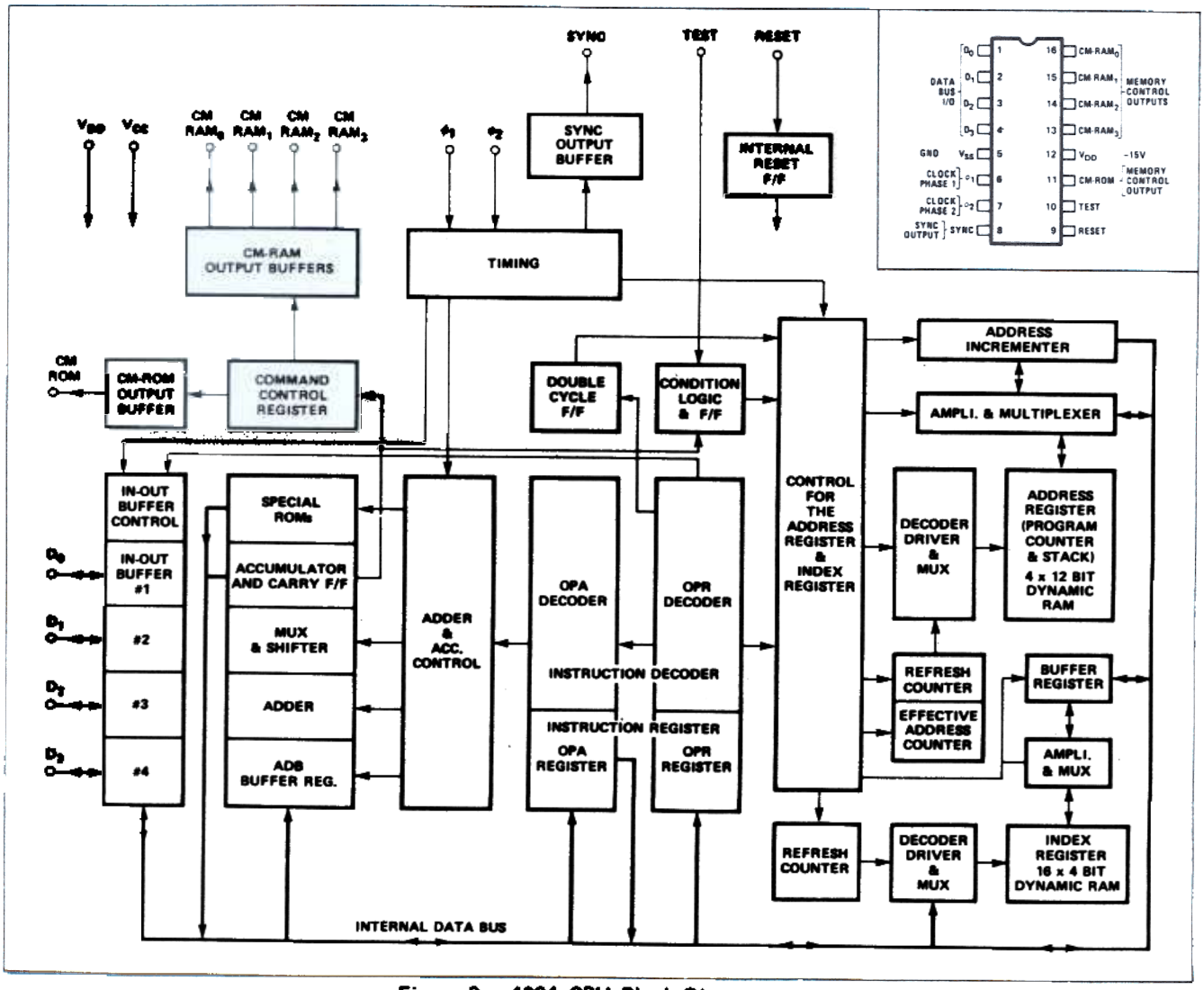


Figure 3. 4004 CPU Block Diagram

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 $\overline{\text{data}}$ to the adder. The other term of the addition comes from the accumulator and carry flip-flop. Both data and $\overline{\text{data}}$ 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

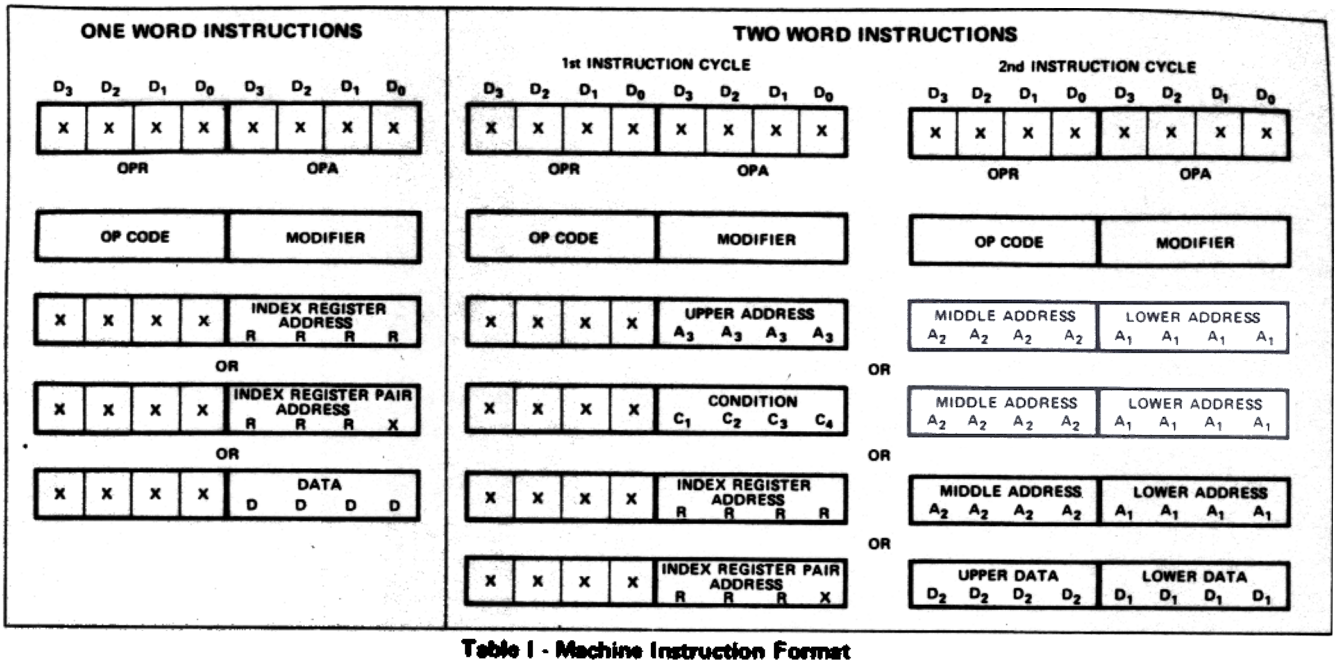


Table I - Machine Instruction Format

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.

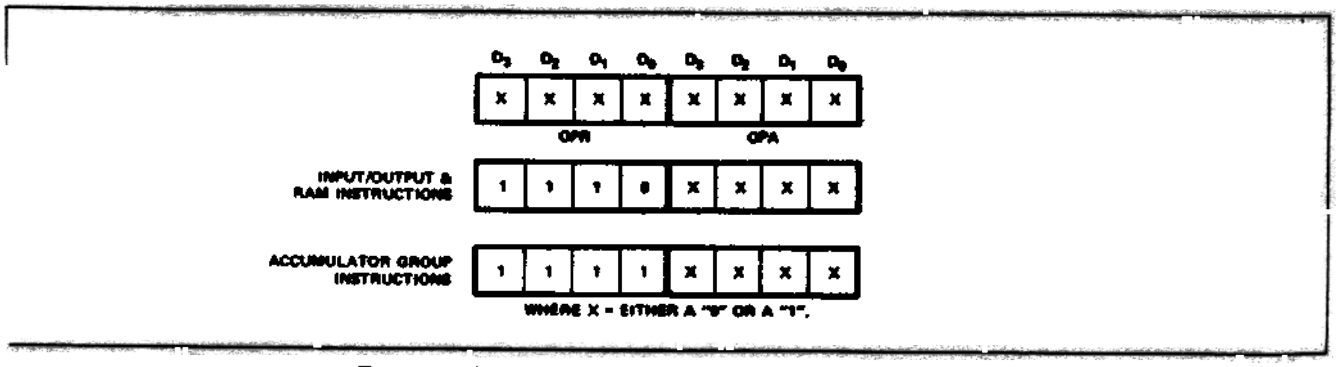


Table II - I/O and Accumulator Group Instruction Formats

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.

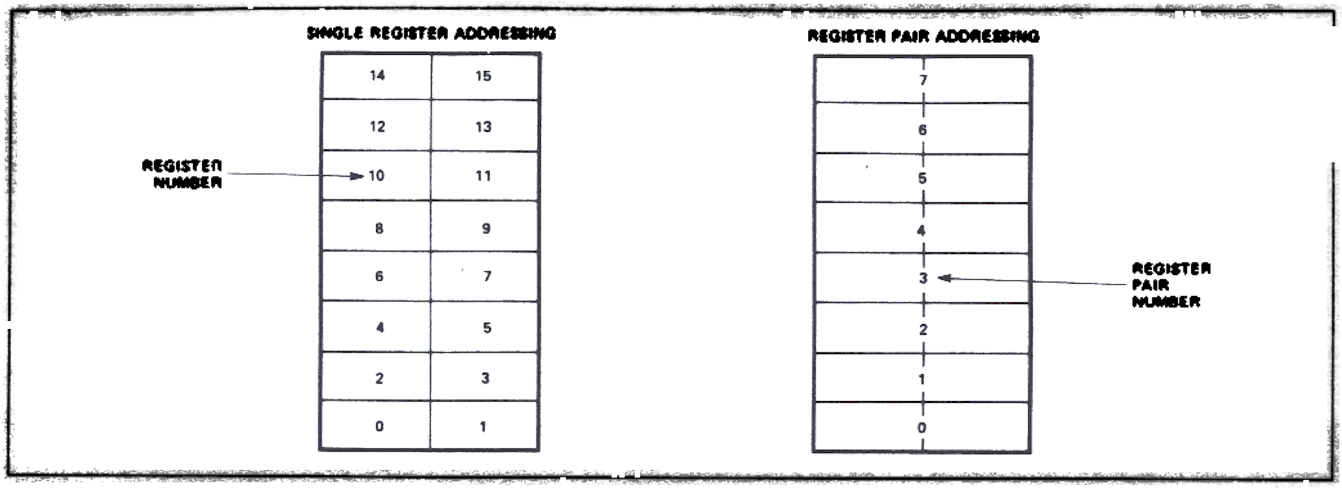


Table III - Index Register Organization

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

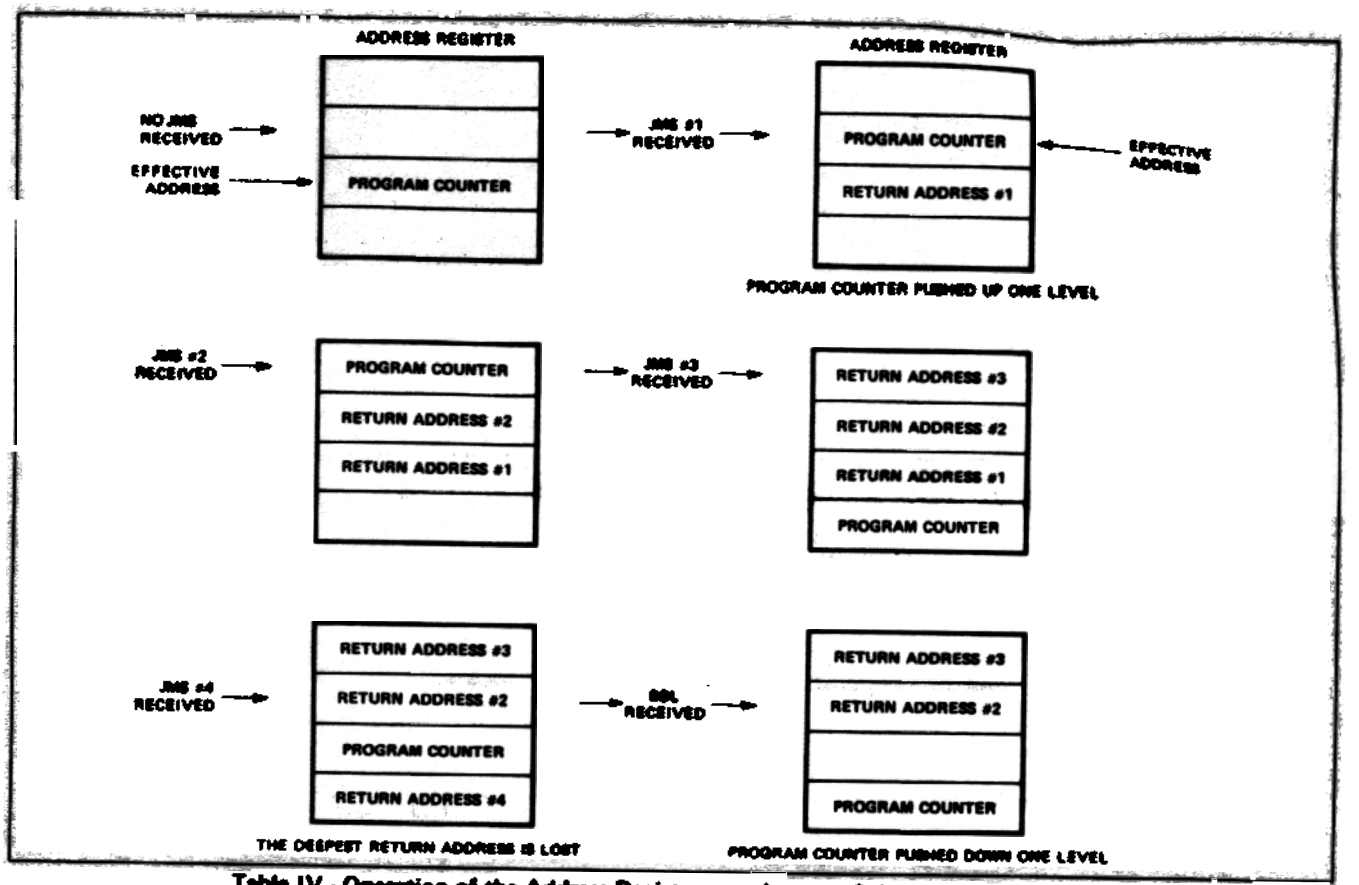


Table IV - Operation of the Address Register on a Jump to Subroutine Instruction

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, ...)

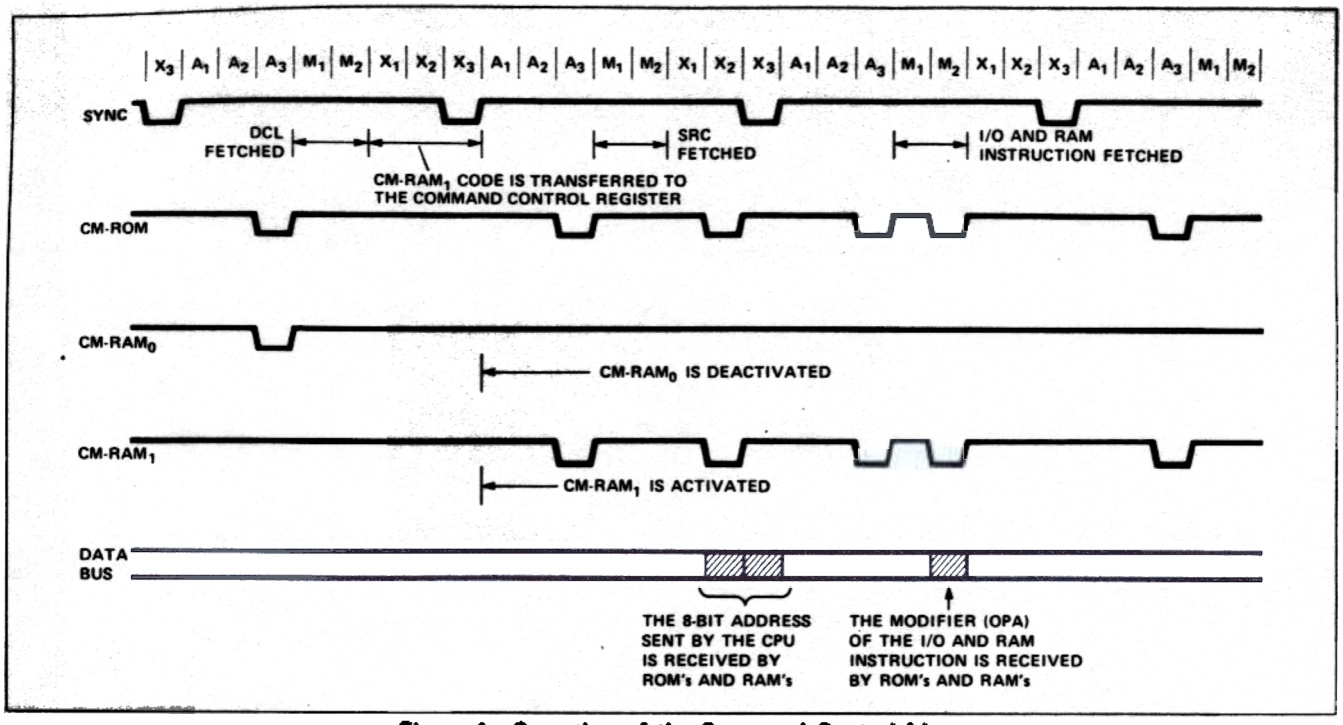


Figure 4. Operation of the Command Control Lines

Following is a detailed explanation of each step.

(1) Prior to execution of the DCL instruction the desired $CM\text{-}RAM_I$ code must be stored in the accumulator (for example through an LDM instruction).

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

The $CM\text{-}RAM_I$ 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 $CM\text{-}RAM_0$. This is because $CM\text{-}RAM_0$ 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 ≠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 $X_2.\phi_2$ 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 control 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.

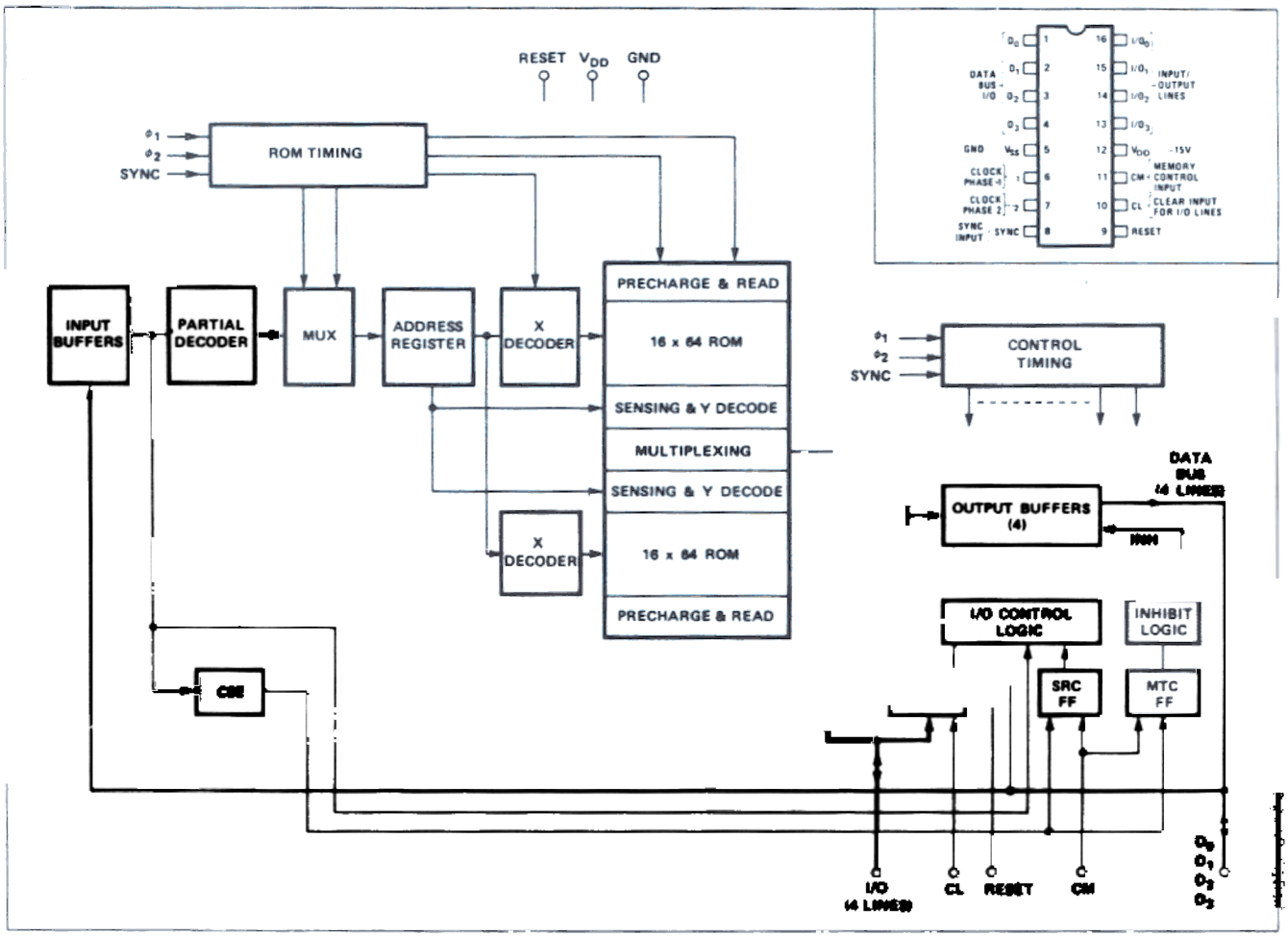


Figure 5. 4001 ROM Block Diagram

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.

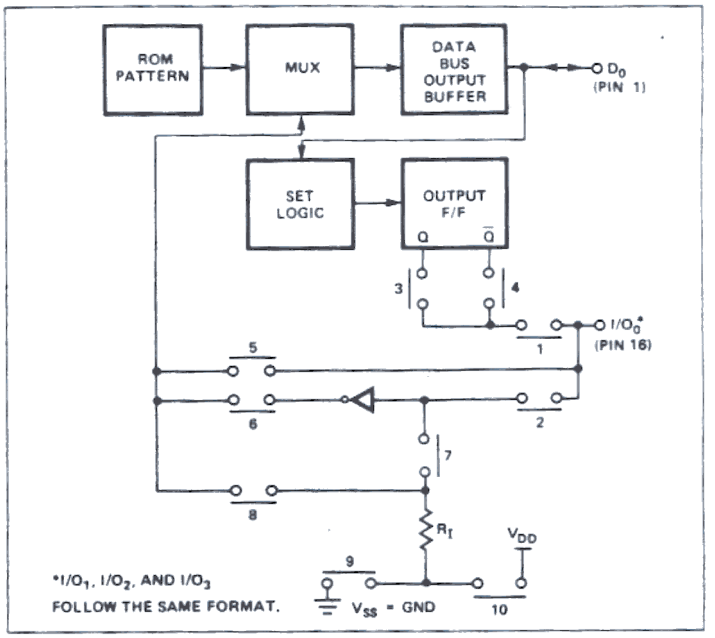


Figure 6. 4001 Available Metal Options for Each I/O Pin

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 previouslv (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
	4002-1	GND	
	4002-1	V_{DD}	
		GND	
		V_{DD}	

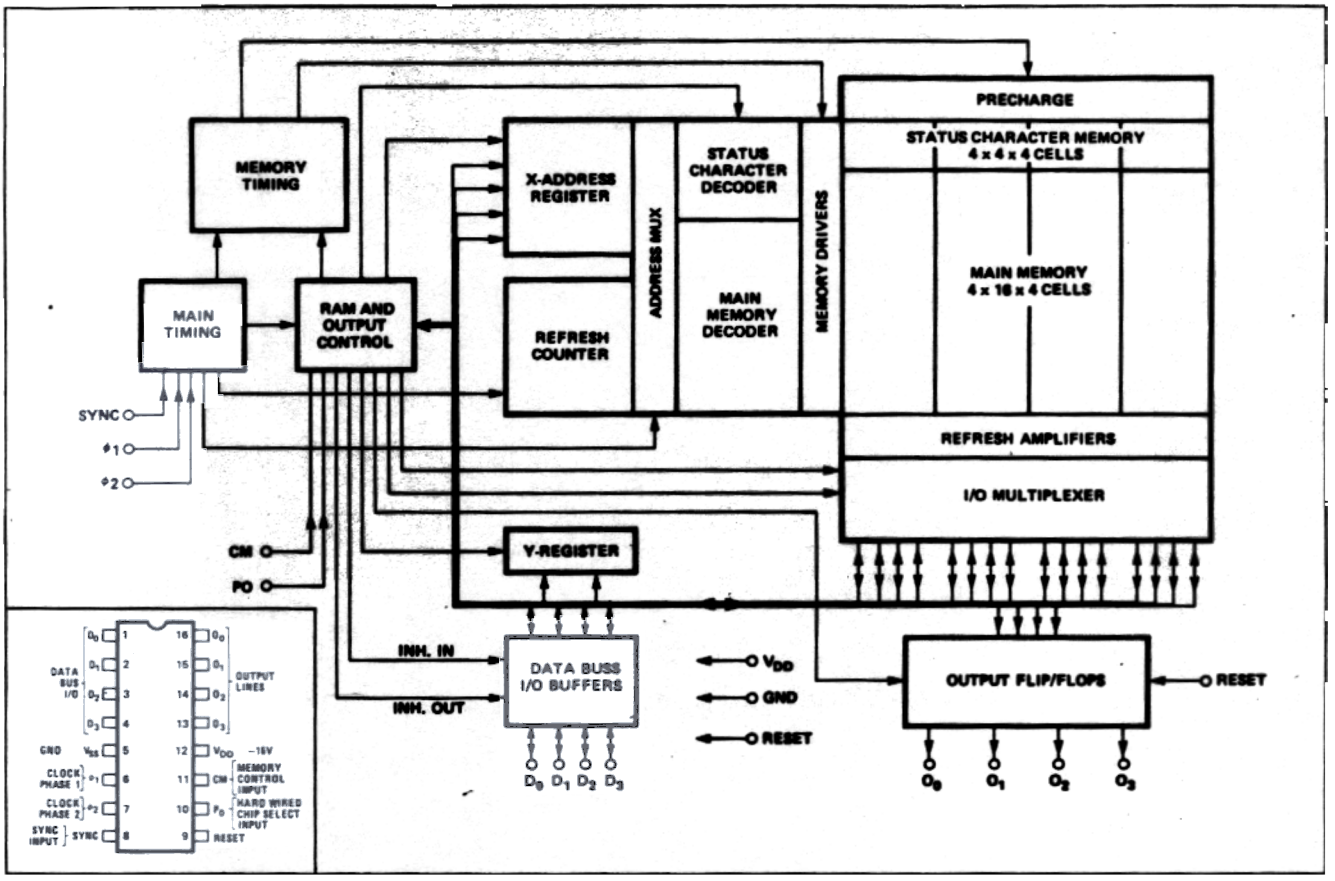


Figure 7. 4002 RAM Block Diagram

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 ($Q_1 = V_{SS}$) 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.

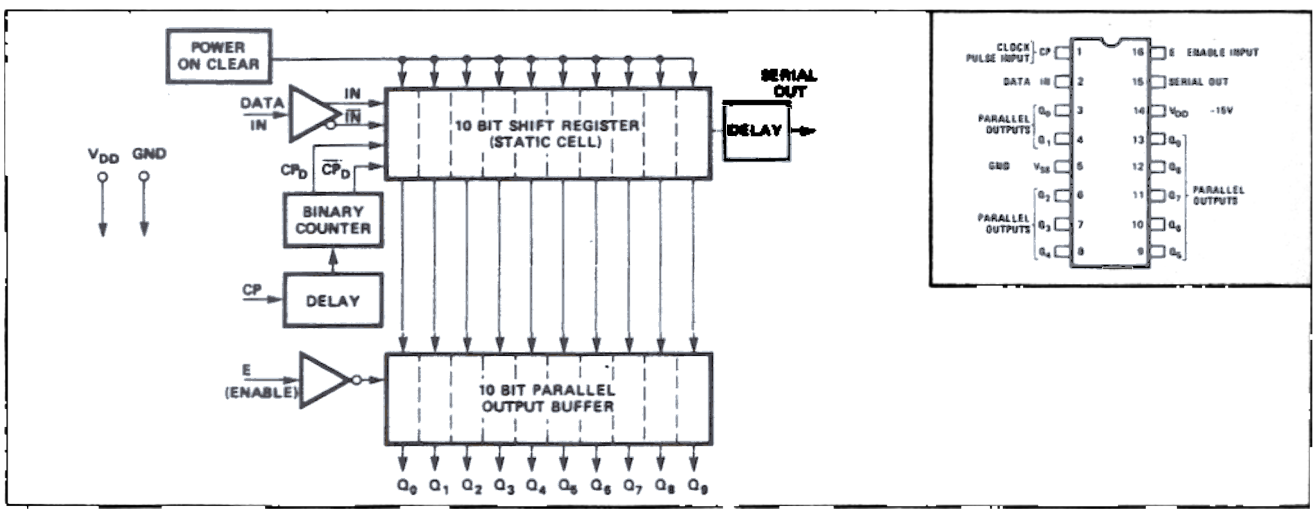


Figure 8. 4003 Shift Register Block Diagram

VII. 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.

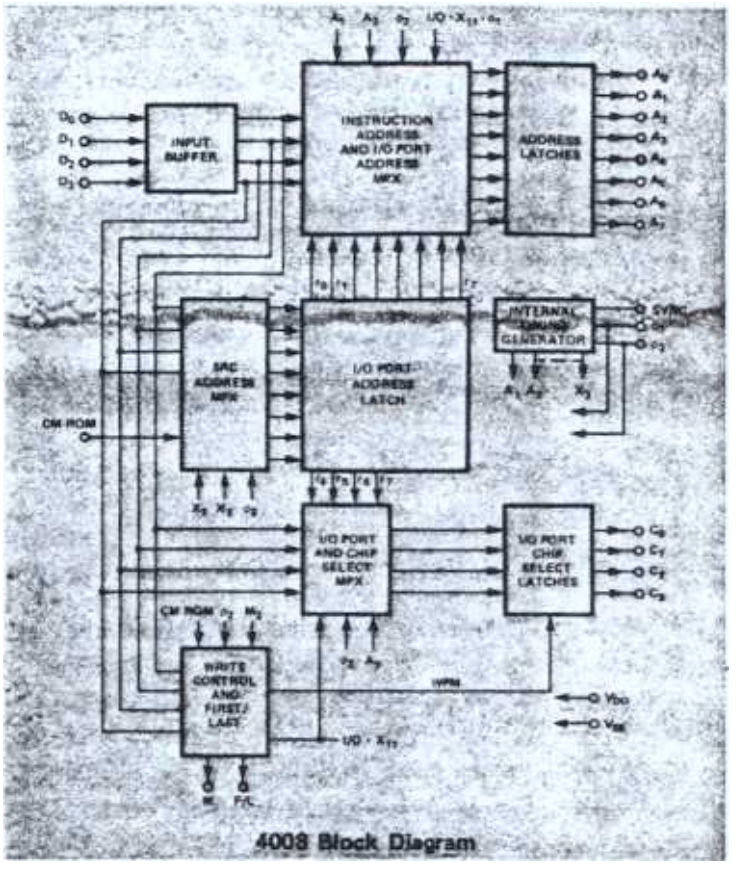
The accompanying diagrams show the internal organization of both the 4008 and 4009.

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.

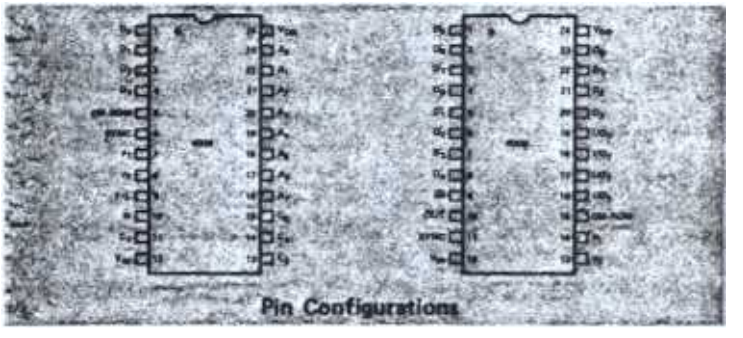
The 4009 then transfers the eight bit instruction from program memory to the 4004 four bits at a time at M1 and M2. The command signal sent by the CPU activates the 4009 and initiates this transfer.

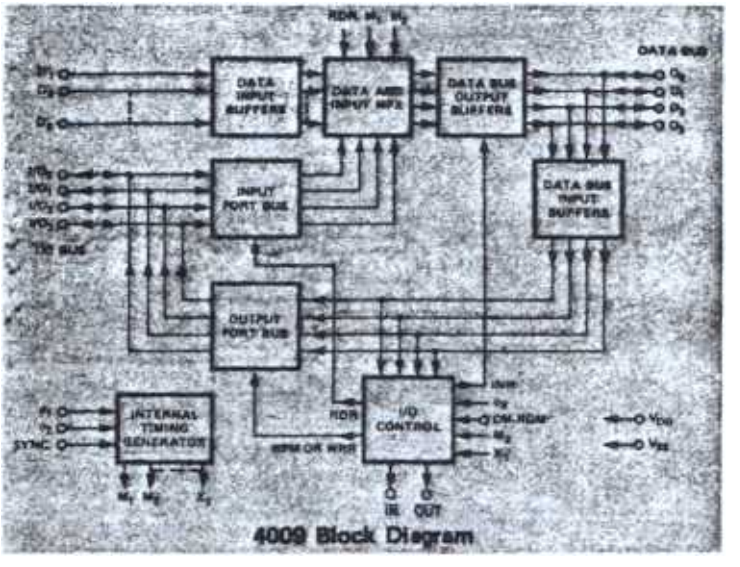
When the CPU executes an SRC (Send Register Control) instruction, the 4008 responds by storing the I/O address in its eight bit SRC register. The content of this SRC register is always transferred to the address lines (A0 through A7) and the chip select lines (C0 through C3) at X1 time. The appropriate 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. 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.

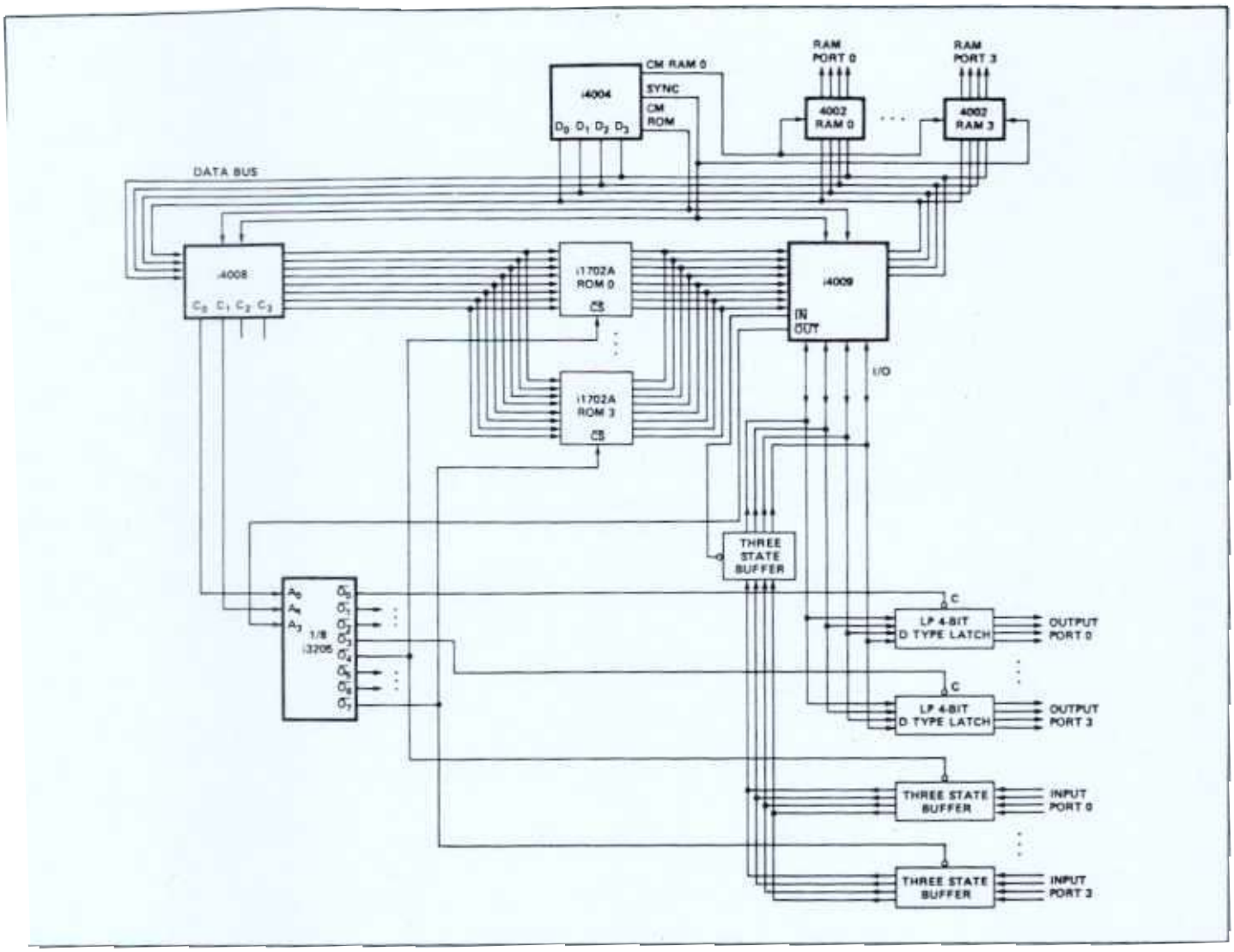
Example 4: Eight 1702A PROMs, eight RAM Blocks, and eight I/O Ports.

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.

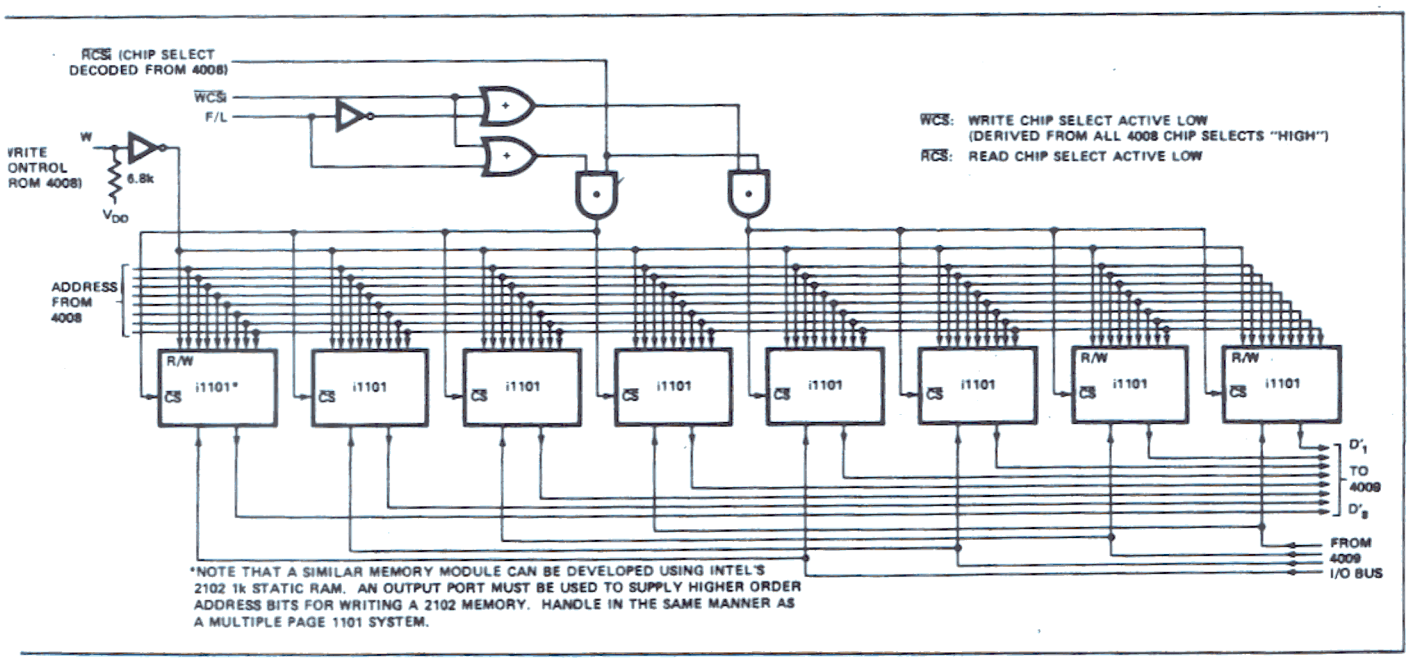
IMPORTANT:

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

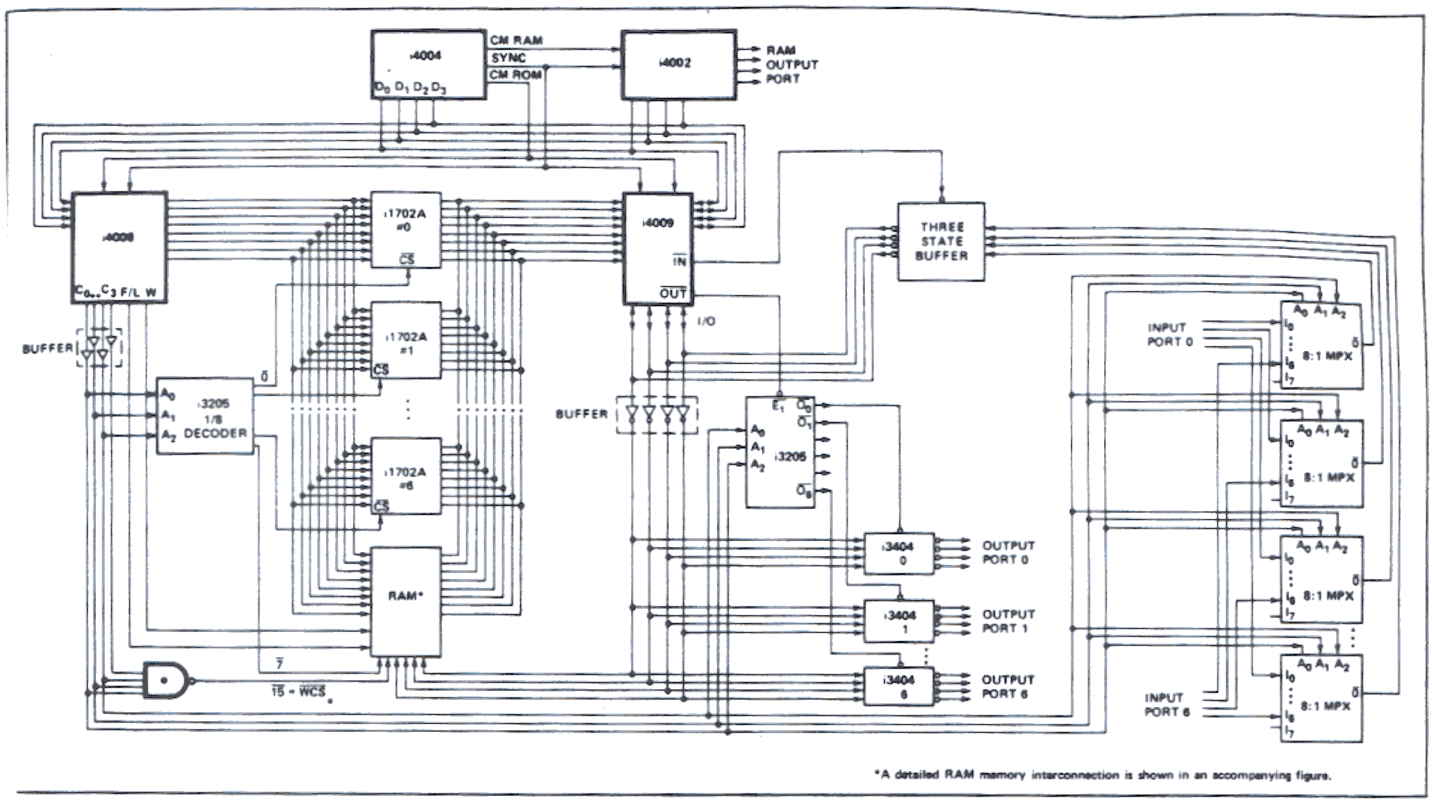
1. For normal operation, 4001 ROMs cannot be used in the same system with 4008/4009.
2. Memory address, memory data, I/O bus, and control lines from both 4008 and 4009 are defined with respect to positive logic. The MCS-4 data and control lines from the 4004 are defined with respect to negative logic. 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.



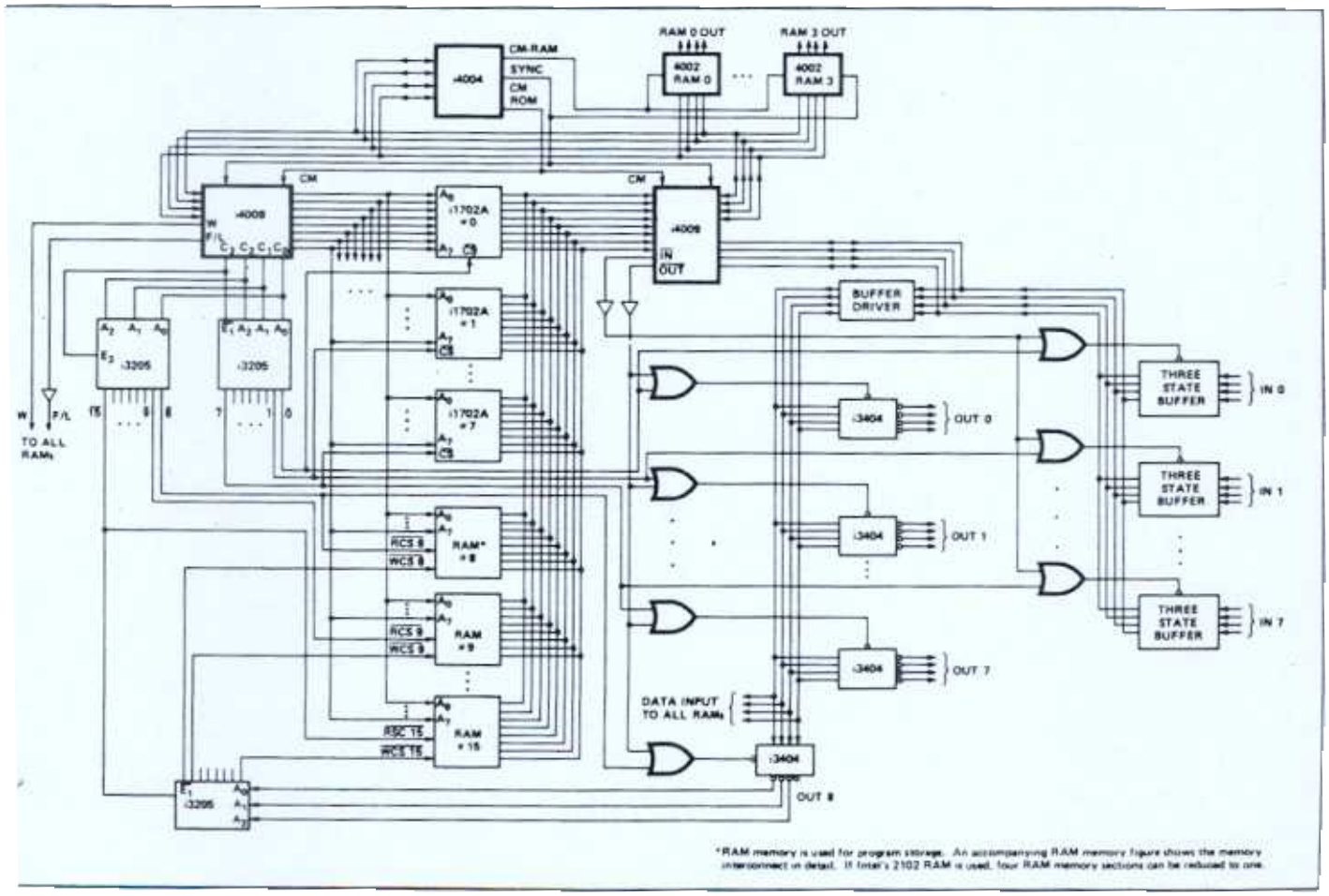
Example 1. Four 1702As and Four I/O Ports



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

VIII. 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 μsec.

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)
ai	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

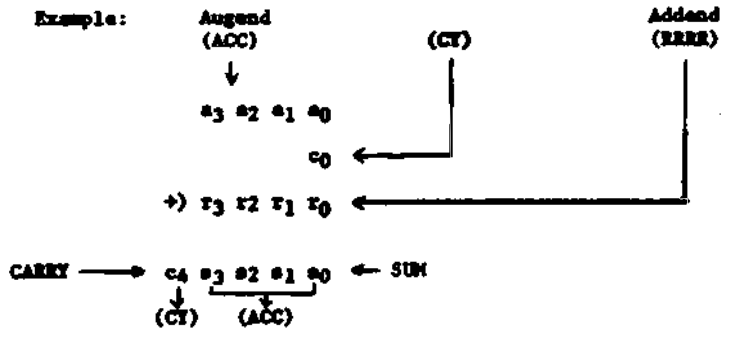
Mnemonic: LDM (Load Data to Accumulator)
OPR OPA: 1101 DDDD
Symbolic: DDDD → ACC
Description: The 4 bits of data, DDDD stored in the OPA field of instruction word are loaded into the accumulator. The previous contents of the accumulator are lost. The carry/link bit is unaffected.

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 accumulator)
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.

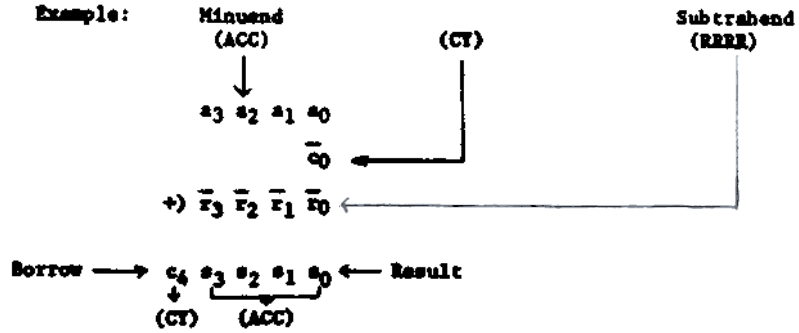
Mnemonic: ADD (Add index register to accumulator with carry)
OPR OPA: 1000 RRRR
Symbolic: (RRRR) + (ACC) + (CT) → ACC, CT
Description: The 4 bit content of the designated index register is added to the content of the accumulator with carry. The result is stored in the accumulator. The carry/link is set 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) + (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 RRRO
Symbolic: (P_H) (0000) (0001) ⟶ ROM address
(OPR) ⟶ RRRO
(OPA) ⟶ 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).

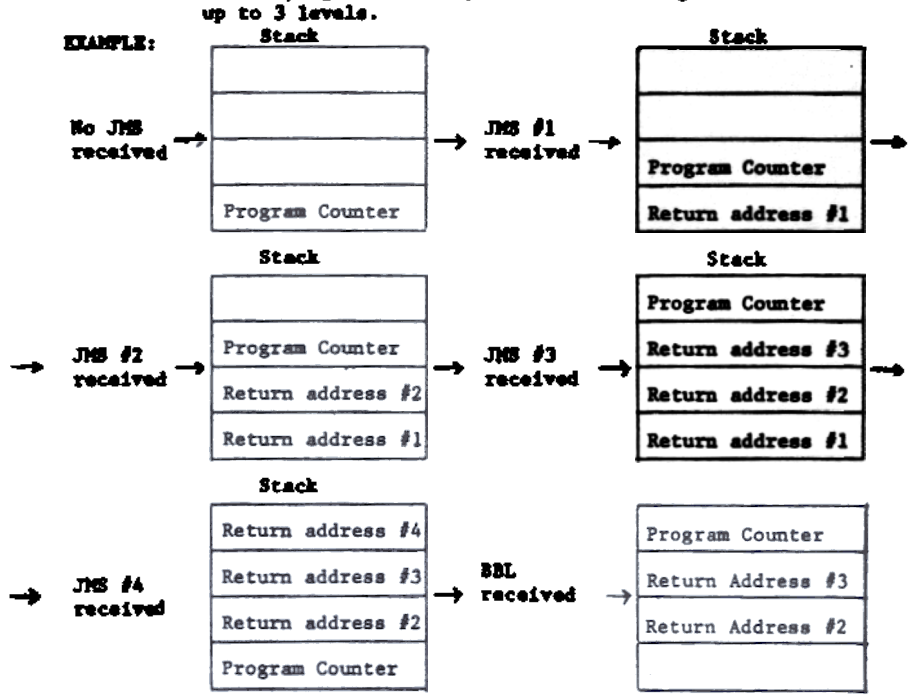
E. Two Word Machine Instruction

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 ⟶ P_L, A_2 A_2 A_2 A_2 ⟶ P_M, A_3 A_3 A_3 A_3 ⟶ 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)⟶Stack
A_1 A_1 A_1 A_1 ⟶ P_L, A_2 A_2 A_2 A_2 ⟶ P_M,
A_3 A_3 A_3 A_3 ⟶ 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 $(P_H) \rightarrow P_H$,
$A_2A_2A_2A_2 \rightarrow P_M$, $A_1A_1A_1A_1 \rightarrow P_L$
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 $A_2A_2A_2A_2$, $A_1A_1A_1A_1$ 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) → 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) → 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 ⇒ 1 (1 or 0) (1 or 0)

Input Data — User can choose

Mnemonic: WRM (Write accumulator into RAM character)
OPR OPA: 1110 0000
Symbolic: (ACC) → 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) $\rightarrow M_{S2}$

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

Mnemonic: WRR (Write ROM port)
OPR OPA: 1110 0010
Symbolic: (ACC) $\rightarrow$ 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) $\rightarrow$ 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 accumulator appears on O_0, Pin 16, of the 4002.)

Mnemonic: ADM (Add from memory with carry)
OPR OPA: 1110 1011
Symbolic: (M) + (ACC) + (CY) $\rightarrow$ 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) + $\overline{(CY)}$ $\rightarrow$ 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 $\rightarrow$ ACC, 0 $\rightarrow$ CY
Description: Set accumulator and carry/link to 0.

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

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

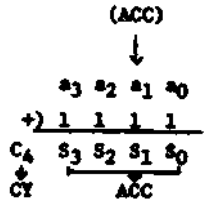
Mnemonic: STC (Set carry)
OPR OPA: 1111 1010
Symbolic: 1 $\rightarrow$ CY
Description: Set carry/link to a 1

Mnemonic: CMA (Complement Accumulator)
OPR OPA: 1111 0100
Symbolic: $\overline{a_3}\overline{a_2}\overline{a_1}\overline{a_0}$ $\rightarrow$ 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 → 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 → 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: $C_0 \rightarrow a_0$, $a_i \rightarrow a_{i+1}$, $a_3 \rightarrow$ CY
Description: The content of the accumulator and carry/link are rotated left.

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

Mnemonic: TCC (Transmit carry and clear)
OPR OPA: 1111 0111
Symbolic: 0 → ACC, (CY) → a_0, 0 → 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 → 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 → ACC if (CY) = 0
1010 → ACC if (CY) = 1
0 → 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

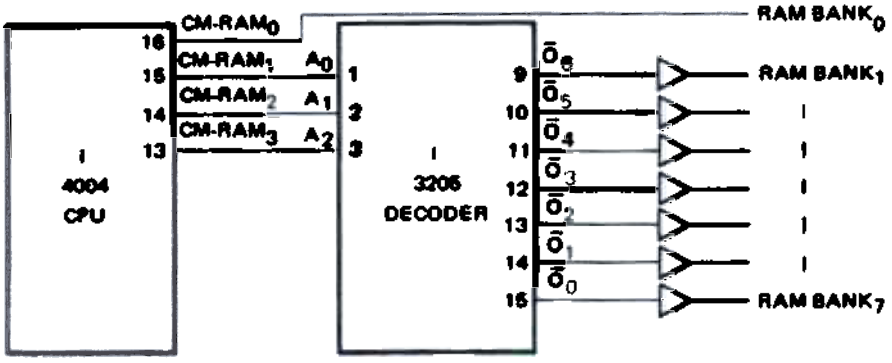
(ACC) before KBP		(ACC) after KBP
0 0 0 0	→	0 0 0 0
0 0 0 1	→	0 0 0 1
0 0 1 0	→	0 0 1 0
0 1 0 0	→	0 0 1 1
1 0 0 0	→	0 1 0 0
0 0 1 1	→	1 1 1 1
0 1 0 1	→	1 1 1 1
0 1 1 0	→	1 1 1 1
0 1 1 1	→	1 1 1 1
1 0 0 1	→	1 1 1 1
1 0 1 0	→	1 1 1 1
1 0 1 1	→	1 1 1 1
1 1 0 0	→	1 1 1 1
1 1 0 1	→	1 1 1 1
1 1 1 0	→	1 1 1 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 command 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.
X000	CM - RAM$_0$	
X001	CM - RAM$_1$	
X010	CM - RAM$_2$	
X100	CM - RAM$_3$	
X011	CM - RAM$_1$, CM - RAM$_2$	
X101	CM - RAM$_1$, CM - RAM$_3$	
X110	CM - RAM$_2$, CM - RAM$_3$	
X111	CM - RAM$_1$, CM - RAM$_2$, CM - RAM$_3$	

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.



IX. 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	0 0 0 0 (Jump to ROM memory Location # 16)

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

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

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

```
                /Rotate left (accumulator and carry)

                /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.

           15   /Exchange contents of index register 15 and accumulator

            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

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

           15
```

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
		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
		WMP		/Write contents of accumulator into RAM output ports
		ADD	15	/Add contents of Register 15 to accumulator
		WMP		/Write contents of accumulator into RAM output ports
		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

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

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

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

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:

Complement of highest order bit of accumulator and Send back to RAM output port (Instruction #8 and 9)

↓

↑ Initial state of RAM output port

↑ Highest order bit of accumulator is complemented again and sent back to the RAM output port (Instructions 10 and 11)

(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.

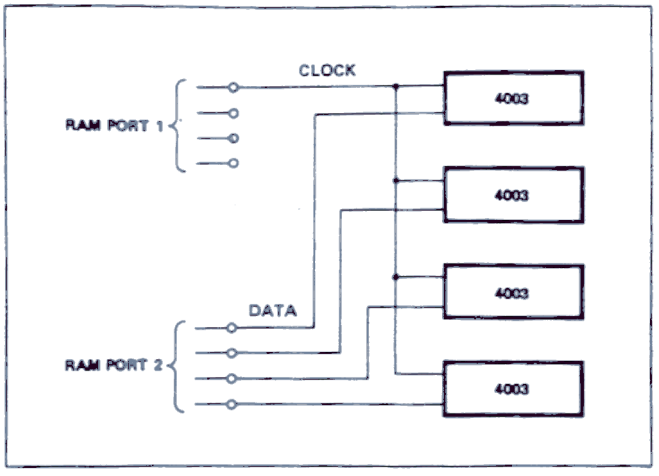


Figure 9. RAM Output Ports Driving Groups of Shift Registers

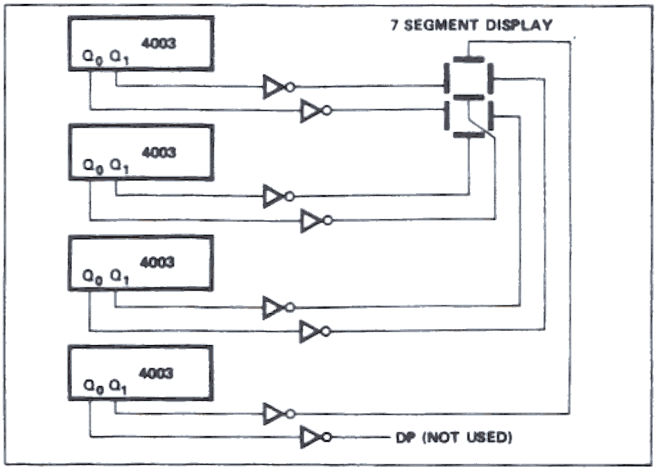


Figure 10. Shift Registers Driving Seven Segment LED Displays

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 1st 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.

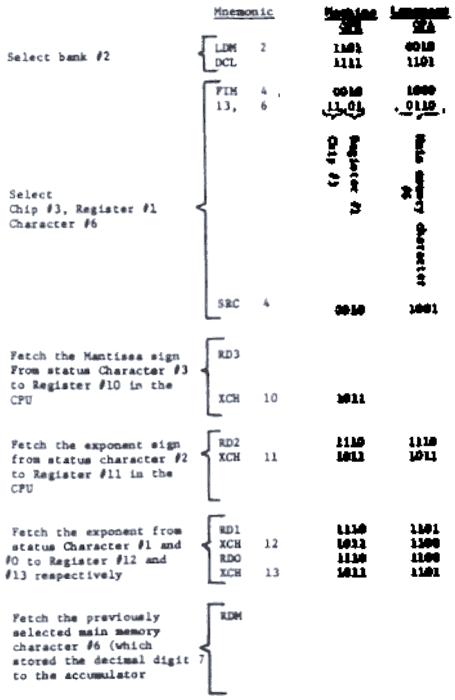
For example let's store the previously shown number in Bank #2, Chip number #3, register #1. It would be stored in the 4002 as follows:

	Register #1				
Decimal digit - 6	0	1	1	0	0
Decimal digit - 0	0	0	0	0	1
Decimal digit - 4	0	1	0	0	2
Decimal digit - 7	0	1	1	1	3
Decimal digit - 8	1	0	0	0	4
Decimal digit - 3	0	0	1	1	5
Decimal digit - 7	0	1	1	1	6
Decimal digit - 5	0	1	0	1	7
Decimal digit - 1	0	0	0	1	8
Decimal digit - 4	0	1	0	0	9
Decimal digit - 9	1	0	0	1	10
Decimal digit - 9	1	0	0	1	11
Decimal digit - 2	0	0	1	0	12
Decimal digit - 7	0	1	1	1	13
Decimal digit - 3	0	0	1	1	14
Decimal digit - 1	0	0	0	1	15
Exponent Value 59	1	0	0	1	0
	0	1	0	1	1
Exponent Sign - Negative	0	0	0	1	2
Mantissa Sign - Positive	0	0	0	0	3

Main Memory Character #

Status Character #

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.

X. 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 1st 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

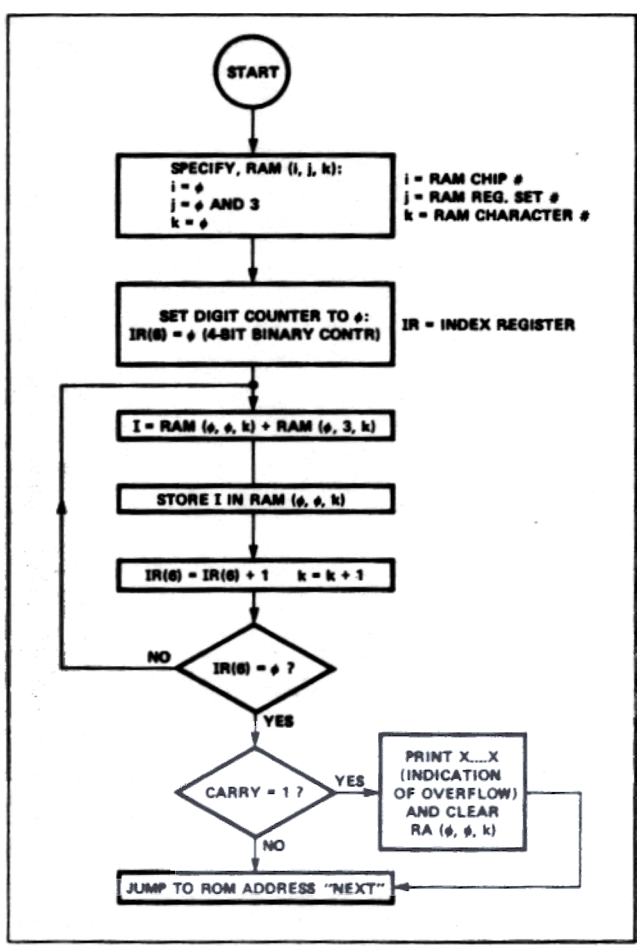


Figure 11. Flow Chart for 16 Digit Decimal Routine

16-DIGIT DECIMAL ADDITION ROUTINE

```
0000 0040
0001 0000 ADDIN, 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
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             / WHITE AC TO RAM
0022 0361          CLC             / CLEAR CARRY REG
0023 0243          LD 3            / LOAD AC WITH C(IR(3))
0024 0295          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 0294          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
```

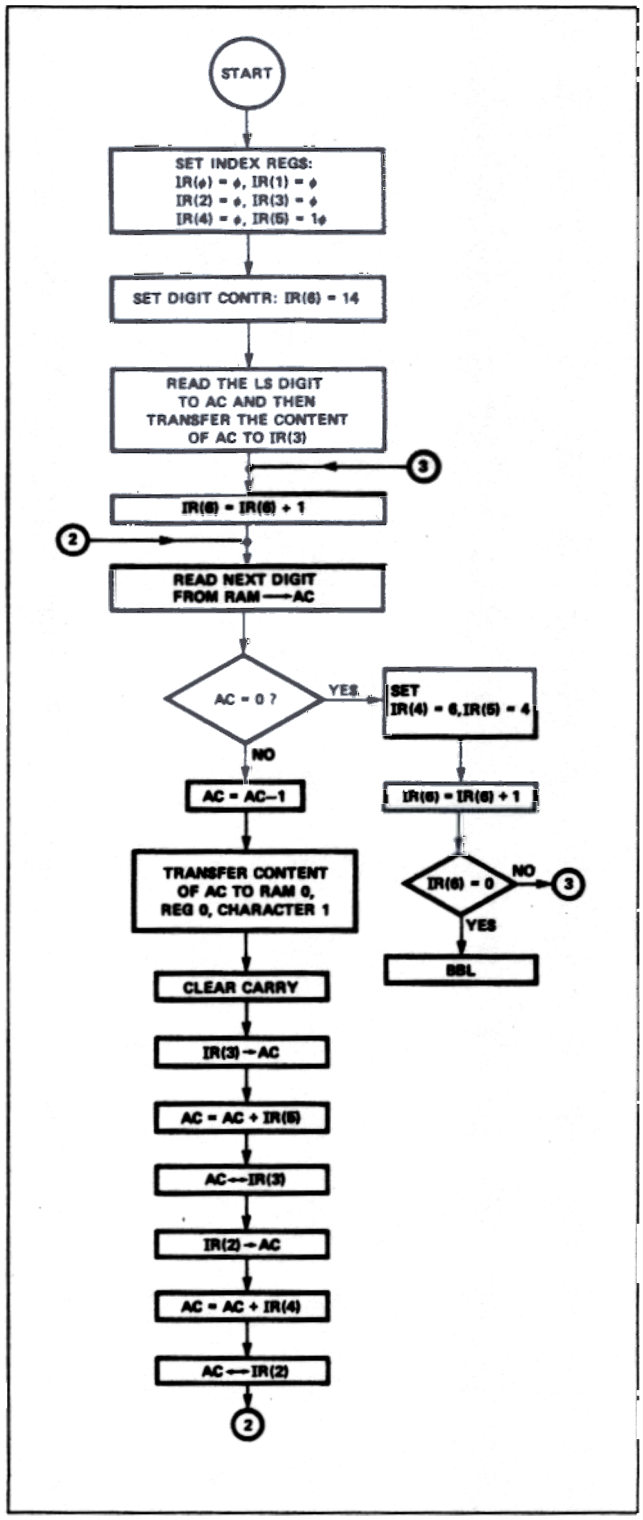


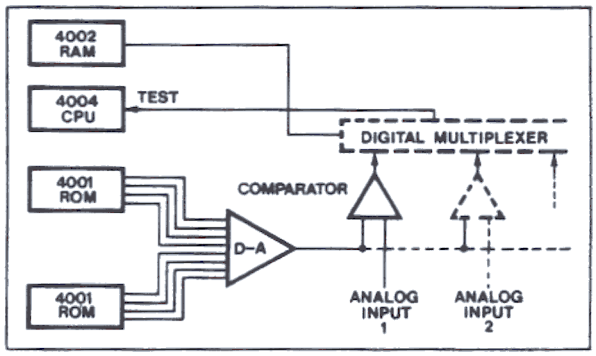
Figure 12. Flow Chart for BCD to Binary Conversion

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 LIBRARY

MCS-4 Cross Assembler and Simulator Software Package

Intel now offers an assembler and simulator software package to help develop programs for microcomputer systems built from Intel's MCS-4 set of integrated computer circuits. The MCS-4 cross assembler translates a symbolic representation of the instructions and data into a form which can be loaded and executed by the MCS-4. By cross assembler, we mean an assembler executing on a machine other than the MCS-4 which generates object code for the MCS-4. Initial development time can be significantly reduced by taking advantage of a large scale computer's processing, editing and high speed peripheral capability.

The software is written in general FORTRAN IV. The package consists of a simulating routine, which enables the computer to simulate the operation of an MCS-4 microcomputer, and an assembly routine, used primarily as an aid to programming the simulated microcomputer.

The routines may be procured from Intel on magnetic tape. Alternatively, designers may contact nation-wide computer time-sharing services – General Electric and Tymshare – for access to the programs.

MCS-4 User's Library

- Cross assembler and simulator for the MCS-4 that runs on the PDP-8.
- MCS-4 logic subroutines AND, XOR, IOR, LOGIC.
- Sixteen digit Decimal Addition Routine (A0700)[1]
- Cross Assembler for NOVA
- Chebychev polynominal approximation subroutines for addition, subtraction, multiplication, division, sine, cosine, arctangent, exponential, and natural logs.

These program listings are available to all members of the microcomputer user's library. 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.

XI 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 $\emptyset$ = 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 "0" 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 "0", the result will be equivalent to a logic "0". By scanning a keyboard with a moving "0", multiple key presses in a row can be resolved. Furthermore, if the 4003 is disabled, all outputs go to logic "0" and all keys can be sampled simutaneously to determine if a scan is required.

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

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

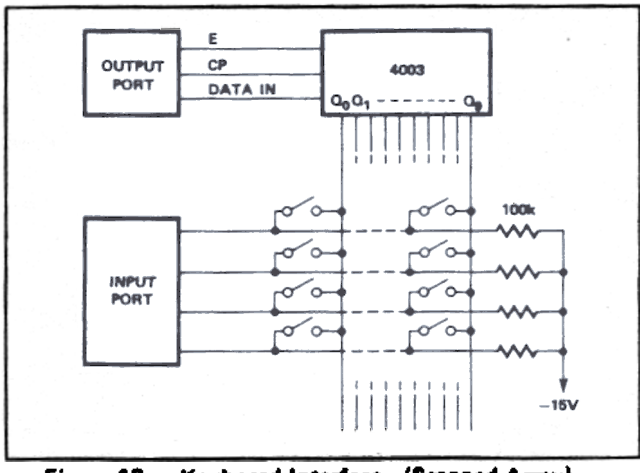


Figure 13. Keyboard Interface - (Scanned Array)

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.

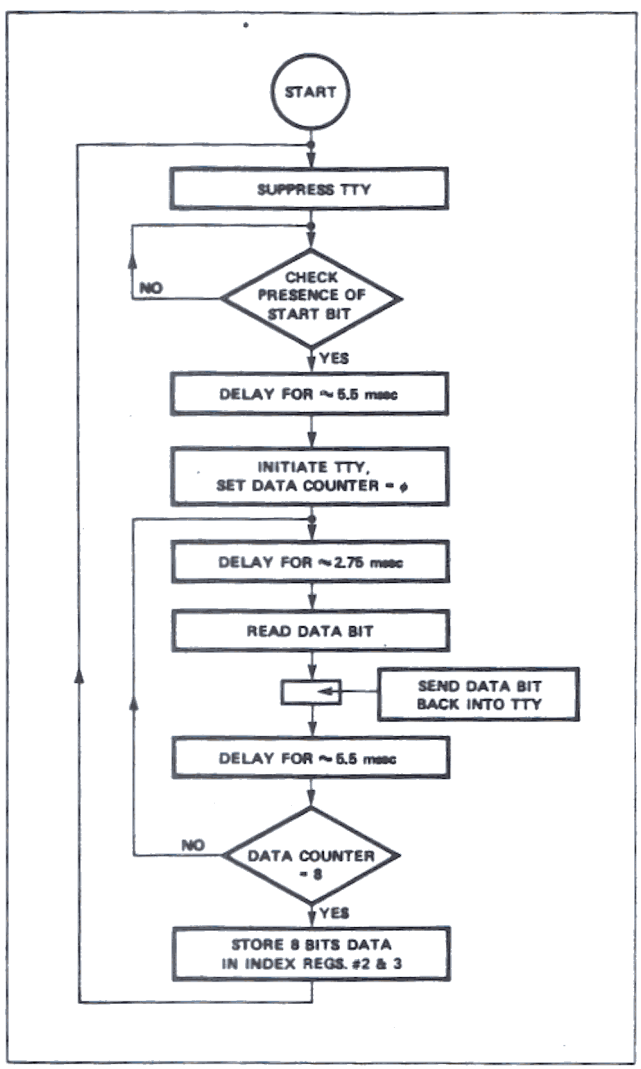


Figure 14. Flow Chart for Teletype Interface

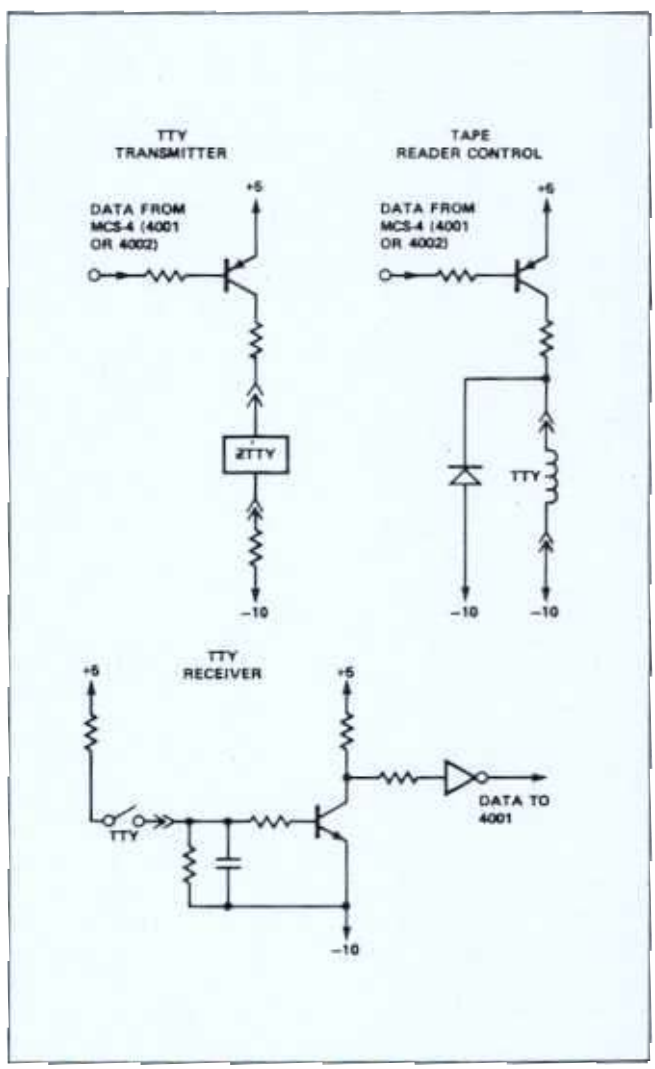


Figure 15. MCS-4 & Teletype Interface Circuits

KEYBOARD INPUT ROUTINE

```
0000 0337 BEGIN, LDM 15          /SILENCE TTY
0001 0040
0002 0000        FIM 0<,0        / BY SETTING 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 8           / 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
                 /
                 /
```

XII. DEVELOPMENT AIDS

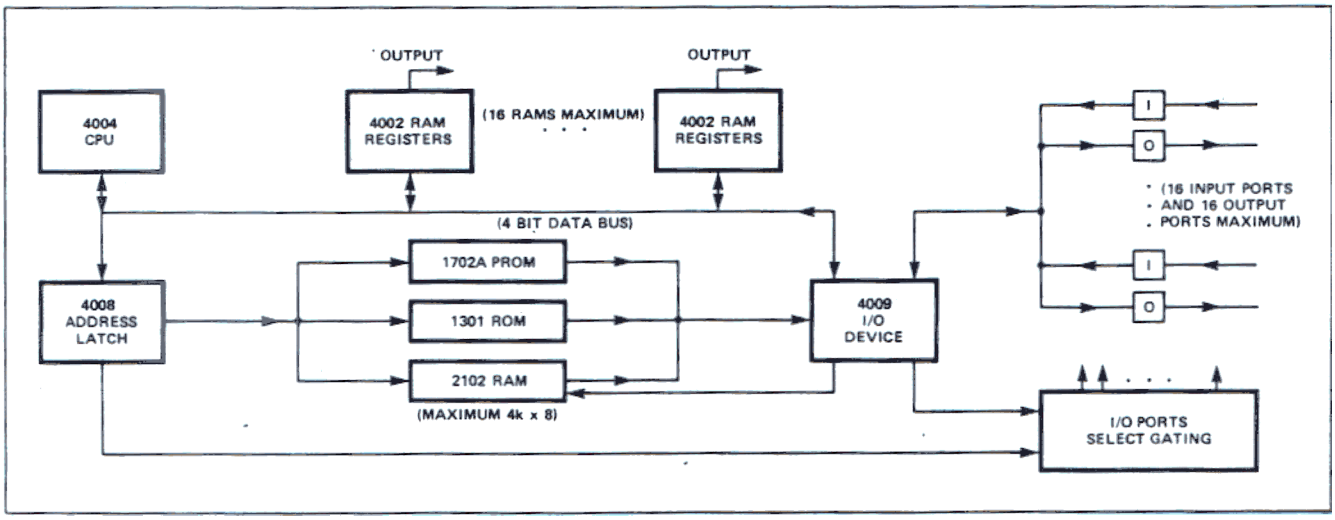
A. MCS-4 Standard Memory and Interface Set (4008/4009)

Prototype systems 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.

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

B. PA4-04 Program Analyzer for MCS-4 Development System

The PA4-04 Program Analyzer is a compact (9" x 9" x 1.5") portable unit providing a powerful real-time analysis capability for MCS-4 users. It was designed as an MCS-4 development tool and for convenient field service of microcomputer systems. It can also be used with any of the SIM systems or the imm4-42 module. Applications consist of software and system debugging, CPU data logging, program event detector, address comparator, binary display unit, and trouble shooting in the field.

The analyzer connects to the 4004 CPU via a 16 pin DIP-CLIP and displays all of the significant CPU parameters. LED displays thus latch and display the contents of the four bit data bus displaying the address sent out by the CPU, the instruction received back from ROM and the execution by the CPU. Displays also indicate which CM-RAM line is active and what the last RAM/ROM point is (SRC-instructions). In the free running mode this display is naturally changing as the program runs.

Provisions have been made for examining the contents of the data bus and the status of the CPU at selected points in the program. This is done by entering the selected instruction number into the SEARCH ADDRESS switches provided on the front panel. Now as the program runs the PA4-04 will latch the data at the selected instruction number. The display will hold until the reset button is hit (which also applies a reset pulse to the MCS-4 system being operated on).

While the display of the search address is latched, the next instruction can be examined by hitting the NEXT INSTRUCTION switch. Pushing the INCREMENT button will increment the program one more count and this can be continued indefinitely. The previous instruction can be examined by using the DECREMENT switch in the same fashion.

A switch selectable pass counter provides interrogation of program loops by delaying the display until after a preset number of passes (1 to 15) have been made through the preset SEARCH ADDRESS.

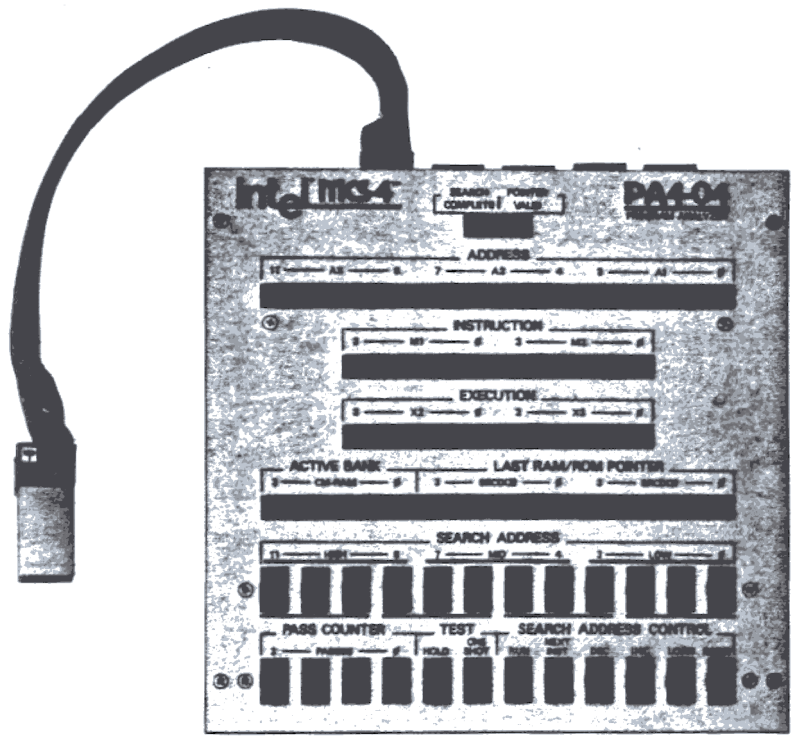
SEARCH CONTROL and TEST switches provide additional features for easy program debugging.

All displayed parameters are also accessible in buffered TTL form via external 16-pin DIP sockets on the back panel. This allows for external monitoring needed for data logging applications.

The PA4-04 requires a single external power supply (+5V DC, 2.0A) which is connected to banana plug provided on the back panel.

Operating Procedures

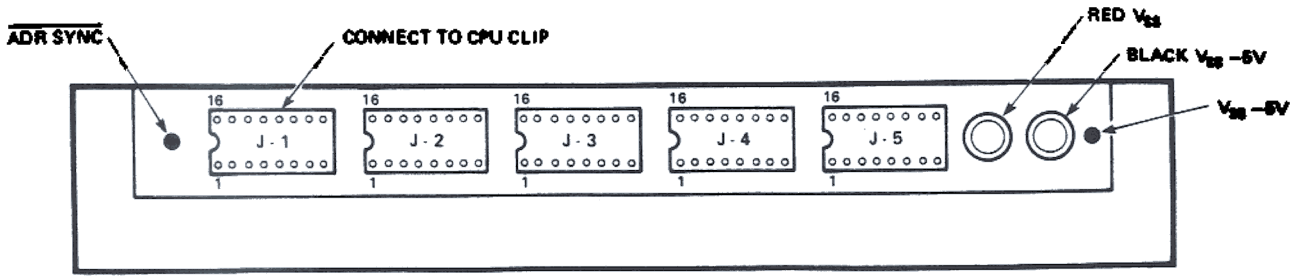
1. Connect 5 volt power supply to analyzer via banana plug connectors. (Connect ground to MCS-4 system common as well.)
2. Connect analyzer to 4004 CPU via DIP-CLIP connector.
3. Set "SEARCH ADDRESS" switches to desired program address.



4. Set "PASS CNTR" switches to desired number of passes.
5. Push "LOAD" (push "RESET" also if execution from location zero desired as well.
6. The "SEARCH COMPLETE" indicator will light when the CPU executes the preset number of passes of the search address or after execution of the next instruction after the search address/pass count ("NEXT-INST" switch on). All CPU data displays will latch-up at "SEARCH COMPLETE" time.
7. The "POINTER VALID" indicator will light when the CPU executes an SRC instruction. The "LAST RAM/ROM POINTER" will display the last SRC executed before "SEARCH COMPLETE" time.
8. Further program analysis can be made by performing any one of the following:
   a. Push "INC" to increment search address by one location. (Push "RESET" switch also if execution from location zero desired as well.)
   b. Push "DEC" to decrement search address by one location. (Push "RESET" switch also if execution from location zero desired as well.)
   c. Set "SEARCH ADDRESS" and/or "PASS CNTR" switches to desired new setting. Push "LOAD" (push "RESET" also if execution from location zero desired as well).
   d. Push "NEXT INST" switch and perform any one of the above operations for analysis of altered-sequence program flow (JUN, JIN, ISZ, JMS, JCN, BBL).
9. Miscellaneous operations:
   a. Push "TEST ONE SHOT" switch to assert CPU test line for 2 instruction cycles.
   b. Push "TEST HOLD" switch to assert CPU test line indefinitely.
   c. Push "RUN" switch to provide a sync pulse at the preset search address. (CPU data display does not latch up in this mode. In this mode the "PASS CNTR" switches must be set to zero.)

INSTRUCTIONS	Data @ X2 $D_3 D_2 D_1 D_0$	Data @ X3 $D_3 D_2 D_1 D_0$	COMMENTS
• NOP	1 1 1 1	1 1 1 1	
• JCN A_2, A_1	1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1	
• FIM RRR0 D_2 D_1	(RRR0) 1 1 1 1	(RRR1) 1 1 1 1	The content of address pair RRR
• SRC RRR1	(RRR0)	(RRR1)	
• FIN RRR0 2nd cycle	(RRR0) 1 1 1 1	(RRR1) 1 1 1 1	
• JIN RRR0	(RRR0)	(RRR1)	
• JUN A_3 A_2, A_1	A_3 A_3	1 1 1 1 1 1 1 1	
• JMS A_3 A_2, A_1	A_3 A_3	1 1 1 1 1 1 1 1	
• INC RRRR	**(RRRR)**	**(RRRR) + 1**	Content of register RRRR;
• ISZ RRRR A_2, A_1	(RRRR) 1 1 1 1	(RRRR) + 1 1 1 1 1	Content +1 of RRRR
• ADD RRRR	(RRRR)	1 1 1 1	Content of register RRRR
• SUB RRRR	(RRRR)	1 1 1 1	
• LD RRRR	(RRRR)	1 1 1 1	
• XCH RRRR	(RRRR)	(ACC)	Content of register RRRR; the content of ACC
• BBL	DDDD	1 1 1 1	Data DDDD
• LDM	**DDDD**	**1 1 1 1**	Data DDDD
• WRM, WR0, WR1, WR2, WR3, WPM, WMP, WRR	(ACC) (ACC)	1 1 1 (CY) 1 1 1 (CY)	Content of accumulator; Content of CY F/F is present on D_0
• RDM, RD0, RD1, RD2 RD3, ADM, SBM, RDR	(M) or (Input)	(M) or (INPUT)	Data fetched from RAM or input
• CLB, CLC, IAC, CMC CMA, RAL, PAR, TCC	0 0 0 0	1 1 1 1	
• TCS	1 0 0 1	1 1 1 1	
• STC, DAC, DCL	1 1 1 1	1 1 1 1	
• DAA	0 0 0 0 or 0 1 1 0	1 1 1 1	X_2 depends on ACC content
• KBP	0000, 0001, 0010 0011, 0100, 1111	1 1 1 1	X_2 depends on ACC content

Summary of MCS-4 Data Bus Content During Execution of Each Instruction



J - 1

PIN	CPU In	
1	D0	DATA BUS
2	D1	
3	D2	
4	D3	
5	V_{SS}	
6	ϕ_1	CLOCK
7	ϕ_2	
8	SYNC	CPU CONTROL
9	RESET	
10	TEST	
11	CM-ROM	
12	V_{DD}	
13	CM-RAM3	MEMORY CONTROL
14	CM-RAM2	
15	CM-RAM1	
16	CM-RAM0	

J - 2

PIN	CPU OUT	
1	DB0	DATA BUS
2	DB1	
3	DB2	
4	DB3	
5	V_{SS}	
6	ϕ1T	CLOCK
7	ϕ2T	
8	SYNCT	CPU CONTROL
9	$\overline{\text{RESETT}}$	
10	TESTT	
11	CMRM	
12	V_{DD}	
13	CMR3	MEMORY CONTROL
14	CMR2	
15	CMR1	
16	CMR0	

J - 3

PIN	CTL OUT	
1	$\overline{\text{X3STB}}$	STATE STROBES
2	$\overline{\text{A1STB}}$	
3	$\overline{\text{A2STB}}$	
4	$\overline{\text{A3STB}}$	
5	$\overline{\text{M1STB}}$	
6	$\overline{\text{M2STB}}$	
7	$\overline{\text{X1STB}}$	
8	$\overline{\text{X2STB}}$	
9	SOC SYNC	
10	RST	
11	$\overline{\text{SRC UPDATE}}$	
12	$\overline{\text{SRC (X2) STB}}$	
13	$\overline{\text{SRC (X3) STB}}$	
14	$\overline{\text{POINTER VALID}}$	
15	ADR CMP	
16	$\overline{\text{SEARCH COMPLETE}}$	

J - 4

PIN	ADR OUT	
1	$\overline{\text{A0}}$	ADDRESS OUT
2	$\overline{\text{A1}}$	
3	$\overline{\text{A2}}$	
4	$\overline{\text{A3}}$	
5	$\overline{\text{A4}}$	
6	$\overline{\text{A5}}$	
7	$\overline{\text{A6}}$	
8	$\overline{\text{A7}}$	
9	$\overline{\text{A8}}$	
10	$\overline{\text{A9}}$	
11	$\overline{\text{A10}}$	
12	$\overline{\text{A11}}$	
13	SA CMP	
14	PC CMP	
15	ADR CMP	
16	$\overline{\text{ADR SYNC}}$	

J - 5

PIN	DATA OUT	
1	$\overline{\text{M10}}$	M1 DATA
2	$\overline{\text{M11}}$	
3	$\overline{\text{M12}}$	
4	$\overline{\text{M13}}$	
5	$\overline{\text{M20}}$	M2 DATA
6	$\overline{\text{M21}}$	
7	$\overline{\text{M22}}$	
8	$\overline{\text{M23}}$	
9	$\overline{\text{X20}}$	X2 DATA
10	$\overline{\text{X21}}$	
11	$\overline{\text{X22}}$	
12	$\overline{\text{X23}}$	
13	$\overline{\text{X30}}$	X3 DATA
14	$\overline{\text{X31}}$	
15	$\overline{\text{X32}}$	
16	$\overline{\text{X33}}$	

NOTES:

1. Connect external 5V (2A) power supply to rear panel banana jacks. The supply must be voltage compatible with the MCS-4 system being analyzed. Red jack = V_{SS} Black jack = V_{SS} –5V. SIM4-03 systems require Red = +5V, Black = GND (common to system GND).
2. Use Test and Reset functions of PA4 only if the MCS-4 system being analyzed has a passive pull-up test and reset driver. (SIM4-03 has passive driver. Use on-board Test and Reset CKT for SIM4-01 and SIM4-02.)
3. All signals are positive logic and TTL compatible except J1-CPU in signals from 4004. (If V_{SS} = +5V, V_{DD} = 10V.)

PA4-04 Rear Panel Layout

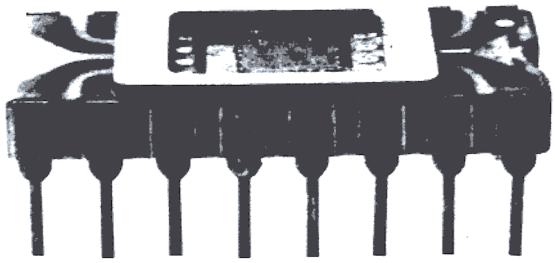
C. Intellec 4

Features

- Ideal for prototyping MCS 4 Systems.
- The Intellec 4 is a complete microcomputer system with 5K bytes of program memory, data storage, I/O, TTY interface, standard software, control panel, power supplies, and a compact finished cabinet (less than 0.8 ft.^{3}).
- The heart of the Intellec system is Intel's four-bit "computer on a chip," the 4004. This is a 4-bit parallel CPU with a repertoire of 46 instructions, sixteen working registers, a four level address stack, and capability of directly addressing over 43K bits of memory.
- Direct access to memory via control console.
- Standard software provided with the Intellec 4 includes a system monitor which provides a loader, hex memory dump, and instruction editing, and an assembler which is loaded to RAM program memory.
- With this system, all program development may be done in RAM memory.
- A complete PROM programmer is provided as an option. After the program is firm, it may be committed to non-volatile storage in Intel's 1702A programmable and erasable Read-Only-Memory.
- Complete system control and hardware debugging aids are provided via the control panel.
- Crystal clocks are provided for system stability.
- System is expandable to 12 microcomputer modules in a single chassis.

Specifications

Word Size:	Data: 4 bits Instruction: 8 or 16 bits
Memory Size:	Instruction Memory: 1K Bytes (8 bits) in PROM switchable to 4K bytes (8 bits) RAM Data Storage: 320 words (4 bit), expandable to 2560 words
Instruction Set:	46, including conditional branching, binary and decimal, arithmetic, register-to-register and I/O
System Clock:	Crystal Controlled
Machine Cycle Time:	10.8 μs
Memory Access:	Standard via the control panel
Memory Cycle Time:	900 nanoseconds
I/O Channels:	4 input ports, 8 output ports, expandable to 16 input ports, 48 output ports
Operating Temperature:	0°C to 55°C
Power Supplies:	+5v ±5% 8 amps* −10v ±5% 1.8 amps* *Larger power supplies may be required for expanded systems.
Physical Size:	Intellec 4: 7" x 17⅛" x 12¼" (table top only)
Weight:	30 lb.
Standard Software:	System Monitor Resident Assembler
Support Software:	Cross Assembler, Simulator } written in FOR-TRAN IV



The 4004 Central Processor Chip is the heart of each Intellec 4 system.

Standard Systems and Optional Modules

INTELLEC 4(imm4-40A) Standard System includes the following Modules and Accessories:

Central Processor Module
Control Module
RAM Memory Module
PROM Programmer Module
Chassis with Mother Board
Power Supplies
Control and Display Panel
Finished Cabinet
Standard Software
System Monitor
Resident Assembler

OPTIONAL MODULES available for the Intellec 4 and Bare Bones 4:

Instruction/Data Storage Module
Input/Output Module
Data Storage Module
ROM Memory Module
Universal Prototype Module
Module Extender
Drawer Slides and extenders for Rack Mounting

Software

Standard All peripheral interface to Intellec 4 standard software is via TTY, model ASR33. All control after system start-up is provided through the TTY.

A. System Monitor

1. Contained in four 1702A PROMs located on the Central Processor Module.
2. Intellec 4 modular microcomputer systems have a control program called a Resident Monitor in PROM so that no "bootstrap" operation need ever be performed.

   The monitor functions are as follows:

   a. Load RAM memory from paper tape, either in BNPF format or hexadecimal format.

   b. Display the contents of RAM memory on a printer.

   c. Modify individual bytes of RAM memory, move blocks of RAM memory, fill blocks of RAM memory with constant data.

   d. Write contents of RAM memory to paper tape in either BNPF or hexadecimal format.

B. Resident Assembler

1. Translates mnemonic code to binary machine code.
2. Loaded into system RAM Memory via paper tape.
3. Data storage devices (4002 RAM) store label and symbols (eight/RAM).
4. This two pass assembler generates a program tape which is reloaded via the monitor.

Developmental Support: Cross Assembler and Simulator In addition to the standard software provided with the Intellec 4, Intel offers a cross assembler and simulator written in general FORTRAN IV and designed to operate on general purpose computers. The package consists of a simulating routine, which enables the computer to simulate the operation of an MCS-4 microcomputer set and an assembly routine used primarily as an aid to programming the simulated microcomputer.

The routines may be procured directly from Intel, or alternatively, designers may contact three nationwide computer time-sharing services—AL/COM, G.E., and Tymshare—for access to the programs.

Microcomputer Modules

Modules may be ordered individually. All modules are 8" wide, 6.18" high and use standard 100-pin connectors.

imm4-42 **Central Processor Module**

- This is a complete microcomputer system with the processor, program storage, data storage, and I/O in a single module.
- The heart of this module is Intel's 4004 single chip four-bit parallel processor—p-channel silicon gate MOS.
- Accumulator and sixteen working registers (4 bit).
- Subroutine nesting up to 3 levels
- For development work, the CPU interfaces to standard semiconductor memory elements (provided by Intel's standard memory and I/O interface set 4008/4009).
- Sockets for 1K bytes of PROM (Intel 1702A PROM) are provided.
- 320 words (4-bit) of data storage (Intel 4002) are provided.
- Four 4-bit input ports and eight 4-bit output ports (includes TTY interface).
- Bus-oriented expansion of memory and I/O.
- Two phase crystal clock.

imm4-22 **Instruction/Data Storage Module**

- This microcomputer module has memory capacity identical to the Central Processor Module and is used for expanding memory and I/O.
- Sockets for 1K bytes of PROM program storage are provided.
- 320 words (4-bit) of data storage are provided.
- Four 4-bit input ports and eight 4-bit output ports.

imm4-24 **Data Storage Module**

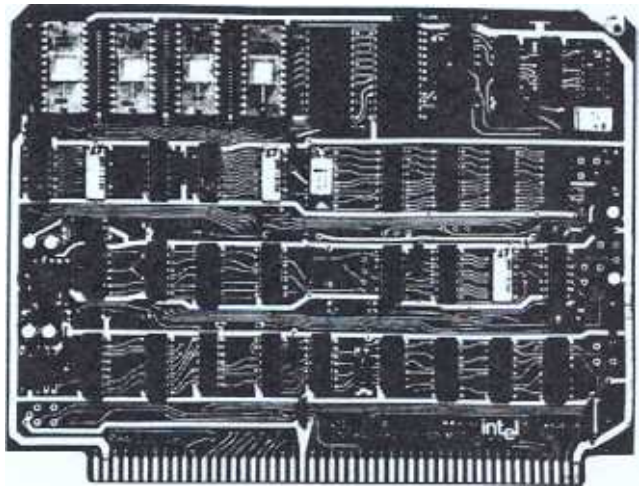
- This microcomputer module has capacity for sixteen Intel 4002 RAMS—1280 words (4-bit) of data storage.
- 320 words (4-bit) of data storage are provided.
- A maximum Intellec 4 system may contain up to 2560 words of storage—decoding for this expansion is provided.
- A 4-bit output port is associated with each RAM on this microcomputer module providing sixteen 4-bit output ports on each module.
- All output ports are TTL compatible.

imm4-60 **Input/Output Module**

- This module provides input and output port expansion without additional memory.
- Eight 4-bit input ports and eight 4-bit output ports are provided.
- Ports on this module are TTL compatible.

imm6-26 **PROM Memory Module**

- Provides sockets for up to sixteen 1702A electrically programmable and erasable PROMs for a system's fixed program memory (maximum 4K bytes/module).
- For volume requirements, Intel 2048-bit mask programmed MOS ROMs (1302) may be substituted in the same module.



imm4-42 Central Processor Module

imm6-70 **Universal Prototype Module**

- Accommodates 14, 16, 24, or 40 pin wire wrap sockets (maximum of 52 16-pin sockets)
- Provides breadboard capability for developing custom and specialized interface circuits.

imm6-72 **Module Extender**

- Extends Intellec modules out of card chassis for ease in test and system debugging.

imm4-76 **PROM Programmer Module**

- Provides all timing and level shifting circuitry for programming Intel's programmable and erasable 1702A PROMS.
- This programmer is controlled by special system software supplied with module.

Control and Display Panel

- Provides complete operator control for Intellec 4 and displays system status.
  - Address and Data Entry switches.
  - Status, instruction code, data and address displays.
- Complete program development tool.
  - ADDRESS, PROGRAM SEQUENCE, and MODE CONTROL switches permit easy examination of the program during the debugging phase of program development.
- Control and socket for 1702A PROM programming is also provided.

Chassis Module (used on the Intellec 4 and Bare Bones 4)

- Capacity for up to twelve microcomputer modules.
- PC Mother Board eliminates back plane wiring—all cards plug into common bus.
- Standard 100-pin connectors (125 mil centers) are used for all boards in the system.
- Space is provided for additional Memory, I/O modules and unique customer-developed systems interface modules.
- A fan is provided.

XIII. 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. 16 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 ($\cong$ 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.

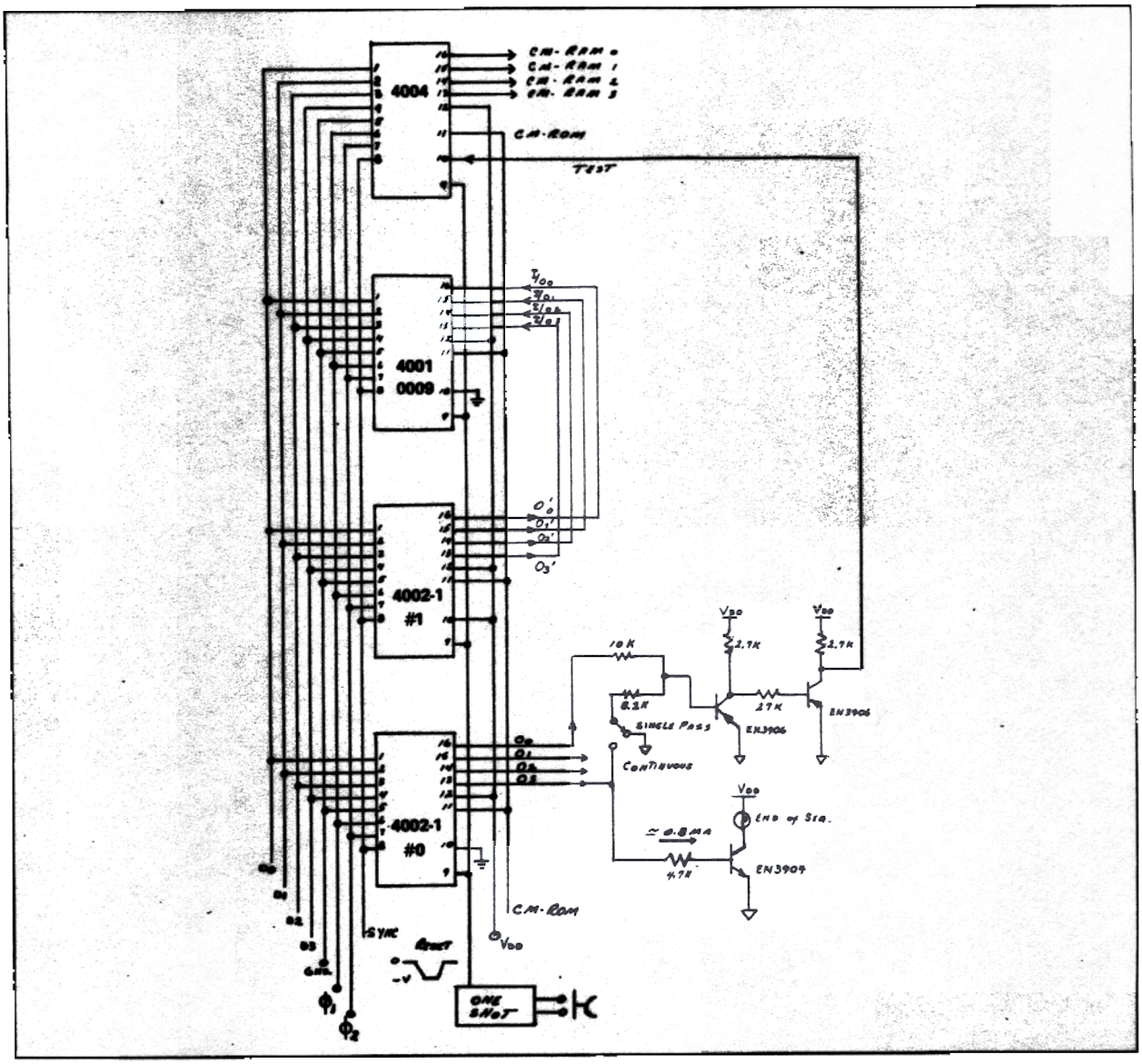


Figure 16. MCS-4 Evaluation Kit Using the 4001-0009

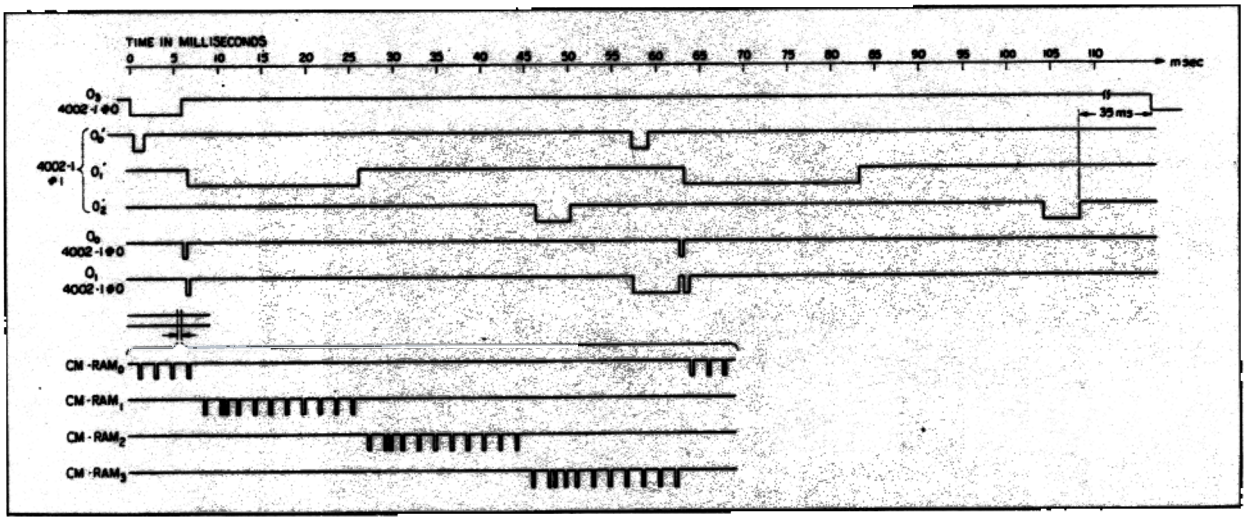
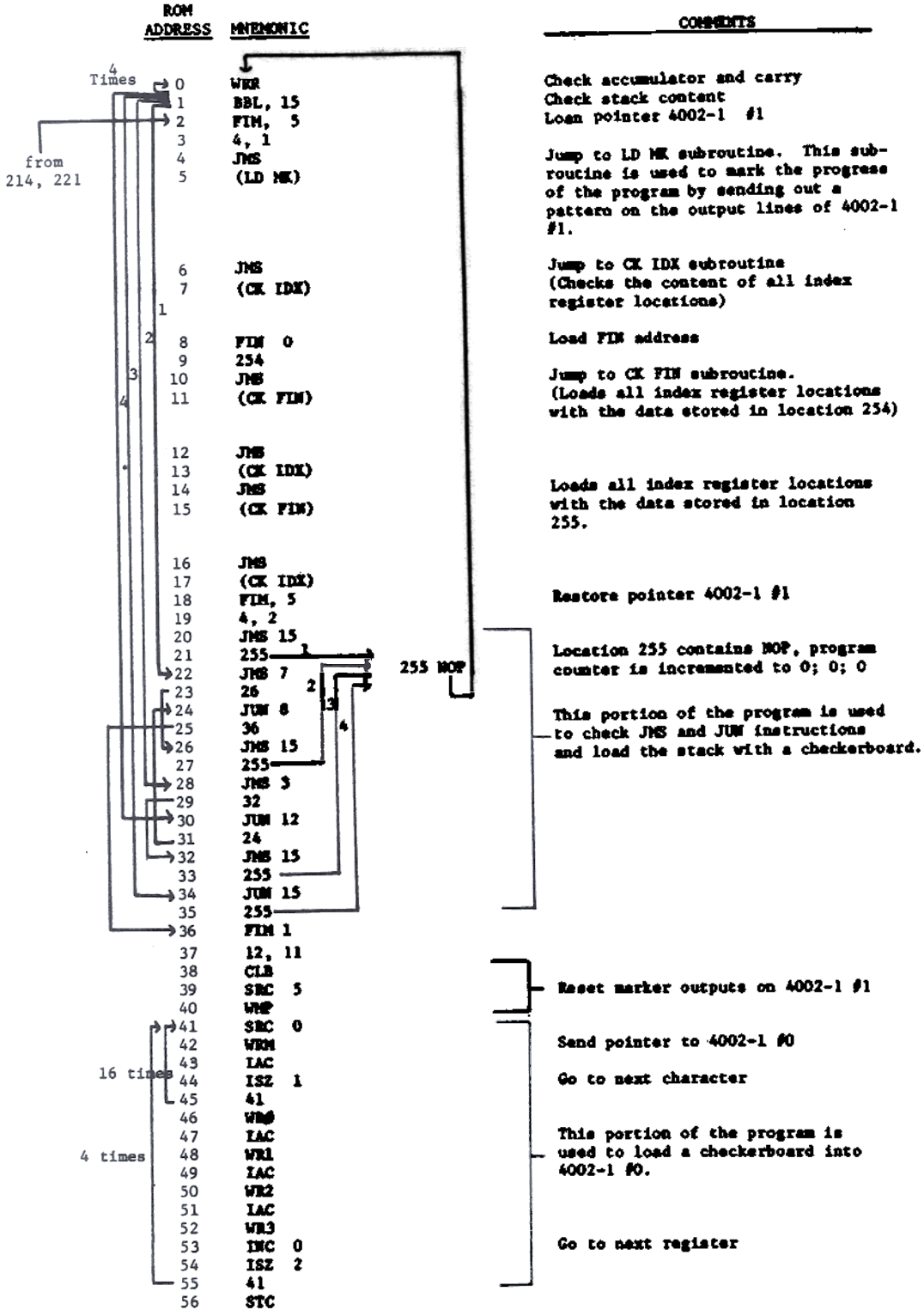
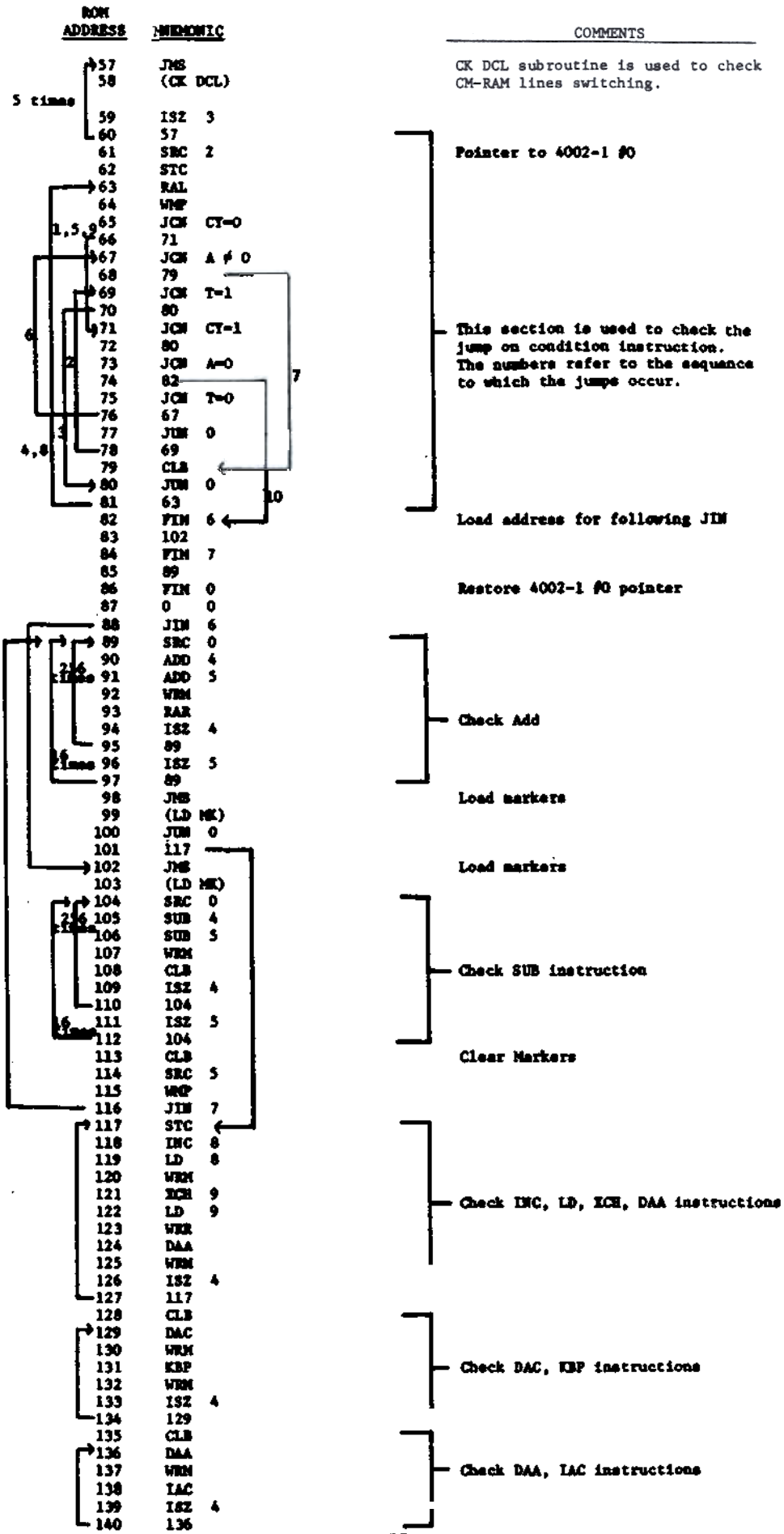
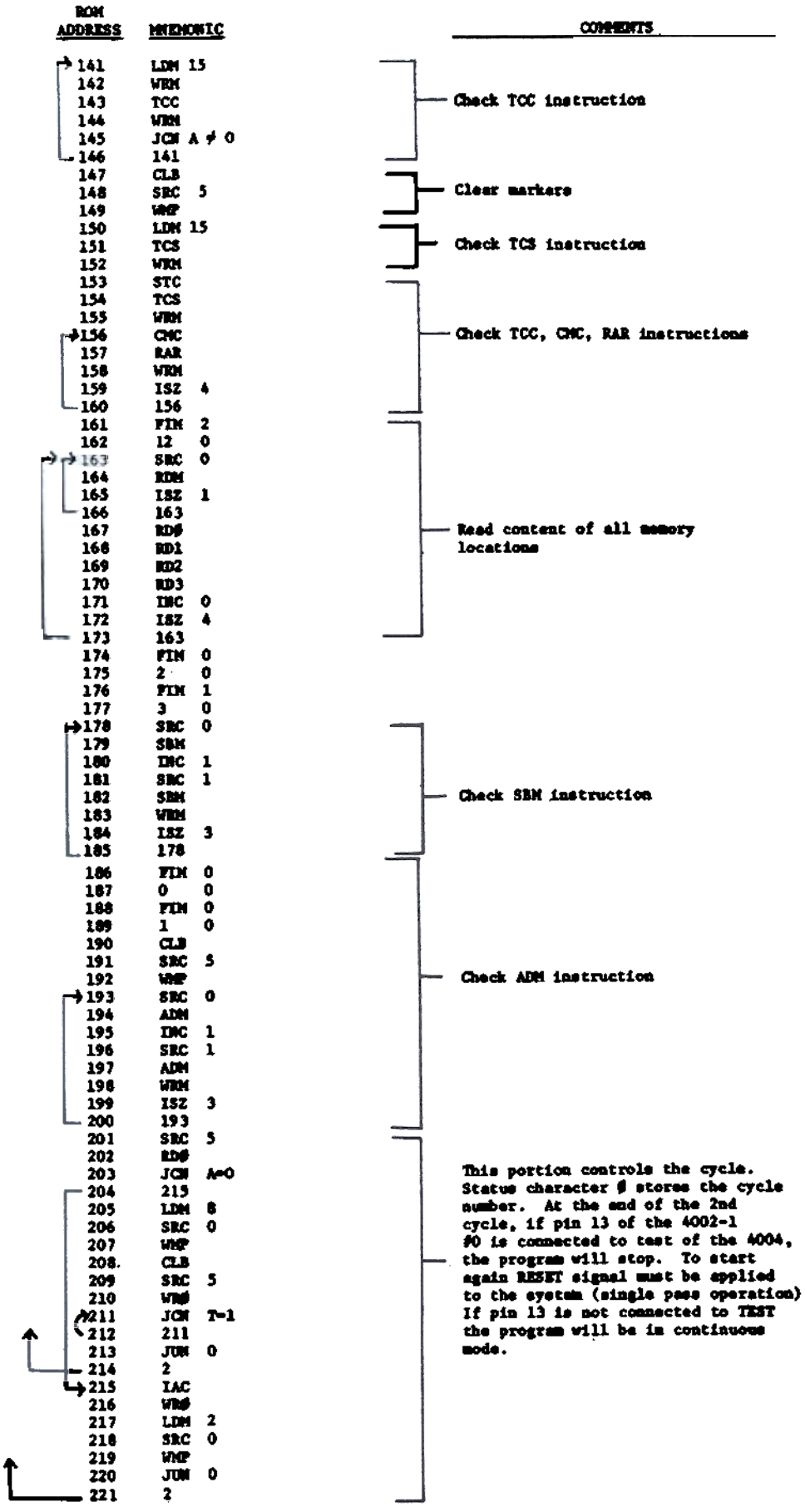


Figure 17. Timing Diagram for the MCS-4 Evaluation Kit Using the 4001-0009

4001-0009 MCS-4 EXERCISER PROGRAM




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


ROM ADDRESS
MNEMONIC
COMMENTS
141 LDM 15
142 WRM
143 TCC
144 WRM
145 JCN A ≠ 0
146 141
Check TCC instruction
147 CLB
148 SRC 5
149 WMP
Clear markers
150 LDM 15
151 TCS
152 WRM
Check TCS instruction
153 STC
154 TCS
155 WRM
156 CMC
157 RAR
158 WRM
159 ISZ 4
160 156
Check TCC, CMC, RAR instructions
161 FIM 2
162 12 0
163 SRC 0
164 RDM
165 ISZ 1
166 163
167 RD0
168 RD1
169 RD2
170 RD3
171 INC 0
172 ISZ 4
173 163
Read content of all memory locations
174 FIM 0
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
Check SBM instruction
186 FIM 0
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
Check ADM instruction
201 SRC 5
202 RD0
203 JCN A=0
204 215
205 LDM 8
206 SRC 0
207 WMP
208. CLB
209 SRC 5
210 WR0
211 JCN T=1
212 211
213 JUN 0
214 2
215 IAC
216 WR0
217 LDM 2
218 SRC 0
219 WMP
220 JUN 0
221 2
This portion controls the cycle. Status character 0 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.

```
                          222       SRC  5
                          223       LD  11
                          224       CLC
                          225       WMP
                          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
                          243       FIN  6
                          244       FIN  7
                          245       FIN  0
                          246       BBL, 0

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

                          253       NOP
Data ----------------┐    254       1111 1111
                     └    255       0000 0000 (NOP)
```

XIV. APPENDICES

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.

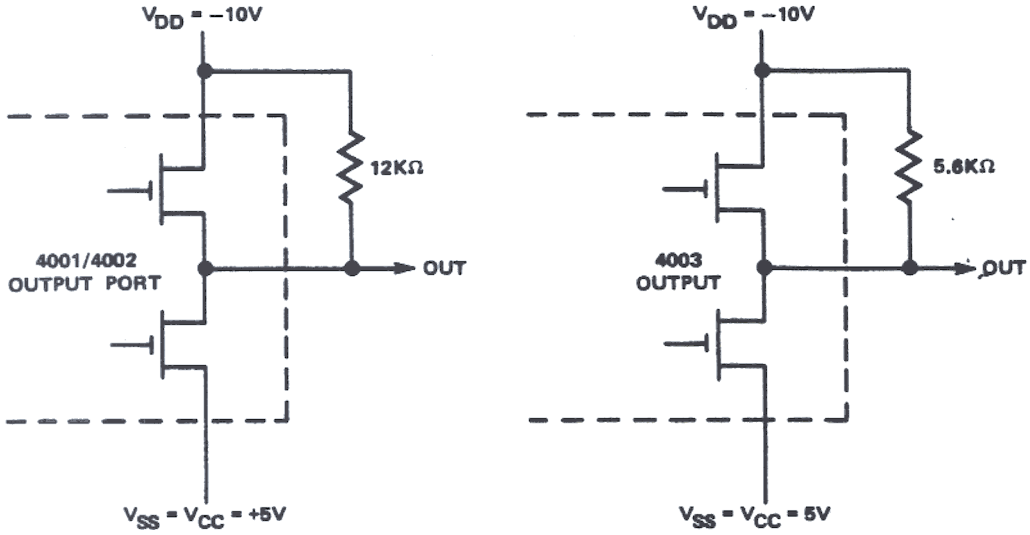


Figure 18. MCS-4 Output Configuration for TTL Compatibility

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).

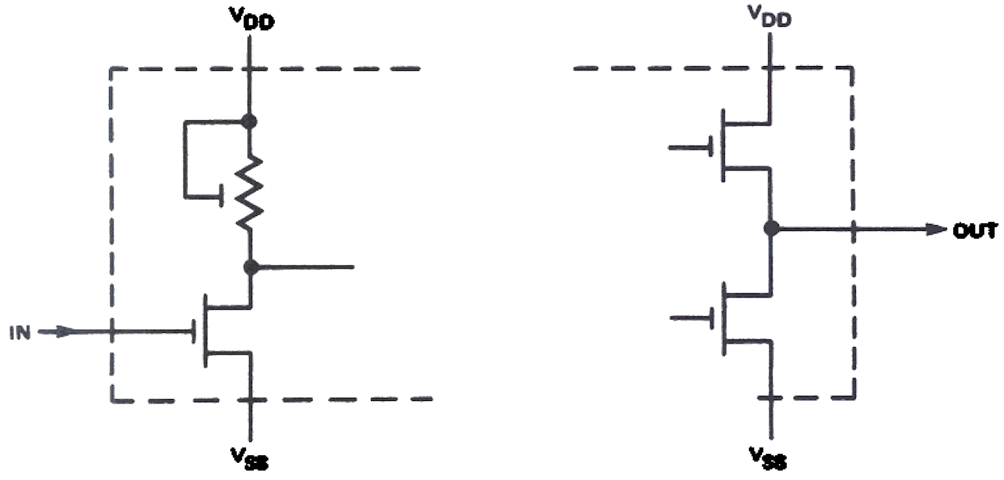


Figure 19. Typical MCS-4 Input and Output Circuitry

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

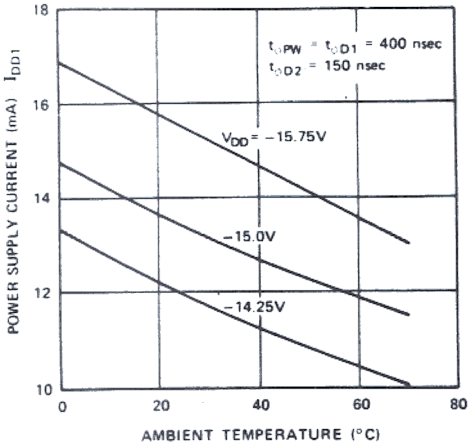
I/O OUTPUT CHARACTERISTICS

4001/2	I_{OL2}	I/O OUTPUT LINES SINKING CURRENT, "1" LEVEL	2.5	5		mA	V_{OUT} = 0V. For T^2L 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 T^2L 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} = 10μA
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

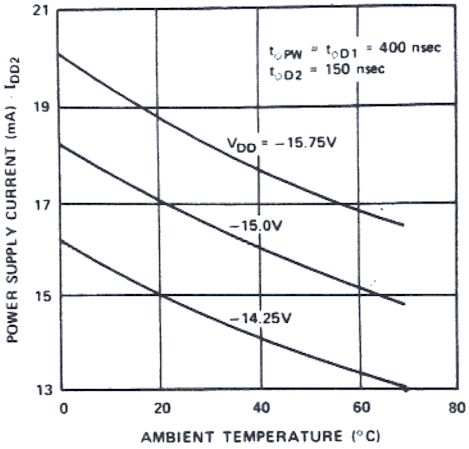
(1) Typical values are for T_A = 25°C and Nominal Supply Voltages.
(2) If non-inverting input option is used, V_{IL} = −6.5 Volts maximum.
(3) For T^2L compatibility on the I/O lines the supply voltages should be V_{DD} = −10V ± 5% V_{SS} = +5V ± 5%

Typical D. C. Characteristics

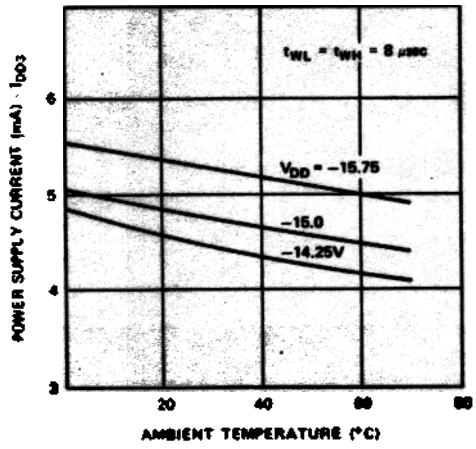
POWER SUPPLY CURRENT VS. TEMPERATURE (4001)



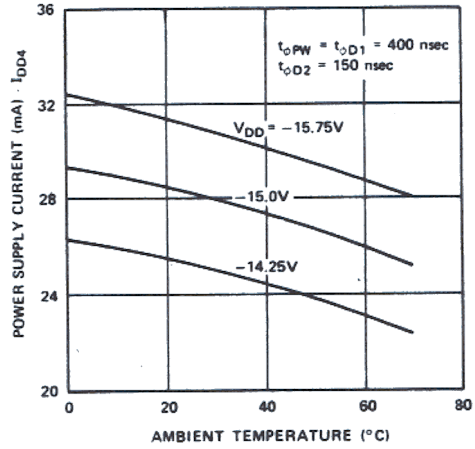
POWER SUPPLY CURRENT VS. TEMPERATURE (4002)



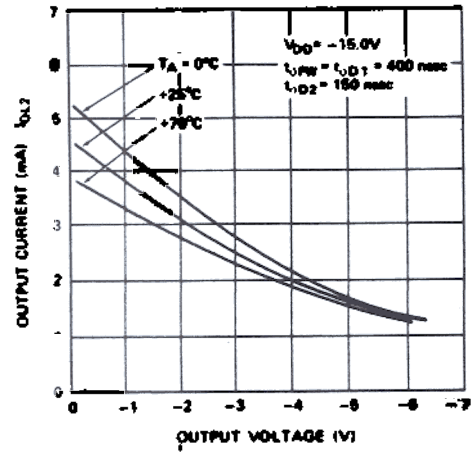
POWER SUPPLY CURRENT VS. TEMPERATURE (4003)



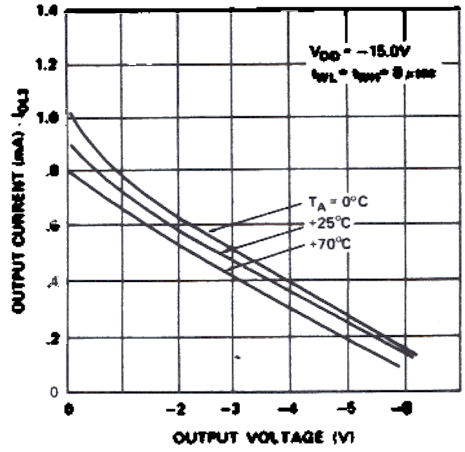
POWER SUPPLY CURRENT VS. TEMPERATURE (4004)



OUTPUT CURRENT VS. OUTPUT VOLTAGE (4001, 4002)



OUTPUT CURRENT VS. OUTPUT VOLTAGE (4003)



4001, 4002, 4004 A.C. Characteristics

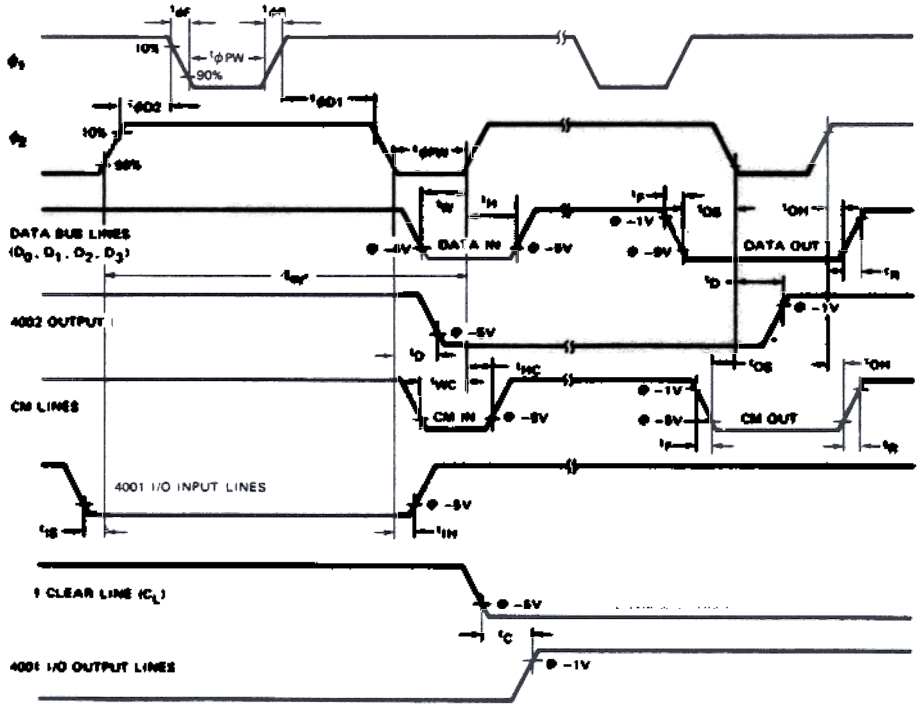
T_A = 0°C to +70°C; V_{DD} = −15V ± 5%, V_{SS} = GND

PRODUCT	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$_0$ lines are clocked out with the trailing edge of the ϕ_2 clock.
[2] The CM-ROM and the selected CM-RAM$_0$ lines are always activated during A_2 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] For 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.



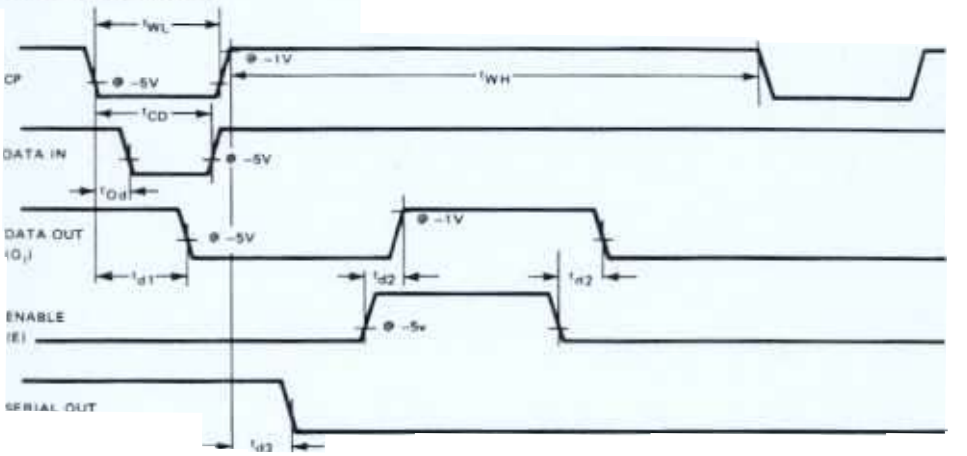
4003 A.C. Characteristics

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.

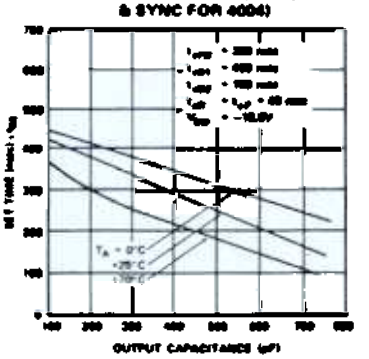
PRODUCT	SYMBOL	TEST	LIMIT (pF) TYP.	LIMIT (pF) MAX.
4001/2/3/4	C_{IN}	INPUT[1] CAPACITANCE	5	10
4001/2	$C_{\phi1}$, $C_{\phi2}$	CLOCK INPUT CAPACITANCE	8	15
4004	$C_{\phi1}$, $C_{\phi2}$	CLOCK INPUT CAPACITANCE	14	20

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

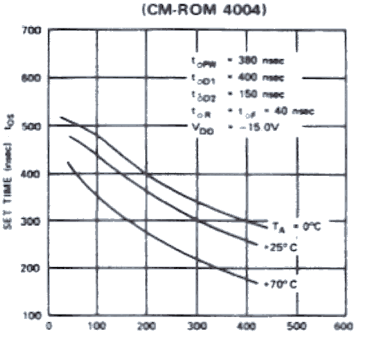
NOTE (1) Refers to all input pins except data bus I/O and ϕ_1 and ϕ_2.

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:

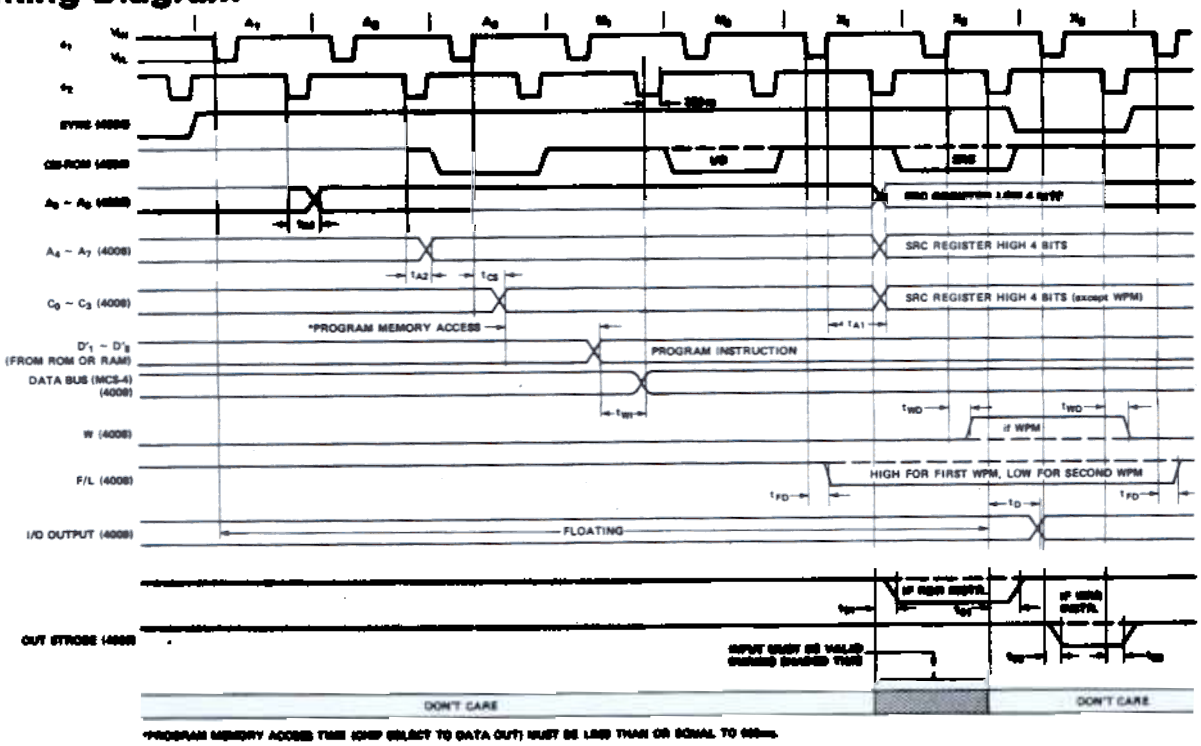
1. For TTL compatibility on the I/O lines, the supply voltages should be V_{SS} = +5V ± 5%, V_{DD} = −10V ± 5%.
2. Typical values are for T_A = 25°C and nominal supply voltages.
3. The address lines will drive a TTL load if a resistor of 470 ohms is connected in series between the address output and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

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	
$t_{\phi R}$, $t_{\phi F}$		4008/4009		50	ns	
	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_{A1}	Address to Output Delay at A_1, X_1	4008		1	μs	C_L = 250 pF
t_{A2}	Address to Output Delay A_2	4008		580	μs	C_L = 250pF
	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	470		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



Capacitance

f = 1 MHz, 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

MCS-4 CUSTOM ROM ORDER FORM

SAMPLE

4001 Metal Masked ROM

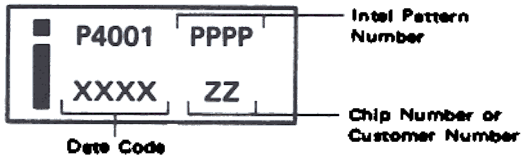
All custom ROM orders must be submitted on forms provided by Intel. Programming information should be sent in the form of computer punched cards or punched paper tape. In either case, a print-out of the truth table must accompany the order. Refer to Intel's Data Catalog for complete pattern specifications.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

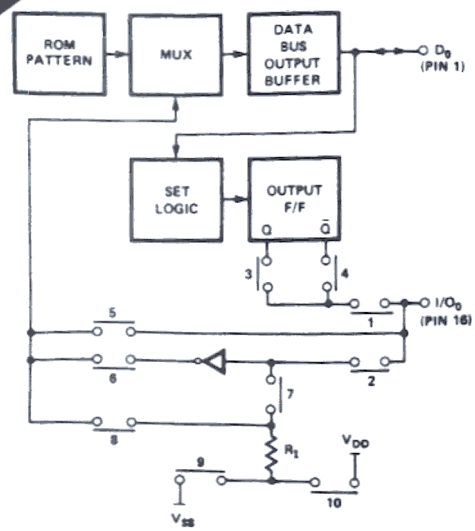
   If the pins on a port are all inputs or all outputs the internal resistors do not have to be connected.

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

4001 I/O Options

SAMPLE

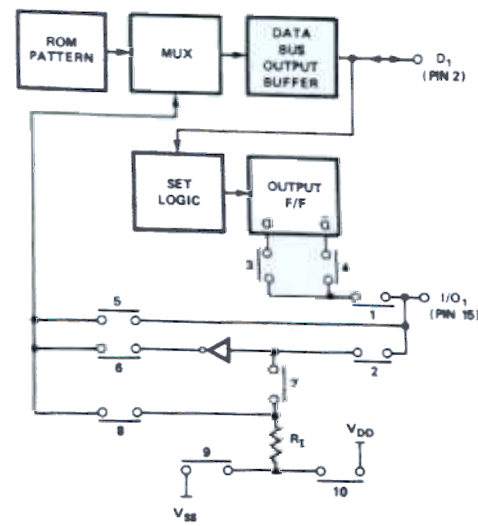


I/O_0 (PIN 16)

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 ±5%, V_{SS} = +5V ±5%

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

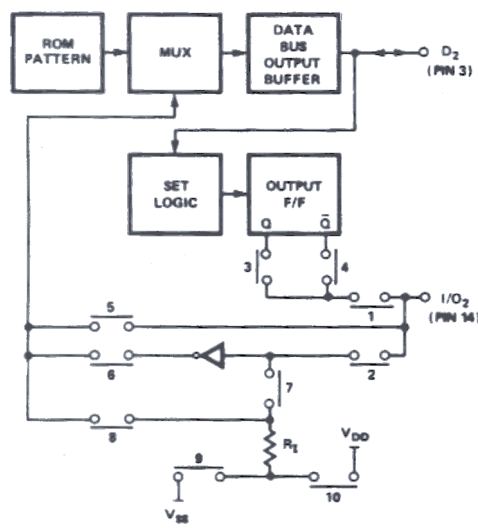


I/O_1 (PIN 15)

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 ±5%, V_{SS} = +5V ±5%

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

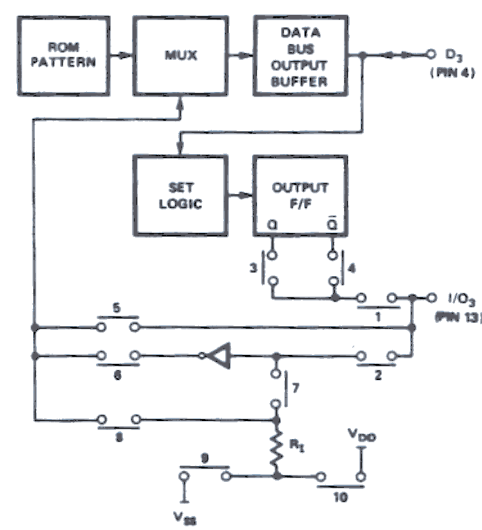


I/O_2 (PIN 14)

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 ±5%, V_{SS} = +5V ±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 ±5%, V_{SS} = +5V ±5%

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

ORDERING INFORMATION

MCS-4

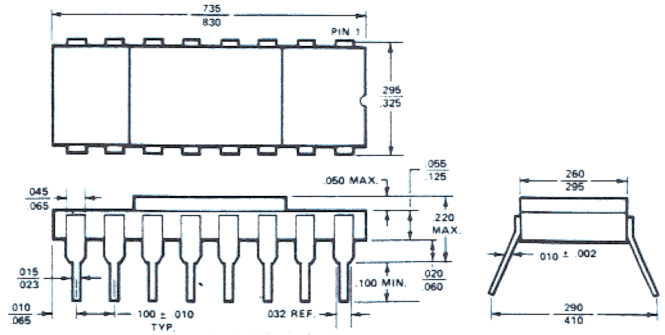
1. The 4004 (CPU) is available in ceramic only and should be ordered as C4004.
2. The 4001 (ROM), 4002 (RAM) and 4003 (SR) are presently available off the shelf in plastic only. Standard devices should be ordered as follows:
   - P4001 Plastic Package
   - P4002-1 (Metal Option #1) – Plastic Package
   - 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. **PA4-04 Program Analyzer**
   Complete MCS-4 data bus activity may be monitored. To order, specify PA4-04.
6. **Intellec 4**
   The Intellec 4 and microcomputer modules must be specified individually by product code:

imm4-40A	Complete Intellec 4 with PROM programmer
imm4-42	Central Processor – includes CPU, four 4002s, sockets for four PROMS, I/O Ports and crystal clock
imm4-22	Instruction/Data Storage – includes four 4002s, sockets for four PROMS, and I/O Ports
imm4-24	Data Storage – includes four 4002s and capacity for twelve additional 4002s
imm8-28	PROM Memory – includes sockets for sixteen 1702A PROMs
imm4-60	Input/Output – 8 input and 8 output ports
imm8-78	1702A PROM Programmer
imm8-70	Universal Prototype Module
imm8-72	Module Extender

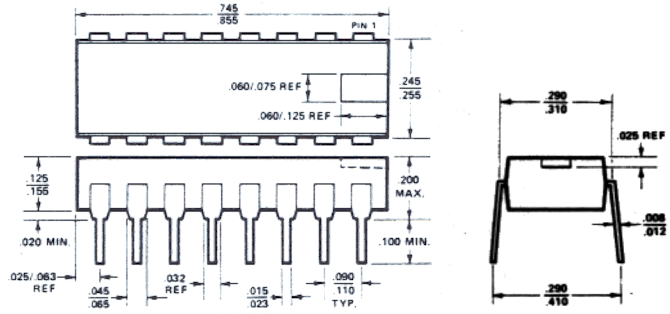
7. **MCS-4 Cross Assembler and Simulator Software Package**
   This software package converts a list of instruction mnemonics into machine instructions and then simulates the operation of the MCS-4 program. These programs are written in FORTRAN IV and are available via time-sharing service or directly from Intel.

PACKAGING INFORMATION

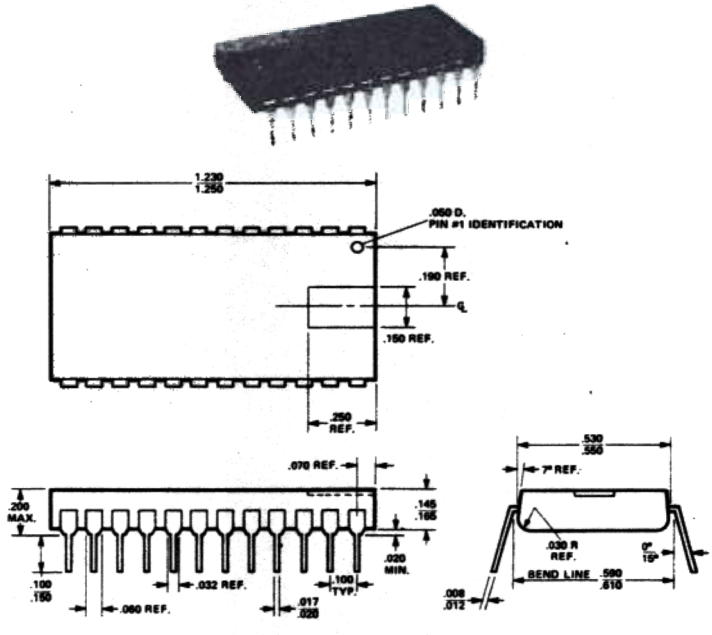
16-LEAD CERAMIC DUAL IN-LINE PACKAGE OUTLINE



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 $A_3 A_2 A_2 A_2, A_1 A_1 A_1 A_1$ (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 $D_3 D_2 D_1 D_0$		Fetch immediate (direct) from ROM Data D_3, D_1 to index register pair location RRR.[2]
SRC	0 0 1 0		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			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 ≠ 0, otherwise skip (go to the next instruction in sequence).
ADD	1 0 0 0		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		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		0 0 0 0	Write the contents of the accumulator into the previously selected RAM main memory character.
WMP		0 0 0 1	Write the contents of the accumulator into the previously selected RAM output port. (Output Lines)
WRR		0 0 1 0	Write the contents of the accumulator into the previously selected ROM output port. (I/O Lines)
	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 4002/4003 only)
	1 1 1 0	0 1 0 0	Write the contents of the accumulator into the previously selected RAM status character 0.
WR1[4]		0 1 0 1	Write the contents of the accumulator into the previously selected RAM status character 1.
WR2[4]		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.
	1 1 1 0	1 0 0 0	Subtract the previously selected RAM main memory character from accumulator with borrow.
RDM		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 0 1 1	Add the previously selected RAM main memory character to accumulator with carry.
RD0[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		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®

intel Microcomputers. First from the beginning.

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